Integrating ColdFusion and Flash

U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

September 2001 Volume: 3 Issue: 9

Sept. 24–25, 2001 New York, NY
ColdFusion fasttrack
See Program Inside 12

Sept. 24–25, 2001 New York, NY
web services EDGE conference & expo
Oct. 24–25, 2001 Santa Clara, CA 41

Oct. 22–25, 2001 Santa Clara, CA
XML EDGE conference & expo 2001 49

Creating a Custom XML Editor in ColdFusion

by Kelly Brown
page 26

CFDJ Feature: ColdFusion-Driven Flash Content Part 2 6
Integrate application logic with dynamic visuals — R. Drisgill & J. Montilla

Foundations: Fusedoc with XML Vision 14
Fusedoc brings clarity to code maintenance — Hal Helms

CFDJ Feature: SQL Server 2000 Meets XML 18
The potential uses for new XML-based features — Simon Horwith

<BF> on <CF>: Probe Your Servers 24
Use this toolkit to correct problems when they occur — Ben Forta

CF Techniques: Using XML Scalable Vector Graphics with CFML 34
Create dynamic charts and diagrams — Christian Schneider

CFDJ Feature: Using XSLT to Deliver CF Apps 38
Add support for additional wireless devices — P. Elisii & S. Ohotin

Application Integration: CF and XML-RPC Part 1 48
Connect applications built in different technologies — Ronald West

CFDJ Feature: Building VoiceXML Applications 52
Using ColdFusion to implement dynamic voice apps — Ben Parson

Editorial
They're Baaack…
Robert Diamond page 5

Product Review
Activedit 2.5 by CFDev.com
Carey Lilly page 46

What's Online
page 58

Letters to the Editor
page 59

CFDJ News
page 62

SYS-CON MEDIA

ABLECOMMERCE

www.ablecommerce.com

CORDA TECHNOLOGIES

www.corda.com

ACTIVEPDF

www.activepdf.com

COLDFUSION Developer's Journal

international advisory board

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

department editors

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

writers in this issue

Kelly Brown, Robert Diamond, Randy H. Drisgill, Paul Elisii, Ben Forta, Hal Helms, Simon Horwith, Carey Lilly, Jason Montilla, Serge Ohotin, Ben Parson, Christian Schneider, Ronald West

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 © 2001 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.

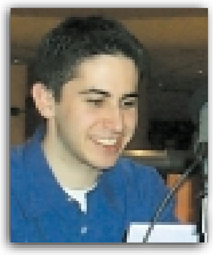
For promotional reprints, contact reprint coordinator:
Carrie Gebert carrieg@sys-con.com

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.

They're Baaaack...



BY **ROBERT DIAMOND**

Last year's Readers' Choice Awards greatly exceeded our expectations. Over 13,000 votes were cast, many more than we imagined. What it proved to us here at ***CFDJ***, and what it demonstrated to the industry as a whole, was the maturity of the ColdFusion industry. It also showed your devotion, as developers, to the products and companies that help you the most.

The products you use daily and that cut down on the time, energy, and money it takes to develop the world's leading Web applications were rewarded with awards – picked by you and presented by us. Those that didn't strike your fancy – well, they didn't fare quite as well. To view the results and our coverage of last year's awards, visit the wrap-up at www.sys-con.com/coldfusion/article.cfm?id=274.

That was last year's Readers' Choice Awards; this year they're back – bigger and better. We have more categories, more products, and more companies represented in the voting, which began on September 1. I'm proud to say that this increase mirrors the continued growth and acceptance of the ColdFusion industry. The awards and the magazine have matured along with the product and the market. Reaction to ColdFusion 5 has been quite good, not just by users of ColdFusion but by the mainstream computer press as well. Check out Macromedia's CF 5 Web site at www.macromedia.com/software/coldfusion/ and the Reviews section if you want to feel good about your choice of Web application servers.

The 2001 Readers' Choice Awards will be running for the next two months on our Web site. Categories for this year's awards include Best Book, Best Consulting Service, Best Custom Tag, Best Database Tool, Best Design Service, Best E-Business Software, Best Education and Training, Best Testing Tool, Best Web Development Tool, Best Web Hosting, Best Web Site, Best Web Application, and Most Innovative CF Application. To vote in this year's awards – visit www.sys-con.com/coldfusion/readerschoice2001. Make your voice heard...this is your chance to show what you think.

This Month's Issue

This month's focus is on XML – and the articles we have lined up will not disappoint. Kelly Brown has written about creating a custom XML editor in ColdFusion. You'll learn the basics of XML and how to edit XML data files in CF. Christian Schneider has a piece on using the XML language with SVG – Scalable Vector Graphics. SVGs are a new technology, a new graphics format you can dynamically generate using CF to make charts and diagrams. Continuing with our XML coverage – Ronald West talks about the XML-RPC. In Part 1 of a multipart feature he lays the foundation for creating an application that ties together Java, ColdFusion, XML, and PHP. It's a must-read. Hal Helms shows the Fusebox approach to XML, and Randy Drisgill and Jason Montilla discuss how to display ColdFusion-selected information in a Flash movie. And, as always, guru-extraordinaire Ben Forta contributes his thoughts.

It's a great issue – and if you haven't voted yet – go online and do it today!

Robert Diamond

ABOUT THE AUTHOR
Robert Diamond is editor-in-chief of **ColdFusion Developer's Journal** *as well as* **SYS-CON**'s *newest magazine,* **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@SYS-CON.COM

ColdFusion-Driven Flash Content

Exciting new ways of integrating application logic with dynamic visuals

CFDJ FEATURE

BY **RANDY H. DRISGILL** AND **JASON MONTILLA**

The current state of interoperability between Macromedia's Flash 5.0 and ColdFusion 4.5 was covered in Part 1 of this article (***CFDJ***, Vol. 3, issue 8). We also discussed the basics of programming a simple Flash application that would display information from a text file or a dynamic ColdFusion file.

Part 2 skips beyond the basics of Flash programming and describes how to use Flash to insert and select information from a database using ColdFusion and then display the results in a Flash movie. We'll assume you've learned some things from Part 1, so if you didn't have a chance to read it, you may have to spend some time in the Flash Help screens.

Let's discuss the basic architecture of this application. It begins with a couple of Flash buttons that allow the user to select either "View Records" or "Add New Record." If the user selects the former, Flash will call a ColdFusion template that will query our contacts database and return a list of all the rows. Alternatively, if they select the latter, the user will be presented with a Flash form to enter a first and last name. When the user clicks "submit" on this form, Flash will send the variables over to ColdFusion, which in turn adds the value to our contacts database. From the end-user's perspective, he or she will never leave the Flash movie that was first loaded. All our menus and results will be displayed seamlessly in Flash.

So, from this description, we know we will need a database. Create a database that has a table named *contacts*, with "firstname" and "lastname" fields. Fill it in with some sample first and last names, and create an ODBC connection to it named contacts. Since this example is only a basic representation of exchanging data between Flash and ColdFusion, the database is simple. A more complex application can be developed using the same methods.

From an application standpoint we'll be preparing three distinct but separate parts that will all work together to act as one complete solution. These three concepts are *ActionScript programming*, *ColdFusion programming*, and a *logical Flash movie frame flow* that will include all the visual objects the user will interact with. We start by creating the flow of the Flash movie and adding all the visual elements in their appropriate frames. After that we attach ActionScript commands to certain frames and objects to complete the Flash movie flow. Finally we create two ColdFusion templates that will display and input database records.

So, let's set up the basic structure of our Flash movie. Open up Flash 5.0 and create three layers: a top layer named *labels*, a

middle layer named *code*, and bottom layer named *data*. We use the labels layer to name each of the functional sections of our Flash movie; the code layer will contain all of our ActionScript code, and the data layer will hold all our visual objects. Next we create 30 frames to give us some room to spread out our work.

Now let's create the keyframes where the actions will take place. Place the keyframes in the following frames of all of the layers: 6, 11, 16, 21, and 26. These will mark off the functional areas of the application. We also create keyframes for some specific programmatic events on the code layer at frames 5, 8, 9, 10, 15, 20, 23, 24, 25, and 30. Now to prevent ourselves from going crazy trying to find different sections of the movie, let's add some labels to the labels layer. We set the frame label by using the Label property of the Frame Panel and create the following labels:

- Frame 1 label, Start
- Frame 6 label, Loading Records
- Frame 11 label, Display Records
- Frame 16 label, Input Record
- Frame 21 label, Inputting Data
- Frame 26 label, Input Success

Finally, let's lock the labels and the code layers to make sure no objects can be accidentally placed on them. This can be accomplished by simply clicking the dot underneath the little lock icon on the actual layer of the Flash movie. We now have the stage completely set up for our Flash movie. Study the screenshot in Figure 1 and make sure your screen looks similar.

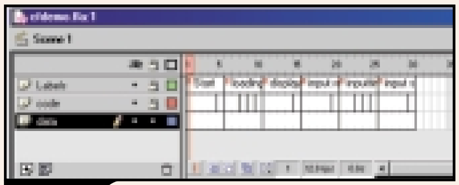


FIGURE 1: Scene one

Now we build all the visual elements of the Flash movie by labeled sections. For each one, make sure you're on the first frame of the data layer for that particular section.

- ***Start:*** There will be a static text field with the text "What would you like to do?" Under that will be two buttons labeled "View Records" and "Add New Record." Remember we can create buttons by simply selecting objects, clicking F8, and selecting "Button."
- ***Loading Records:*** There will be a static text field with the text "Loading Data."
- ***Display Records:*** There will be a static text field with the text "Database Records," and under that a dynamic text box set to "Multiline," "Word wrap," "HTML," and "Selectable" with a "Variable" assigned to it named "records" (see Figure 2). Also, we place a "Back" button at the bottom for navigation (make sure you select the shape you're using, then click F8 and select "Button" to make it an actual button).

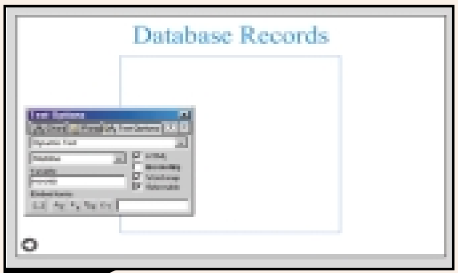


FIGURE 2: Database records

- ***Input Record:*** There will be a static text field with the text "Input Record," and under that two static text fields with the text "First Name" and "Last Name," one above the other (see Figure 3). Beside each of these will be an input text box (each with variables named "fname" and "lname" appropriately). Set each of these to "Input Text," "Single Line," "Selectable," and "Border/Bg". Under this we place a "Submit" button and a "Back" button for navigation.

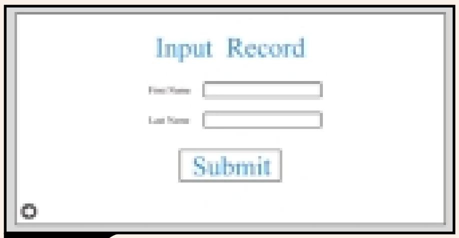


FIGURE 3: Input record

- ***Inputting Data:*** There will be a static text field with the text "Inputting Data."
- ***Input Success:*** There will be a static text field with the text "Database Updated" and a "Back" button for navigation.

This sets up all the visual elements, so now we can go through and add ActionScript code to the various buttons. These ActionScripts will be placed on the buttons, not the code layer. For each of these, right-click on the button and select "Actions." This will open the ActionScript editor in "Default Mode." Be sure to change to the "Expert Mode" and enter the code as it's shown here. For each of them, Flash will wait for either a mouse click or, in the case of the "back" button, a keypress. Once this interaction is detected, the movie immediately jumps to the labeled section specified in the "gotoandplay" field.

- ***Start section:*** "View Records" button

```
on (release) {
gotoandplay ("Loading Records");
}
```

- ***Input Record section:*** "Input Record" button

```
on (release) {
gotoandPlay ("Input Record");
}
```

- ***Input Records section:*** "Submit" button

```
on (release) {
gotoandPlay ("Inputting Data");
}
```

- ***Back buttons:*** All of them (see Figure 4)

```
on (release, keypress "<Enter>") {
gotoandPlay ("Start");
}
```

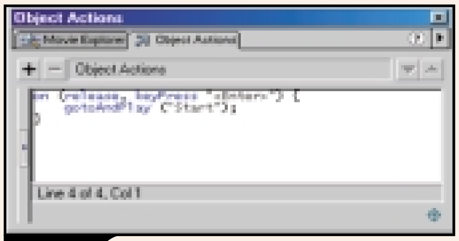

```
on (release, keyPress "<Enter>") {
    gotoAndPlay ("Start");
}
```
Line 4 of 4, Col 1

FIGURE 4: Object actions

This covers all the ActionScript for the buttons; we can now start coding the actual application logic using ActionScript in the code layer of the movie. Make sure you're on the code layer, and double-click the appropriate frame referenced below to open up the ActionScript editor. Remember to switch to Expert Mode and enter the code shown in the following section.

Start Section

In Frame 5 we set "datacomplete" to "no" and initialize "records," "fname," and "lname" to nothing (see Figure 5). These are all variables that we use throughout the Flash movie that need to be set to nothing, otherwise Flash won't know how to display them. Last, we do a "stop();" to hold the movie until user interaction.

```
datacomplete = "No";
records = "";
fname = "";
lname = "";
stop ();
```

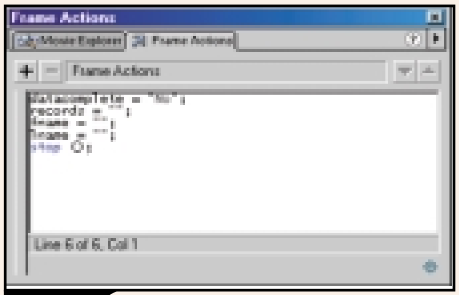

```
datacomplete = "No";
records = "";
fname = "";
lname = "";
stop ();
```
Line 6 of 6, Col 1

FIGURE 5: Frame actions

Loading Records Section

In Frame 8 we load the variables from the ColdFusion select query using loading.cfm as the URL parameter. The location parameter we can leave blank (it defaults to where we're currently located in the Flash movie). The variables parameter will use "POST," which is necessary to force Flash not to cache the results of the ColdFusion page.

```
loadVariables ("loading.cfm", "",
"POST");
```

Frame 10 will test for the datacomplete variable to equal "Yes." We utilize this because Flash processes data quickly and may actually try to continue processing before our variables are completely loaded. Since the datacomplete variable is the last variable we'll be sending back from our ColdFusion template, when Flash reads this value as "Yes," we know the entire variable string has been read. If it has, we go to the "Display Records" section; otherwise our code will loop back to Frame 9 (this is why we kept Frame 9 blank, so that Flash can continually loop between the two until it's done).

CFXHOSTING
www.cfxhosting.com

```
if (datacomplete == "Yes") {
 gotoAndPlay ("Display Records");
} else {
 gotoAndPlay (_currentframe-1);
}
```

Display Records Section

Frame 15 simply has a stop (); action because this is where the results are displayed. Flash will wait until the user clicks the "Back" button to take users out of the stop condition and bring them back to the "Start" section.

```
stop();
```

Input Record Section

Frame 20 also has just a stop (); command. This is where we begin to add a new record to the database. The "stop" command holds Flash until the user does something.

```
stop();
```

Inputting Data Section

Frame 23 has a loadvariables command that works similar to the one in Frame 8 except we're using submitting.cfm. This is the ColdFusion template that will actually insert the new record into the database.

```
loadVariables ("submitting.cfm", "",
"POST");
```

Frame 25 is exactly like Frame 10, except we go to the Input Success section on success, rather than the Display Records section. Once again we utilize the datacomplete variable to tell us that ColdFusion has finished processing the data.

```
if (datacomplete == "Yes") {
 gotoAndPlay ("Input Success");
} else {
 gotoAndPlay (_currentframe - 1);
}
```

- ***Input Success Section***

In Frame 30 we stop(); and wait for user input via the "Back" button.

```
stop();
```

This is all we need to do from the Flash side of things; we can save the file as "cfdemo.fla", export a movie file called "cfdemo.swf", and place them both in the same directory we plan to place the ColdFusion files in . Now the last thing left to do is create the ColdFusion templates to send and receive information between the Flash movie and the database. First we have to remember that for any ColdFusion template that's created for use with Flash, we *must* include the following line:

```
<cfsetting enablecfoutputonly="Yes"
showdebugoutput="No" catchexceptions-
bypattern="No">
```

This will be used to turn off any output other than that in a <cfoutput> tag and will suppress any debug information for just this one page. This is important because the debug info and all the white-spaces that ColdFusion creates will instantly break a Flash application. Also, we should remember that Flash expects the output of these files to be a URL string of variables, and again that spaces and carriage returns are forbidden.

After we create the ColdFusion files, it's sometimes helpful to test them by calling them straight from a Web browser, passing in any necessary URL variables. This way we can view the source on them and make sure there are no stray spaces or carriage returns. The following are the ColdFusion templates and explanations of what each will be doing. Be sure to place them in the same directory as the Flash movie.

- ***loading.cfm***

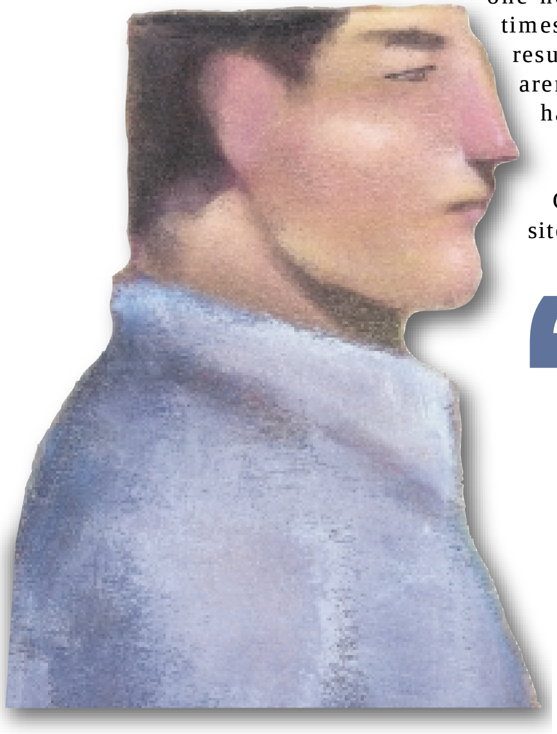
After the <cfsetting>, we make a database query to retrieve the "FirstName" and "LastName" values from the database. We should note that the Flash variables we're using are called "fname" and "lname"; this name difference helps us to specifically control how ColdFusion and Flash will utilize these variables, otherwise some things would happen automatically due to the way Flash makes use of variables.

We initialize the records variable using a <cfparam>, and then loop through the data creating a list of lastname, firstname, and <br> (the query is outputting them in alphabetical order by last name). Then we <cfoutput> them all with "&records=# urlencodedformat(records)#". The ampersand is used because Flash expects all variables to be set with an ampersand before them, and we urlencode the string since Flash expects one long URL string of variables. We also include the "&datacomplete=Yes" at the end of the string to tell Flash that we're done (see Listing 1).

- ***submitting.cfm***

After the <cfsetting> we do a simple insert query to insert "fname" and "lname" into the "firstname" and "lastname" columns of the table. Luckily Flash automatically submits variables in a format that ColdFusion can automatically use. Next we simply <cfoutput> "&datacomplete=Yes" so that we know the input was successful. If we wanted to, we could later get creative and actually send database errors back in the URL string, so that we can display them in Flash as well (see Listing 2).

This is all we need to do; try it out for yourself. You should test ColdFusion-enabled Flash applications like this one outside of the Flash environment in an actual Web browser, because Flash doesn't have the capabilities to test the ColdFusion Web-enabled content. Also, one note about cached results: sometimes we've noticed that IIS caches results, therefore dynamic changes aren't reflected immediately. If you have any problems with your display not changing when you *know* you changed the text or ColdFusion file, change your Web site preferences in IIS under "HTTP



"Flash is a picky animal when it comes to sending and receiving data"

Headers" and select "Expire Content Immediately." This should allow your pages to update dynamically again.

This application is fairly simple, but it should give you a good idea of some of the cool things you can do with Flash and ColdFusion. If you want to take your knowledge further, you can experiment with dynamic graphics using Macromedia Generator, or try to utilize XML or WDDX to import structured data into Flash from ColdFusion.

As Macromedia takes control of the ColdFusion product development efforts, we can be sure to see some exciting new ways of integrating the application logic with dynamic visuals. Above all, remember that Flash is a picky animal when it comes to sending and receiving data. If you have any problems with the examples, just stick with them and test each part individually for syntax errors. With a little luck, you now have the hang of ColdFusion-driven Flash content and are ready to make your own applications.

About the Authors

Randy H. Drisgill is the CTO and cofounder of Vshift (a premier consulting partner). An Allaire Certified Professional, he's been using ColdFusion since 1998 and has extensive experience with data-driven Web applications, content management solutions, Web design, mobile applications, and data-driven Flash applications.

Jason Montilla, an Allaire Certified Professional, works for Vshift. He's been using ColdFusion for two years and Flash for four, and is currently building several Web and mobile data-driven Flash applications.

rdrisgill@vshift.com

jmontilla@vshift.com

Listing 1

```
<cfsetting enablecfoutputonly="Yes" showdebugoutput="No"
catchexceptionsbypattern="No">

<cfquery name="getData" datasource="Contacts"
dbtype="ODBC">
 SELECT FirstName, LastName FROM Contacts ORDER BY
LastName;
</cfquery>

<cfparam name="records" default="">

<cfloop query="getData">
 <cfset records = records & LastName & ", " & FirstName
& "<br>">
</cfloop>

<cfoutput>&records=#urlencodedFormat(records)#&datacom-
plete=Yes</cfoutput>
```

Listing 2

```
<cfsetting enablecfoutputonly="Yes" showdebugoutput="No"
catchexceptionsbypattern="No">

<cfquery name="getData" datasource="Contacts"
dbtype="ODBC">
 INSERT INTO Contacts (FirstName, LastName) VALUES
('#Trim(fname)#', '#Trim(lname)#');
</cfquery>

<cfoutput>&datacomplete=Yes</cfoutput>
```

CODE LISTING

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

DTN FINANCIAL SERVICES

www.finwin.com

COLDFUSION fasttrack

The ONLY ColdFusion Event Backed by the Power of SYS-CON MEDIA and COLDFUSION Developer's Journal

Cf COLDFUSION fasttrack

REGISTER ONLINE TODAY! Save $300

Save the Dates

Prepare for ColdFusion Certification

Kevin Lynch

Kevin Lynch is president of Macromedia Products. He joined Macromedia in 1996 and has been instrumental in forming its Web strategy. As president of products, Kevin is responsible for developing Macromedia's award-winning family of software and solutions.

Jeremy Allaire

As Chief Technology Officer, Jeremy is instrumental in guiding Macromedia's product direction and is the company's primary technology evangelist, responsible for establishing key strategic partnerships within the Internet industry. Jeremy was the founder and former chief technology officer for Allaire Corporation, which merged with Macromedia in March 2001.

Ben Forta

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 4.0 Development, *as well as* Allaire Spectra E-Business Construction Kit *and* Sams Teach Yourself SQL in 10 Minutes. *He recently released* WAP Development with WML and WMLScript.

Prepare for the ColdFusion Certification Exam with Comprehensive Technical Sessions Designed for Beginner and Advanced ColdFusion Developers

Session Topics Include:

- Web Fundamentals
- Application Development
- Database Concepts
- Client State Management
- Troubleshooting, and A Special Focus on ColdFusion 5.0.

Plan to Attend

webServices JOURNAL · JAVA DEVELOPER'S JOURNAL · XML JOURNAL · wireless · LINUX BUSINESS WEEK · COLDFUSION Developer's Journal · WebSphere DEVELOPER'S JOURNAL · Dr.Dobb's · webtechniques

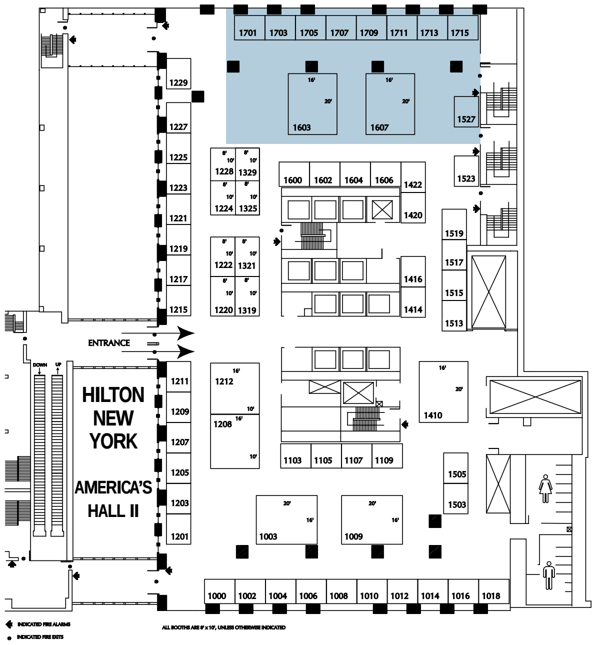
Sponsored by CF Advisor™ & ColdFusion Developer's Journal

HILTON NEW YORK, NYC, SEPTEMBER 24-25

COLDFUSION PAVILION

Hilton New York, NYC
September 24-25

Exhibit at the "COLDFUSION PAVILION"



Supported by a **Macromedia ColdFusion FastTrack & Certification Program,** the **"CF Exhibit Pavilion"** will be sponsored by **SYS-CON Media's** market-leading **ColdFusion** publication, ***ColdFusion Developer's Journal.***

Don't miss your best and only opportunity to reach **ColdFusion** experts and just-certified professionals.

Join ***ColdFusion Developer's Journal*** and bring your **ColdFusion** products, services, and solutions to **3,000+ business** and **technology professionals.**

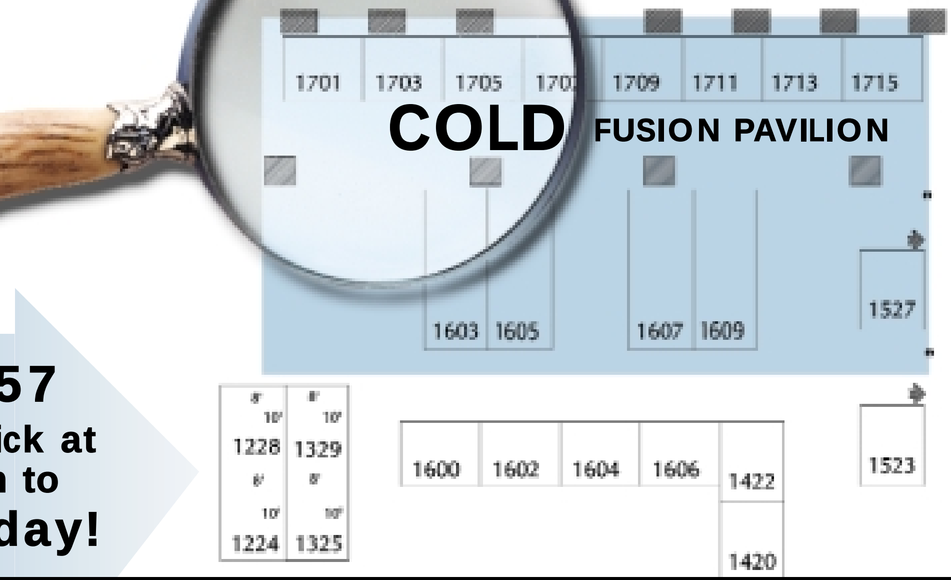
Colocated with

JDJ EDGE conference & expo

&

web services EDGE conference & expo

www.sys-con.com



Call 201.802.3057

or e-mail Michael Pesick at michael@sys-con.com to

Reserve Space Today!

Owned and Produced by
SYS-CON EVENTS

135 Chestnut Ridge Road, Montvale, NJ 07645
(201) 802-3069 Fax: (201) 782-9051
visit
WWW.SYS-CON.COM
for more information

NEITHER SYS-CON MEDIA NOR SYS-CON EVENTS, INC., ARE SPONSORS OF, OR ARE AFFILIATED IN ANY WAY WITH, CAMELOT COMMUNICATIONS, INC., OR ANY EVENTS PRODUCED BY CAMELOT COMMUNICATIONS, INC.JAVA AND JAVA-BASED MARKS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF SUN MICROSYSTEMS, INC., IN THE UNITED STATES AND OTHER COUNTRIES. SYS-CON PUBLICATIONS AND SYS-CON EVENTS ARE INDEPENDENT OF SUN MICROSYSTEMS.

Fusedoc with **XML Vision**

Bringing clarity to code maintenance

BY
HAL HELMS

The U.S. Army used to have a recruiting campaign featuring colorful posters of exotic locales (populated by welcoming people) with large, bold type that said, *"See the world. Join the Army."*

I didn't exactly join the Army. Instead I received a personal invitation from Uncle Sam – a command performance of sorts. Back then it was called "being drafted."

But no matter how a recruit got into the Army, the exciting destinations would have been more exciting if "vehicle" (translation: truck) maintenance or "policing" the grounds (translation: garbage pickup) weren't part of the daily regimen. It didn't matter whether the Army compound was nestled in the plains of Texas or amid the cathedrals (and beer halls!) of Germany; I discovered the truth behind the old saying, "Unless you're the lead dog, the view is pretty much the same."

Now while the IT world doesn't use the same marketing techniques – in this case coders hammering away at their laptops with idyllic scenes of the Swiss Alps in the background – it does promise the heady excitement of working on large, exciting projects where you'll get to use all the cool technology you read about.

Many people assume they'll be using UML to transform use cases into state diagrams that they will then use to write EJBs to, for example, data mine hidden information about customer buying trends – bringing them eventual power, glory, and riches. More often, though, they find themselves doing drudge work – maintaining code (someone else's code, in fact) – and, on bad days, cursing the day they ever entered programming.

There's good news and bad news on this front. The bad news is this: code maintenance is what it's all about – forget the hokeypokey. Looking at the life-cycle cost of a piece of code, as much as 90% of it lies in maintaining the old stuff. We're not going to escape this as long as there's code to maintain. (In fact, I suspect that one of the allures of new languages is that there's no old code to maintain!)

More than once I've groused about the idiot who left me a particular piece of code to maintain only to find that I was the idiot who wrote that particular bit of code. And this is where the good news comes in: we can use a standardized system that both forms a sort of contract (telling the initial coder what he or she is responsible for) and provides guidance to the maintenance coder on what the code does. If your job involves maintaining code, take a look at Fusedoc.

I've written about the system of Fusedoc before. As part of the Fusebox methodology (and in anticipation of the October launch of the version 3 specification of Fusebox), Fusedoc 2.0 has just been released and it has caught the XML vision.

Like the prior version, Fusedoc 2.0 is set in a ColdFusion-style comment at the top of each code page. The entire Fusedoc is contained in an XML packet with a root element of <fusedoc>. Subelements of <fusedoc> are <responsibilities>, <properties>, and <io>. (A fourth section, <assertions>, is specified for future development. This will allow coders to provide assertions about pre- and postconditions that will be enforced at runtime. A DTD is available at www.halhelms.com.)

The <responsibilities> section contains a plain-language statement of what the code is responsible for. See Listing 1 for an example of a page that validates a user login against a database.

Notice that I haven't said anything about what the structure, CurrentUser, should consist of. This is deliberate: responsibilities shouldn't repeat what will be covered by another section. In this case the structure of CurrentUser will be denoted in the <io> section. (If you're not familiar with the concept of XFAs, this is a Fusebox technique for making code

reusable by identifying exit points of a code file – conditions that would cause a new HTTP request – and assigning variable names [XFAs] to them. The values for XFAs are set outside the code file and can be changed to reflect the context the code file is being used in.)

After the responsibilities of the code file have been set down, you may want to include other information about the code file, such as who wrote the file, who revised it, and what the revisions were. All of this information is captured in the properties section. In fact, the only suggested element for <properties> is <history>: a method of keeping track of who did what to the code.

Other than that, the kind of information that is encoded is wide open – all that's required is a name/value pair. Some shops assign a complexity rating to an individual code file so coders can work at the level they're comfortable with. Listing 2 is an example of <properties> for that same act_validateLogin.cfm file.

Next comes the <io> section, the area I find to be of greatest help to maintenance coders (*io* stands for input/output and this section tells the coder at a glance what variables are passed into the fuse; what variables, files, or cookies are globally available; and what the coder is responsible for setting). Allowed subelements of <io> are <in>, <out>, and <pass-through>.

The in and out elements are straightforward enough. Variables passed into the code belong in the <in> section and variables passed out should be set in the <out> section, but the <passthrough> section may not be immediately obvious. In fact, <passthrough> is really just shorthand for saying, "You're going to get a variable passed into your code and you are responsible for passing it along – without changing it." One obvious example of passthrough variables would be the hidden fields in multipart forms that pass information gathered in an earlier form page to later pages.

While none of this is conceptually difficult (it's just straight XML), learning the allowed subelements (and the respective attributes) requires some work.

Listing 3 gives the <in> subelement of the <io> section of act_validateLogin.cfm.

The XML tells us we're going to get two local XFA variables and a username and password that will appear as attributes. (Fusebox automatically copies form and URL variables into the attributes scope.) It then says that we're going to have a local recordset – the result of a query – that will have four columns, and it goes on to describe those columns.

The <out> section is a little more interesting (see Listing 4).

This code tells us we're going to create a URL variable called *fuseaction* whose value will be one of the XFAs passed into the code. It then says we're responsible for creating a WDDX'd client structure called *CurrentUser* if the user was validated, and it gives us instructions on what the keys and values for that structure should be.

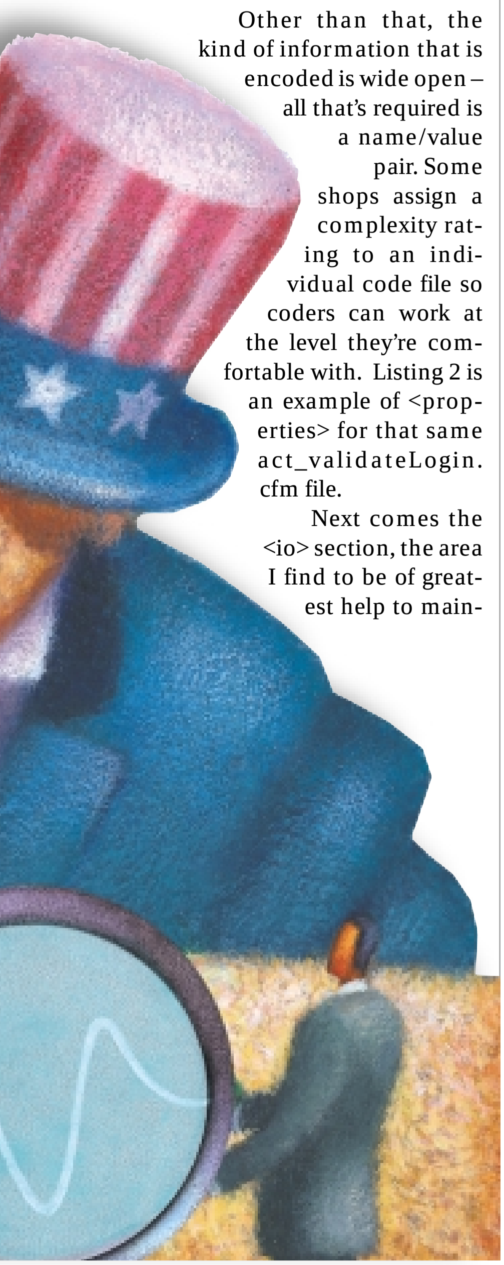
If you're wondering how you're going to remember all of those elements and attributes, you'll appreciate this: there are Studio .vtm files to help with writing Fusedocs. They work just like all of the tag editors built into Studio and will dramatically reduce the time needed to master the Fusedoc specification. These files are available for downloading and come with excellent installation instructions from www.fusebox.org.

Fusedocs work best when they're written before the actual code. In such cases they serve as a sort of program definition language (PDL) that makes communication between architect and coder clear and concise. To the maintenance coder, Fusedocs can help maintain sanity under the stressful conditions of working on old code. Too often, maintaining old code resembles mind reading more than problem solving. Fusedocs can change that.

What's really exciting is the possibility that XML Fusedocs can be used by automated tools to do things like autogenerate documentation and provide runtime validation.

That's cool stuff that's in the works to help maintenance coders, but none of it is a magic wand. It won't make maintenance coding the next big thing, but adoption of Fusedoc can make a dramatic impact on the speed and maintainability of code.

I believe it was Miss Manners who said, "If the written word alone were enough, we wouldn't have universities attached to libraries." For the written word alone, you can head over to www.halhelms.com. But here's a better idea: the second annual Fusebox Tutorial Conference is going to be held on October 20, the Saturday before Macromedia's DevCon starts. The conference will be held at the Disney Hilton – just across from the Disney Dolphin, which is the DevCon conference hotel. Join other developers to hear all about the new, XML-based Fusedoc specification along with a host of new features that make up release 3 of Fusebox. In addition to in-depth tutorial sessions, Jeremy Allaire will be the keynote speaker. For more information, go to www.fusebox.org. See you in Orlando!



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

HAL.HELMS@TEAMALLAIRE.COM

Listing 1

```
<!---
<fusedoc fuse="act_validateLogin.cfm">
 <responsibilities>
 I check to see if the userName and password passed to me
have a match in the USER table. If so, I create a struc-
ture called “CurrentUser” and initiate a fusebox request of
XFA.successfulLogin. Otherwise, I return to the fusebox
with XFA.failure.
 </responsibilities>
</fusedoc>
--->
```

Listing 2

```
<properties>
 <history
  author="Hal Helms"
  email="hal.helms@teamallaire.com"
  date="4 August 2001"
  role="architect"
  type="create"
  comments="Not all of these attributes are required." />
 <property name="complexity" value="1" />
</properties>
```

Listing 3

```
<in>
 <string name="XFA.successfulLogin" scope="variables" />
 <string name="XFA.failure" scope="variables" />
 <string name="userName" scope="attributes" />
 <string name="password" scope="attributes" />
 <recordset
  name="UserInfo"
  primarykeys="userID"
  scope="variables">
  <number name="userID" precision="integer" />
  <string name="firstName" />
  <string name="lastName" />
  <number name="userGroups" precision="integer" />
 </recordset>
</in>
```

Listing 4

```
<out>
 <string name="fuseaction" comments="an XFA" scope="url" />
 <structure
  name="CurrentUser"
  onCondition="on successful validation"
  format="WDDX"
  scope="client"
  optional="true"
  comments="key values match column values from UserInfo
recordset">
  <number name="userID" precision="integer" />
  <string name="firstName" />
  <string name="lastName" />
  <number name="userGroups" precision="integer" />
 </structure>
</out>
```

CODE
LISTING

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

PAPERTHIN

www.paperthin.com

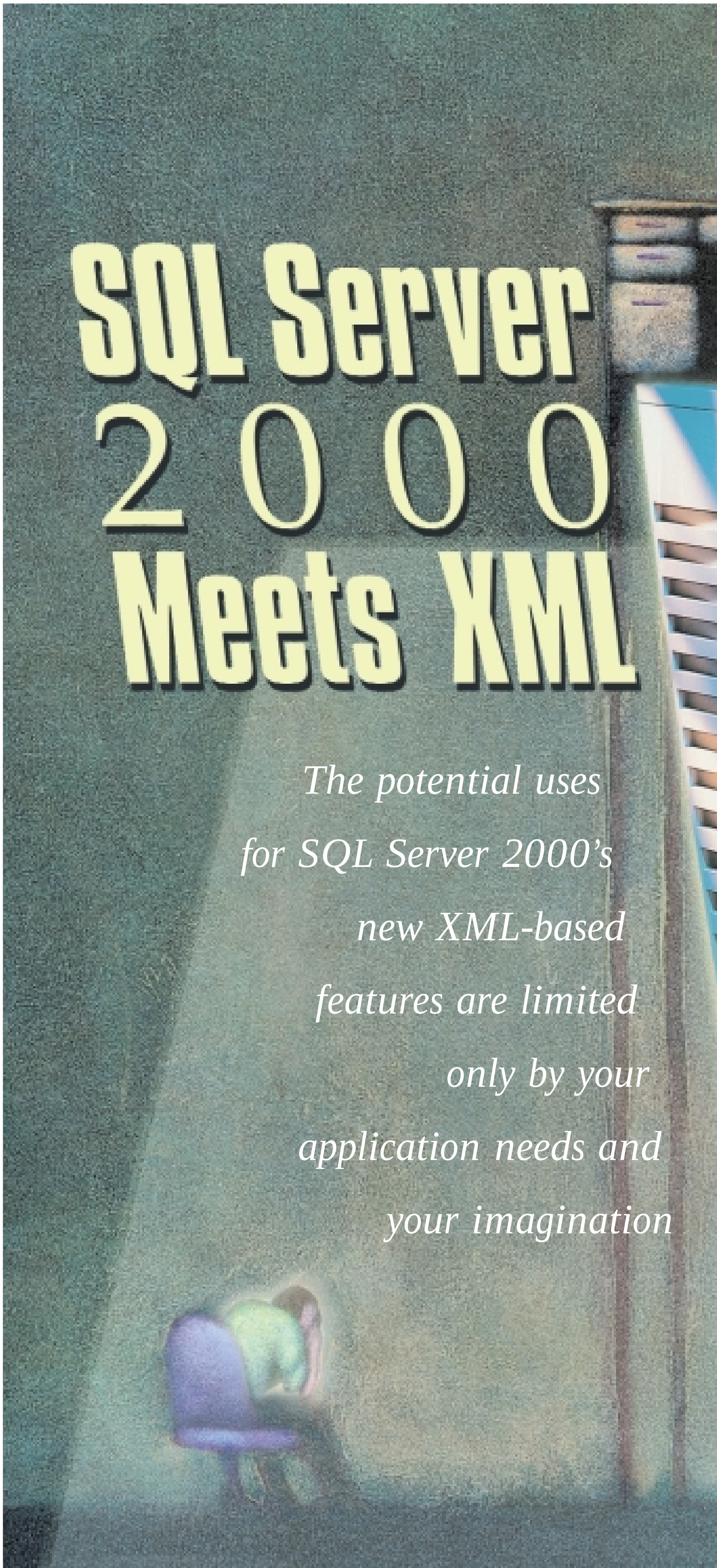
FUSEBOX

www.fusebox.com

BY SIMON HORWITH

SQL Server 2000 Meets XML

The potential uses for SQL Server 2000's new XML-based features are limited only by your application needs and your imagination



The overwhelming majority of ColdFusion applications on the Web and in our intranets are tightly integrated with a relational database management system of some sort. Because of its low cost, ease of installation and administration, and excellent performance, Microsoft SQL Server is one of the more widely used RDBMS platforms by ColdFusion developers. The recent release of SQL Server 2000 has added many new features to the SQL Server product line, most significantly a suite of features that leverage the power of XML.

Traditionally, ColdFusion developers had three methods of working with XML within their ColdFusion templates.

1. Manually create or read XML packets by either building a string or parsing a file within a loop
2. Employ an XML parsing construct, such as the Microsoft XMLDOM COM object or a COM, Java, or CORBA object
3. Most common, leverage WDDX with the <CFWDDX> tag

These three methods of have always served ColdFusion developers fairly well, but they don't always scale well and may have other drawbacks, such as being verbose, resource-intensive, and/or too dependent on external technologies.

Finally, the common best practice of "letting the database work for you" can be extended to working with XML, as ColdFusion developers can leverage the power of their SQL Server databases when working with data in XML format.

The three new main features in SQL Server 2000 are:

1. ***FOR XML***: A Transact-SQL clause for extracting data as XML
2. ***Virtual Mappings:*** Directly access your SQL Server Database via HTTP
3. ***OpenXML:*** A T-SQL function for parsing through data formatted as XML

We will briefly examine how each of these new features works, and how they may affect ColdFusion applications that need to make use of XML representation of your database data. The examples in this article use a simple table ("States") made up of three columns ("StateID", "StateAbbreviation", and "StateName") and populated with 51 rows of data.

FOR XML

The FOR XML clause is used to return data from a SQL Server 2000 database in XML format by appending the FOR XML clause to a SQL SELECT statement. The syntax to retrieve data as XML is:

```
SELECT {columns}

FROM {table}FOR XML mode [,XMLDATA] [,ELE-
MENTS] [BINARY base64]
```

The only required argument to FOR XML is its mode, which has three possible values:

1. ***RAW:*** Returns data in XML format using default element names. Each query row is represented as a <ROW/> element, with table column names used as element tag attributes whose attribute value is the same as the column value. For example, the first row from my "States" table would be returned as <row StateID="1" StateAb-breviation="AL" StateName="Alabama"/>.
2. ***AUTO:*** Returns data in XML format using table and column names to represent both tags and attributes. The optional FOR XML argument "ELEMENTS" is used in conjunction with a FOR XML AUTO clause to denote that data is to be returned as XML. This XML uses column values inside of opening and closing column name tags, surrounded by opening and closing table name tags, to denote each query row (<tablename><column name>value </column name></tablename>).

   The preceding RAW example would look like this in FOR XML AUTO, ELEMENTS format:

```
<STATES><StateID>1</StateID><StateAbbrev
iation>AL</StateAbbreviation><StateName>
Alabama</StateName></STATES>
```

   In AUTO format without the keyword ELEMENTS specified in a query, the XML generated uses the table name for each element name, with column names and their values represented as element attributes and their respective attribute value (<tablename columnname=value columnname=value/>). The same row shown in the two prior examples would be formatted like the following when FOR XML AUTO is used in the absence of the ELEMENTS argument:

```
<STATES StateID="1"
StateAbbreviation="AL"
StateName="Alabama"/>
```

3. ***EXPLICIT:*** It's too complex an implementation to detail here; it allows for user-defined element names and element-nesting (mapping) definitions. Whenever possible, use one of the other two mode values, unless you have time to become comfortable with FOR XML EXPLICIT syntax.

In addition to the required mode parameter and the optional ELEMENTS keyword attribute, there are two more optional attributes when selecting FOR XML data; both (like ELEMENTS) can be specified only when mode is set to AUTO:

1. ***XMLDATA:*** When this keyword argument is specified, the XML packet of data is returned, preceded by an XML data definition that describes the schema, element types, and attribute types. A snippet of the data definition from the above example would look like:

```
<AttributeType name="StateID"
dt:type="i4"/><AttributeType
name="StateAbbreviation"
dt:type="string"/><AttributeType
name="StateName" dt:type="string"/>
```

2. ***BINARY Base64:*** The default attribute for AUTO mode SELECT statements; it specifies that the data should be returned in Binary Base64–encoded format. Base64 is a common content-encoding format for data sent via HTTP, which a browser will then decode and display in readable format.

If I wanted to extract all of the data from my States table as XML, I would type the following query, SELECT * FROM STATES FOR XML AUTO, into my SQL Server Query Analyzer window.20

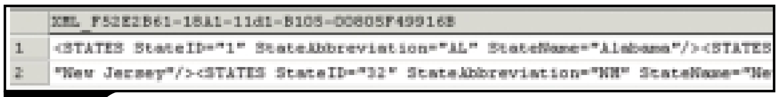
And the (partial) result in Query Analyzer would look like Figure 1.

Note that all States table records are returned as XML in a single column query result. This makes the data less difficult to store, transfer, or syndicate, and more compatible with any application (CF or otherwise) that parses XML "straight from the source." Also note that the database assigns a UUID as the name for this column. UUIDs are not valid variable names in ColdFusion because of their use of special characters. Running the same query in a CF template will result in no errors and a record count of one, but trying to output, parse, or reference the UUID named XML packet will result in an error.

To run these FOR XML queries, we use another one of SQL Server 2000's new features – Virtual Mappings – to pass the SQL statement directly to the database via HTTP by appending it to a special URL or requesting an XML file resource.

Virtual Mappings

SQL Server 2000 closely integrates with Internet Information Server to allow for direct database access via HTTP. This means that the SQL statement can be executed by users and resulting query results displayed by making a URL request. This feature is implemented by creating virtual directories and virtual names. The first thing that must be done to configure a SQL Server 2000 database to allow HTTP access is to create a virtual mapping.



```
   XML_F52E2B61-18A1-11d1-B105-00805F49916B
1  <STATES StateID="1" StateAbbreviation="AL" StateName="Alabama"/><STATES
2  "New Jersey"/><STATES StateID="32" StateAbbreviation="NM" StateName="Ne
```

FIGURE 1: Query Analyzer

A SQL Server virtual directory is no different from a Web server virtual directory, except it points to a database on the database server rather than a directory on your hard drive. To create a virtual directory, launch the Configure SQL XML Support in the IIS utility found in the SQL Server folder of your Windows start menu, as shown in Figure 2.

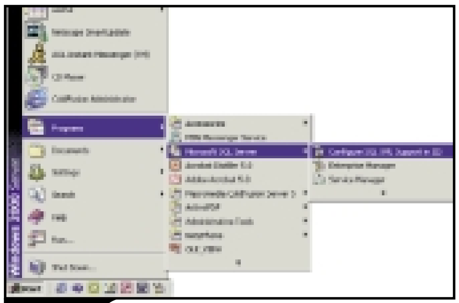


FIGURE 2: Configure SQL XML support in IIS

"Configure SQL XML Support in IIS" is a Microsoft Management Console utility that allows developers and administrators to create virtual directories and virtual names. In addition to assigning a name (alias) to the new virtual directory, you may also specify a path to a local directory if you're going to access local files (see the virtual names section of this article for more details).

In addition to assigning an alias and pointing it to a directory, before accessing the virtual name you must point it at an accessible database server and database residing on that server, and then specify the Windows OS or SQL Server–level security settings the new virtual name should use when accessing the database.

It isn't necessary to change the default settings that allow or deny user operations when accessing the database via a virtual name (see Figure 3); not doing so should be thought of as the equivalent to creating a directory off of your Web root and not verifying user permissions in that directory. While the default settings favor strong security, not verifying them is an irresponsible practice. The permissions you can set are:

- ***Allow URL queries:*** Enabling this allows SQL statements to be passed to the database directly on the URL.
- ***Allow template queries:*** The only setting enabled by default. It allows XML templates containing SQL statements to be directly called by Web browsers.
- ***Allow XPath:*** Allows users to execute XPath queries against mapping schemas.
- ***Allow Post:*** Allows HTTP post operations to the virtual directory. This is necessary for operations, such as passing form variables into stored procedures.

Virtual Names

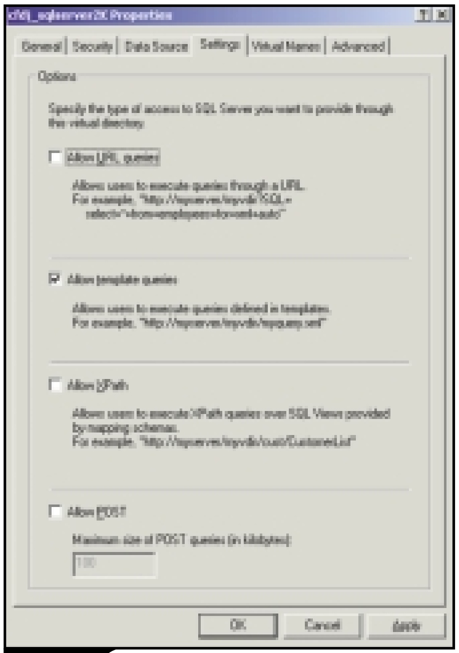
Virtual names are aliases similar to virtual directories, but they're used to allow access to a single file or to a directory of files. A virtual name can also be created to allow access to schemas or to dbobjects. When creating a virtual name in the virtual name tab of the Configure SQL XML Support in IIS utility, specify the type as either dbobject, schema, or template (if you're going to use the virtual name to access files).

Let's look at the differences between using the URL and using XML files to access your SQL Server 2000 database via HTTP.

To access a SQL Server database via the URL, point your browser to the machine's DNS or IP address (the virtual name), and then append your SQL statement. The SQL is appended as a URL variable called SQL, and by using a "+" to replace spaces in the SQL statement.

It's also important to note that the laws that govern well-formed XML dictate that there needs to be one parent element encapsulating child elements, so you must also tell SQL Server 2000 what name to use for the parent tag that will contain all of the child tags ("elements") that represent rows. This is done by also appending a root=rootname clause to the URL with an ampersand (rootname is whatever name you want to give to base the element).

A side note to this is that those of you who are familiar with XSL and want to create an XSL document to format and/or parse through the XML created by a SQL HTTP request passed to SQL Server 2000 can do so by passing the optional


cfdj_sqlserver2K Properties

General | Security | Data Source | Settings | Virtual Names | Advanced

Options

Specify the type of access to SQL Server you want to provide through this virtual directory.

☐ Allow URL queries

Allows users to execute queries through a URL.
For example, "http://iisserver/iisvdir?SQL=select+*+from+employees+for+xml+auto"

☑ Allow template queries

Allows users to execute queries defined in templates.
For example, "http://iisserver/iisvdir/myquery.xml"

☐ Allow XPath

Allows users to execute XPath queries over SQL Views provided by mapping schemas.
For example, "http://iisserver/iisvdir/cust/CustomerList"

☐ Allow POST

Maximum size of POST queries (in kilobytes):
100

OK | Cancel | Apply

FIGURE 3: Virtual Directory Properties

"XSL=xslfilename" attribute to the URL (where "xslfilename" equals the name of the XSL document they want to load). If I wanted to retrieve the same data result as before (an AUTO mode XML packet of all the rows in my States table) with a base element of CFDJStates, I would browse to http://127.0.0.1/cfdj_sqlserver2K?SQL=SELECT+*+FROM+STATES+FOR+XML+AUTO&root=CFDJStates.

The resulting page in Internet Explorer is shown in Figure 4.

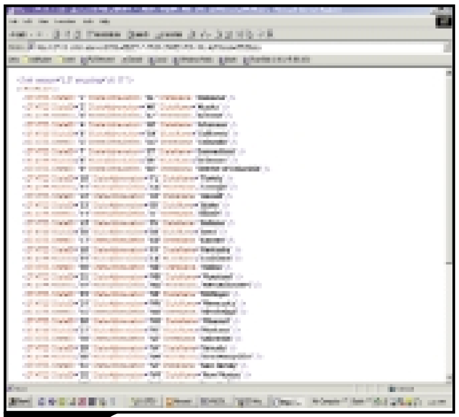


FIGURE 4: Auto mode XML packet

There are two obvious problems with this method of data retrieval.

The first is that it's highly insecure, as the SQL statement is in plain sight of anyone who browses it, and can thus be manipulated by end users. This is a problem if you don't want your database schema to be known, want to restrict what data can and can't be retrieved, or would like to reduce temptation to users to call system SPs, delete data (or drop tables), or do any of the other nasty possibilities when you have direct access to pass SQL statements to a remote database.

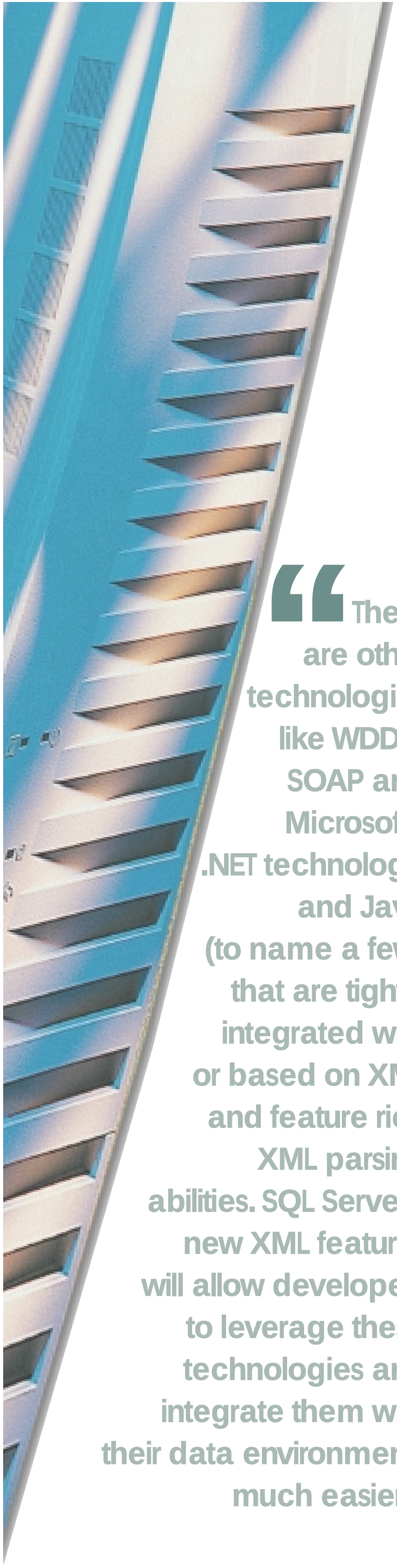
The other reason you might not want to use this approach (as if security isn't enough) is the simple fact that you would probably like to either manipulate the data or present the data in another format to your end users.

The best workaround to this is to request the URL via the CFHTTP tag. If you request a URL with SQL appended to it via CFHTTP, specify "GET" for the value of the METHOD attribute in most scenarios. Requesting the XML packet in the URL as part of a CFHTTP request not only prevents exposing the SQL to users, but places it into a variable with a valid name so that we can work with it in our CFML templates by accessing #CFHTTP.FileContent#.

The second method of HTTP retrieval of data as XML is via file. You can create an XML file that has SQL

MACROMEDIA

www.macromedia.com/go/devcon01



"There are other technologies like WDDX, SOAP and Microsoft's .NET technology, and Java (to name a few) that are tightly integrated with or based on XML and feature rich XML parsing abilities. SQL Server's new XML features will allow developers to leverage these technologies and integrate them with their data environments much easier"

embedded inside it, create a virtual name that points to that file or its directory, and then request that virtual name (append the filename as if it were any other file embedded in any Web root subdirectory if your virtual name points to a directory) in order to execute your SQL. An XML file that executes the same SQL shown in previous examples would look like this:

```
<ROOT xmlns:sql="urn:schemas-
microsoft-com:xml-sql">
    <sql:query>
      SELECT  *
      FROM    States
      FOR XML AUTO
    </sql:query>
</ROOT>
```

If I were to save this text as a file called "getstatesdata.xml" in the same directory to which I pointed a virtual name called "xml_files", pointing my browser at http://127.0.0.1/cfdjsqlserver2K/xml-files/ getstatesdata.xml would yield the same results I received in the last HTTP request made on the URL directly, but the SQL details have now been obfuscated by simply requesting a file that contains the SQL I want to execute.

OPENXML

The last of the major new enhancements to the SQL Server product line with the release of SQL Server 2000 is the OPENXML command, which, in my opinion, is the most significant of the new features.

OPENXML is a command used within stored procedures to read in an XML packet and cache it internally, then operate on the XML data. Before calling OPENXML in a stored procedure, you must call the system stored procedure sp_xml_preparedocument to internally cache the XML packet before being used by OPENXML, and call sp_xml_removedocument after the call to release the internal cache of the XML packet. The OPENXML function accepts three attributes (see Listing 1).

In Listing 1, "@statedata" is the variable used to hold the XML packet, and @hDoc is the index variable used to identify the XML packet. Note that the variable is declared internally, passed to sp_xml_preparedocument, passed to the OPENXML statement, and finally passed to the sp_removedocument stored procedure.

In order to efficiently and securely copy all 51 rows from the "States" table to the "StatesArchive" table, a few steps should be performed.

First create a stored procedure to receive, index, and then insert an XML packet into another table. Once that stored procedure is created, create an XML file that calls the stored procedure in a virtual name directory. Last, create a CFML template that retrieves all of the data from the States table as an XML packet, and removes the first line of text ("<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">" is not part of a valid packet expected by the OPENXML function, as it's neither a parent nor a child element, but an XML declaration), and, last, passes the XML packet to the stored procedure we just created.

Let's assume the T-SQL example for OPENXML from above is saved as a stored procedure called *copystates*. The only thing left to do to retrieve the data in the States table and implement the OPENXML stored procedure is create and execute the template shown in Listing 2.

The average time to retrieve all of the data in the States table as XML, pass it to a stored procedure, and have the stored procedure insert all of the data from the XML packet into the StatesArchive table (51 rows – about 5K of data) was 10 milliseconds.

This is lightning fast compared to results using any other method – primarily because of SQL Server's internal caching of the XML, and the fact that my database traffic to pass the data to be inserted to the database is cut down to a single call. The benefits of using OPENXML for archiving are impressive. Coupled with the new HTTP accessibility features, ColdFusion developers also have some powerful new tools for B2B data syndication.

Where Do We Go from Here?

This article only touches on the new features available in SQL Server 2000. If your company or client has already made the investment in SQL Server 2000, I hope this article has served as enough of a primer to get you started.

If you're not yet working with SQL Server 2000, you should now feel fairly comfortable in any situation where you might be asked whether or not upgrading or investing in a SQL Server 2000 database server offers any benefits to your company or client. Or you may even be asked how a project plan or application code base might be approached differently were SQL Server 2000 used as the database platform instead of some other database.

The potential uses for SQL Server 2000's new XML-based features are limited only by your application needs and your imagination. For example, XSL offers developers a way to format the layout and appearance of XML in a browser, and has its own simple, conditional logic and looping constructs. This could be leveraged along with the features discussed in this article to run queries, loop over that query result set, and manipulate the display for users – all without burdening your CFAS with any CFOUTPUT, CFQUERY, or CFIF calls.

Letting the client's processor execute loops of this sort can make a significant impact on site performance as well as enhance user experience. There are other technologies like WDDX, SOAP, and Microsoft's .NET technology, and Java (to name a few) that are tightly integrated with or based on XML and feature rich XML parsing abilities. SQL Server's new XML features will allow developers to leverage these technologies and integrate them with their data environments much easier.

If you want to learn more about the new XML features in SQL Server 2000, I recommend reading the recently released *Professional SQL Server 2000 XML* by Paul J. Burke, et al (Wrox Press). It covers these topics and many, many more in much greater detail and is an excellent resource for those wishing to delve into SQL Server's new XML features.

About the Author

Simon Horwith is a senior developer and Macromedia Certified ColdFusion instructor at Fig Leaf Software in Washington, DC. He has built applications with ColdFusion since version 1.5. Previously, he developed applications with Visual Basic and AS/400 Relational Database RPG. Horwith is also a contributing author of Professional ColdFusion 5.0 *(Wrox Press), and recently launched www.how2cf.com, a ColdFusion how-to site (hosted by CFDynamics).*

shorwith@figleaf.com

Listing 1

```
CREATE PROCEDURE copystates
@statedata varchar(8000)
 AS
DECLARE @hDoc int
exec sp_xml_preparedocument @hDoc OUTPUT, @statedata
INSERT INTO StatesArchive
SELECT *
FROM OPENXML(@hDoc,'/ALLSTATES/STATES')
WITH testtable
EXEC sp_xml_removedocument @hDoc
GO
```

Listing 2

```
<CFHTTP METHOD="GET" URL="http://127.0.0.1/cfdj_sqlserv-
er2K?SQL=SELECT+*+FROM+STATES+FOR+XML+AUTO&ROOT=ALL-
STATES">
<CFSET temp = ListDeleteAt(CFHTTP.FileContent, 1, ">")>

<CFQUERY DATASOURCE="CFDJ" NAME="copystates">
{ call copystates ('#Trim(temp)#') }
</CFQUERY>
```

CODE LISTING

The code listing for this article is also located at

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

HOSTMYSITE.COM

www.hostmysite.com

Probe Your Servers

Correct problems when they occur

BY
BEN
FORTA

For ColdFusion to work properly lots of bits and pieces have to be in place and functioning correctly. Web servers need to be up and running, database servers need to be accessible, any external components need to be reachable, and, of course, ColdFusion must be functioning properly.

If any of these fail, so will your applications; if failure does occur, you need to know as quickly as possible, preferably before end users find out. Which brings us to the subject of probes.

Understanding Probes

There are lots of ways to check that code or services are working, but the simplest is to just give it a try – if an error is generated, what you're testing is broken. For example, to test that a database is accessible you could run a simple query against it using ColdFusion <CFQUERY>. If <CFQUERY> returns an error, you'll know that the database was not accessible – simple as that.

Now imagine you had a way to automatically execute that simple query at regular intervals, perhaps every few minutes. Also imagine you had a way to examine any output generated by the query and could take some action based on what it contained. Well, don't imagine. That's exactly what a probe is, and ColdFusion has them built right into the product.

The basic principle behind probes is very simple. You tell ColdFusion what to execute and how often, how to recognize a failure condition, and what to do when one occurs – ColdFusion does the rest. You can create as many probes as you need, and you can edit and remove probes too, all using the ColdFusion Administrator.

Note: Probes are only supported in ColdFusion Enterprise, not in ColdFusion Professional.

Standard Probes

Before we discuss writing your own probes, it's worth mentioning that ColdFusion has several standard probes built right into the product, and you can use these without defining or creating anything. To access these, open the ColdFusion Administrator, select the Tools tab, and select Alarms from the Server Monitoring options. You'll see a screen like the one shown in Figure 1, which lets you specify the e-mail address (or multiple addresses separated by commas) to which alarms should be sent if any of the following events occur:

- Web server is inaccessible
- Web server is busy
- Load balancing server (if one is being used) is inaccessible

You may also specify which e-mail address to use for notification if any other probes fail, as well as the SMTP server to use to send the mail.

Creating Probes

The real fun (okay, so fun may be a bit strong, but bear with me) is in creating your own probes. To do this go to the Server Probes page in the ColdFusion Administrator by selecting the Tools tab, and then System Probes from the System Monitoring options. As seen in Figure 2 this page displays a list of probes along with their status (green if okay, red if failed, yellow if temporarily suspended), along with buttons that allow you to create, edit, or delete probes.

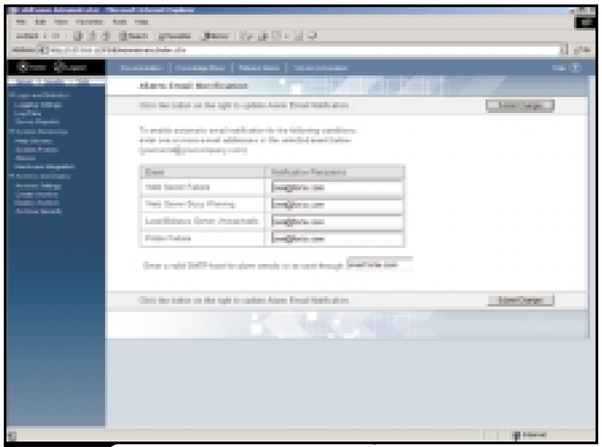
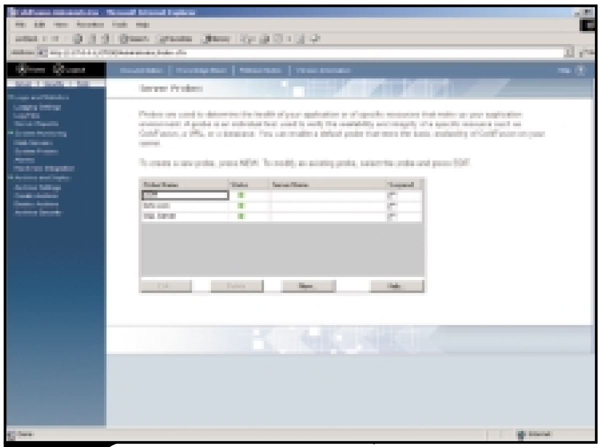


FIGURE 1: Alarm notifications may be sent to e-mail addresses of your choice



Server Probes

FIGURE 2: The Server Probes page displays all probes and their statuses

To create a probe click the New button to display the Server Probe Setup screen seen in Figure 3. You'll be prompted for two settings: Probe Settings, which define the probe itself, and Failed Probe Response Settings, which specify the actions to be taken upon probe failure.

First you must define the probe type and you have two choices: Default Probe executes a URL (usually a CFM page, but it doesn't need to be so) that allows you to inspect the generated results; Custom Probe executes an application (an EXE file, for example) that allows you to take action based on return values. You'll also need to name the probe using any unique name, and specify either the URL (if creating a Default Probe) or application name (if creating a Custom Probe to be executed).

Next comes the important part – you'll also need to tell ColdFusion what to look for to determine success or failure, and this too differs based on the probe type. If you're creating a Default Probe specify a string of text to search for (the match is case-sensitive); if you're creating a Custom Probe specify the return value that's expected upon success. Either way, each time the probe has been executed ColdFusion will be able to determine success or failure by inspecting the results.

You may also specify a frequency (how often the probe should be executed, default is every 60 seconds) and a timeout (when should ColdFusion give up and assume the probe is failing, default is also 60 seconds). Be careful not to specify too low a frequency; you wouldn't want all those requests being executed unless you absolutely need that level of monitoring.

Having defined the probe you'll then need to tell ColdFusion what to do when a failure occurs. You have several options here:

- Specify a server against which actions should be applied
- Restrict traffic (preventing further access) to a server
- Execute a script or file of your choice
- Restart ColdFusion (this option is only available if creating a Default Probe)
- Log that ColdFusion was restarted (this option is available only if creating a Default Probe)

And of course, if you specified an e-mail address in the alarms page (see Figure 1), an e-mail will be sent as well.

What to Probe

Now you know how to create probes. But what should you probe? Here are some suggestions:

- You should have one global probe to check that your site is up. Have it request your home page, or any other page of your choice, to check that the site is up and responding.
- Monitor your databases. Either retrieve no rows (write a query that returns nothing) or execute a stored procedure – the data returned is not important, you just need to make sure no errors are returned. To do this, wrap the query in a <CFTRY> block and output either SUCCESS or FAILED (and set the probe to check for SUCCESS).
- If you have a production site and aren't expecting code changes, monitor the file and directories (traverse the tree finding date and time stamps and compare them to known values). You don't want to execute code like this too often, but running it periodically will let you know that someone has been up to no good. You could even specify an action that would restore the server to a prior known state (self-unhacking, now that's a cool idea!).
- If your DBMS or any other system has diagnostic utilities, set up Custom Probes for them to ensure it's up and running.
- You could even create a special probe page that performs all sorts of tests and sets results as needed. This way a single defined probe could test for multiple potential problems, and you'd be able to update the list easily without even having to access the ColdFusion Administrator.

Give it some thought, you'll have no problem coming up with a list of items or services to probe.

“...if failure does occur, you need to know as quickly as possible, preferably before end users find out”

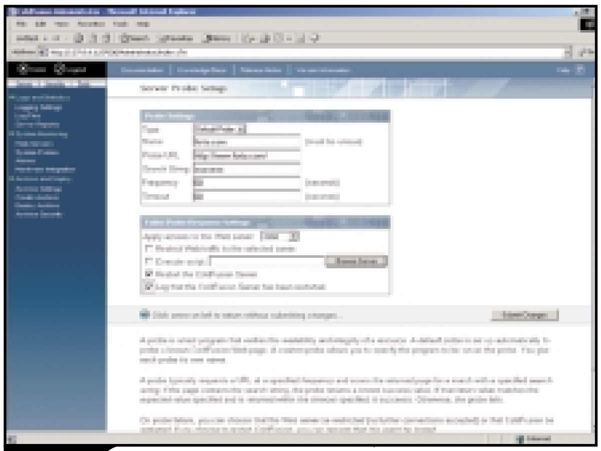


FIGURE 3: The Server Probe Setup Page is used to define system probes

Summary

Exactly what you probe is up to you, as well as where you probe it from (using Default Probes you can probe remote servers just as easily as you could local servers), and what you do when a failure occurs. Probes are not just the 24/7 eyes and ears monitoring the health of your servers, they're also the toolkit you can utilize to correct problems when they occur. If you're not using probes yet, you should...and quickly.

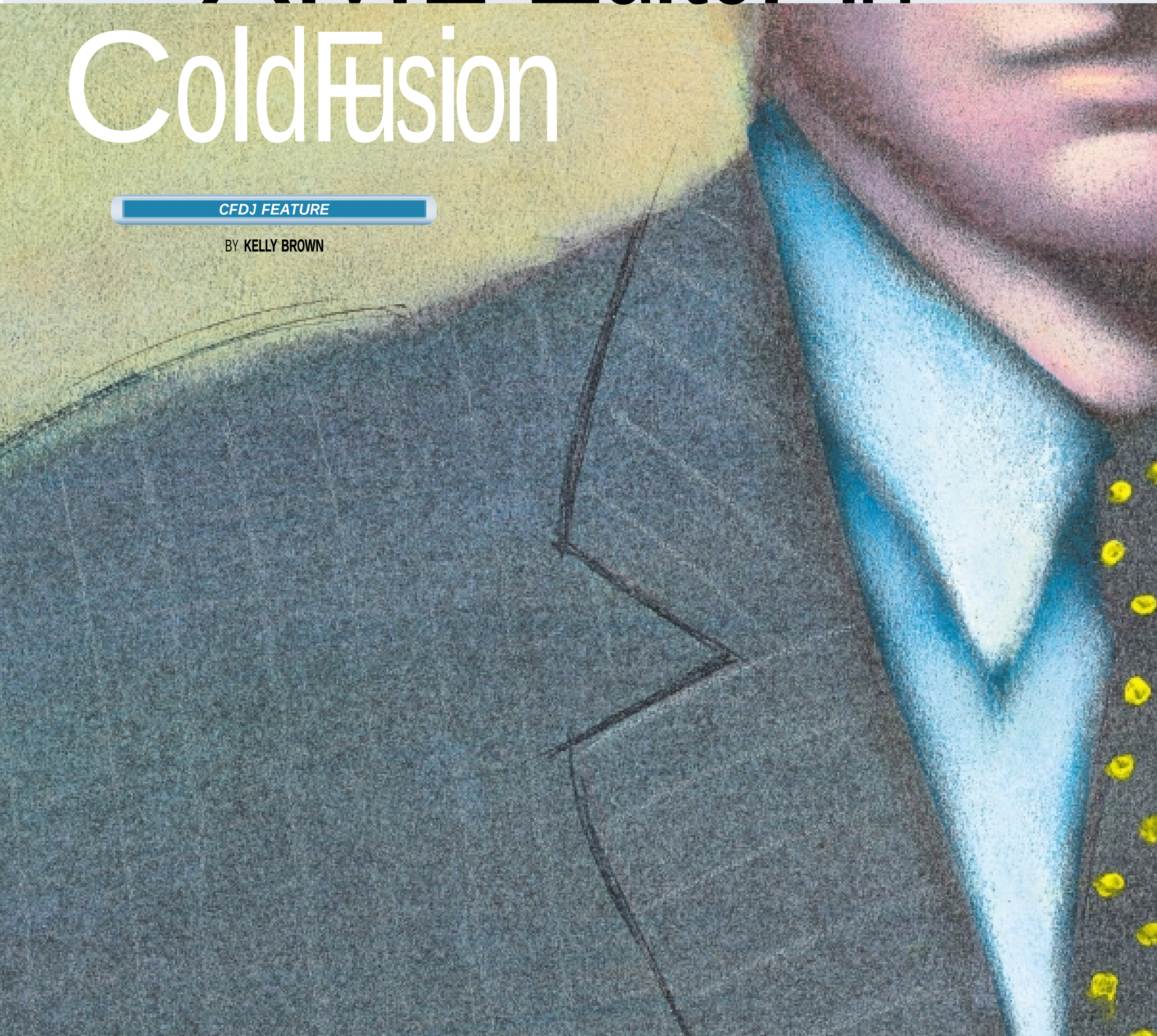
ABOUT THE AUTHOR
Ben Forta is Macromedia's senior product evangelist and the author of numerous books including the recently published ColdFusion 5 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 5 Development. *For more information on Ben's books visit www.forta.com.*

BEN@FORTA.COM

Creating a Custom XML Editor in ColdFusion

CFDJ FEATURE

BY KELLY BROWN



It's almost impossible to pick up a trade magazine these days and not find a reference to XML. It's an important enabling technology for the sophisticated Web applications of the future.

This article will cut through some of the hype to explain what XML can be used for and, equally important, how it can work effectively with ColdFusion. To illustrate how these two technologies can be used in conjunction, I'll create a simple ColdFusion application allowing a user to edit the content of an XML file.

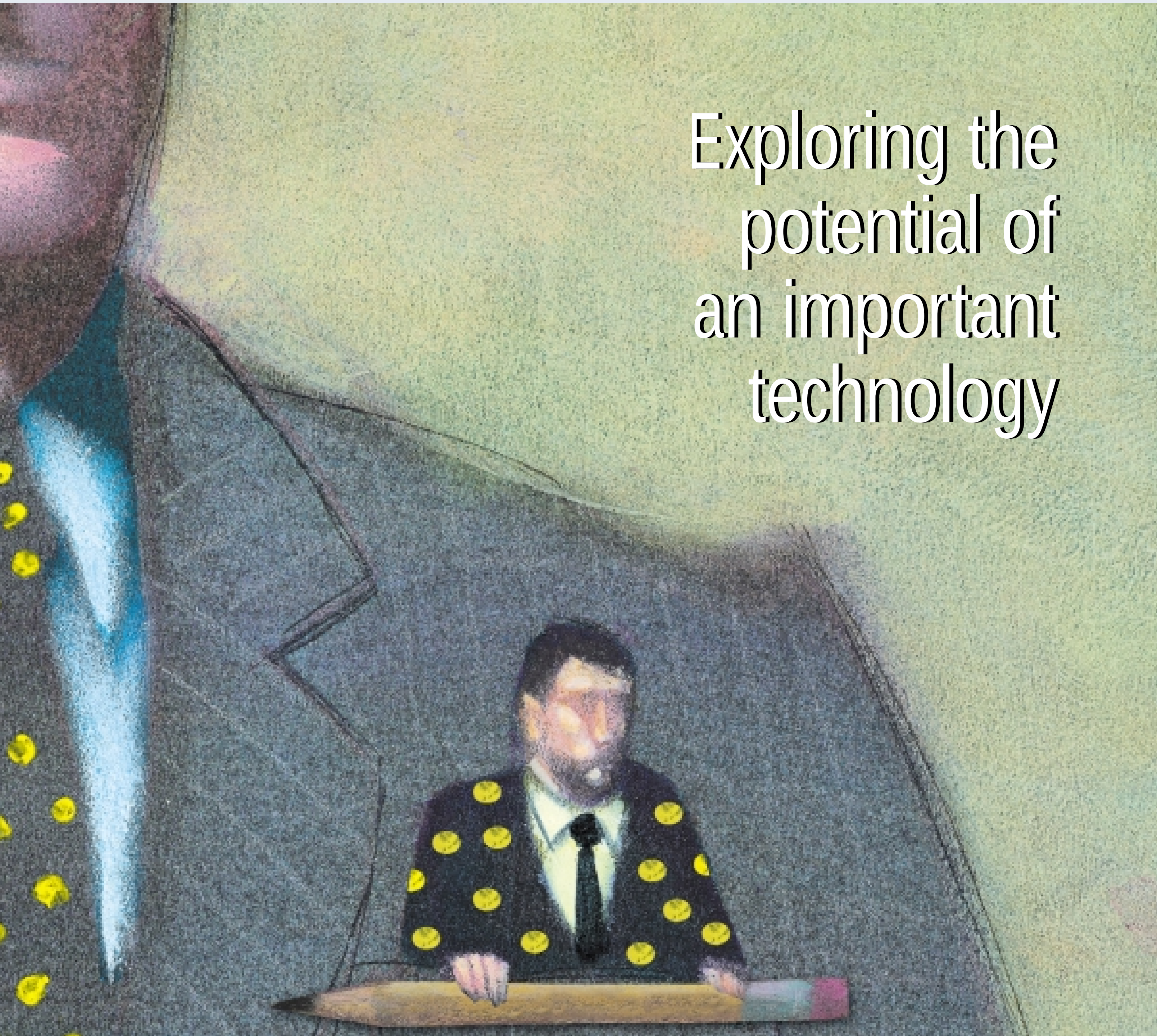
XML 101

For those of you not familiar with XML I'll introduce some of the basic concepts. XML stands for eXtensible Markup Language. It's similar to HTML, but instead of describing *how* to display the information, it describes *what* the information is. For instance in HTML you use the <B> tag to tell the browser to display a name in a bold font. In XML you'd put the <person> tag around the name to indicate that the text is a person's name. So you've told the computer what the name is, but how do you get it to display properly? That's where eXtensible Stylesheet Language comes into play. XSL allows you to define how tags in an XML file should be displayed. For example, we could use XSL to display the text in the <person> tag mentioned previously using a bold font.

XML allows us to separate the structure of the content from the manner in which the content is displayed. This makes XML content considerably more reusable than HTML content, where content and display logic are inextricably mixed.

Another big difference between XML and HTML is that in XML you get to define what your tags are. I used the example of a person tag. There's no predefined person tag in XML. There was a person's name in my data so I just decided I would create a tag to identify it.

You mean you can just create any tag you want? Won't it be confusing if you just go along creating new tags all the time? Absolutely. To create organized XML, you should first create a document type definition (DTD). The DTD defines what XML tags I'm going to use and how I'm going to use them. It sets the rules I'll use when I create my XML documents.



Exploring the potential of an important technology

Each type of data you work with will have its own DTD with tags that are relevant for that data. There are several companies trying to standardize DTDs for certain purposes, but right now things are still up in the air. It's really a matter of agreeing on a DTD with whomever you're going to be sharing data. If you're creating the data you may just determine the DTD and provide it to partners so that they can use your XML content. Several companies in an industry may get together and decide they're all going to use the same DTD, thereby creating a standard for that industry.

This illustrates two more advantages of XML. First, you define whatever tags are a natural fit to your data. Second, a DTD provides an effective way of communicating the structure of your content to any other person or organization that wants to use it.

On to the Application

For purposes of this article, we'll be editing books. Each book will have a title, the number of pages, a price, an ISBN number, and an author. The book can also belong to several categories. Listing 1 shows the DTD for our book.

The DTD is created using tags with "less than" and "greater than" that look similar to tags in HTML. The first line in our DTD tells us what version of XML we're using, which is 1.0. Next we start defining the elements. An element will be a tag in our XML document. We create a definition for each of our elements. The definition tells us what the name of the element is, and what kind of data will be in the element.

The first element we define is the title, which means there will be a <title> tag in our XML document. We also define what kind of data will be in the title tag. In this case it will be PCDATA, which stands for parsed character data. This basically means that the tag will have some text in it. We then define our page count, price, ISBN, and category as PCDATA.

The author tag is a bit different. We break the author name down to the first name and last name. In the DTD we define the author tag as a list of other tags, using the parentheses to select the elements that the author tag will consist of. Next we define the firstname and lastname elements that will be used for the author's name.

The last element is the book tag. The book tag contains all the other tags. This creates a hierarchy of tags within tags. So there will be a top level <book> tag with all the other tags in it. There are a couple things to notice about the book element. The firstname and lastname elements we defined aren't in the book.

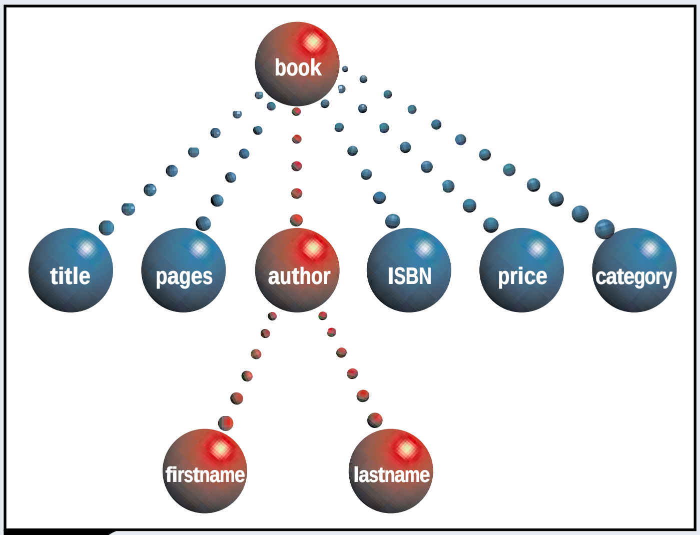


FIGURE 1: Sample XML tree

The book includes the author element, which includes these tags. The other thing is that the category element has an asterisk after it.

The asterisk indicates that the element can occur zero or more times. The other elements didn't have a notation after them, so they can occur only once in the XML document. There are also some other indicators you can use. A plus sign after the element indicates the element will occur one or more times in the XML, and a question mark indicates that the element will be used zero or one times. If you're familiar with regular expressions, this is the same syntax used for pattern matching.

Listing 2 shows a sample XML file that conforms to our DTD. The first line once again specifies what version of XML we're using. The second line tells us what DTD is used for this document. In this case we're using the book.dtd file that we defined above. We have our top-level book tag followed by the other tags we defined. The author tag contains the firstname and lastname tags that we defined.

XMLDOM Object

Now we have an XML file, but how do we access it from ColdFusion? Fortunately, Microsoft has provided us with an XML parser COM object that is both powerful and easy to use. The object is called XMLDOM. You can download this component from the Microsoft site for free. Windows 2000 ships with this COM object, but it's an older version and I suggest you download and use the latest version. You can find this component on the Microsoft site by searching for the Microsoft XML SDK.

Once you download and install the component you can instantiate it in ColdFusion with the following syntax:

```
<cfobject type="COM"
          name="objXMLDOM"
          class="Microsoft.XMLDOM"
          action="CREATE">
```

Once the object is created we can use the LOAD method to retrieve an XML file. Before we do this we want to set the async property. The async property stands for asynchronous and tells the component to wait until the XML file is loaded to continue. This is an important property to set. The default is true, which means your code will continue before the file is loaded and you won't be able to access any of the data. Note that you must provide the full path along with the file name as shown below:

```
<cfset objXMLDOM.async = false>
<CFSET objXMLDOM.LOAD("c:\samplexml\book.xml")>
```

After we make changes to the XML we can save the changes using the SAVE method. The syntax is the same as the LOAD and takes the full path for the XML file to be saved.

Finding Information Using XPath

Once we load an XML document we have several methods to access and manipulate the data. The structure of the data in the COM object is referenced as a tree structure of nodes. Figure 1 shows a visual representation of our XML book structure with each element as a node. The book is the top node with the title, pages, price, ISBN, author, and category as child nodes. The author node has two children nodes of firstname and lastname.

We can find data in our XML file using the SelectSingleNode method. This method uses the XPath language to find a node. The XPath language is very powerful, full-featured, searching language. We will cover only some of the basic searching syntax.

MACROMEDIA

www.macromedia.com/downloads

“XML is an important technology for the future. In a medium, such as the Internet, where content is king, XML is an enabling technology that will allow organizations to maximize the potential of their content”

The XPath syntax is similar to a directory path in the file system. The top node is like the top directory that contains the other nodes as directories. The following syntax would select the title node for our book:

```
SelectSingleNode("/book/title")
```

Once we have the book node we access the TEXT property to get the data contained in the tag. If we wanted to print out the title in ColdFusion it would look like this:

```
<CFSET title_node=
SelectSingleNode("/book/title")>
<CFOUTPUT>#title_node.text#</CFOUT-
PUT>
```

The text in the node can be changed by assigning data to the TEXT property of the node.

```
<CFSET title_node.text="My New Book">
```

This does not automatically save the changes back to the XML. It simply changes the data in memory. You have to use the SAVE method to update the file.

More Searching Techniques

There are other search mechanisms available to us in XPath. A double forward slash will search the entire XML structure for the specified node. The example below would return the firstname node even though it’s a subnode of author.

```
SelectSingleNode("//firstname")
```

Be careful when using this searching method. If I had another firstname tag, like the name of the editor for instance, I would not know whose first name I was accessing.

In our DTD we defined category with the ability to occur multiple times within our document. We need to use the SelectNodes method if we want to get more than the first occurrence of a node. The SelectNodes method returns a collection of nodes that meets the specified criteria. Here’s how we’d get a list of the categories defined for our book:

```
SelectNodes("/book/category")
```

XPath also provides us with the capability to search for nodes that contain specific data. We can include additional search information in square brackets as shown:

```
SelectSingleNode("/book/category[. =
""mycategory""]")
```

This would return a category node where the text is “mycategory”. We’re using the “ ” in this example to get quotes within our quotes.

Adding and Deleting Nodes

We can create new nodes for our XML using the CREATENODE method. This method takes three parameters. The first is the node type. Type one is an XML element. The second parameter is the name of the node, which in this case is category. The last parameter is the URI name space, which we aren’t using so we’ll leave it blank.

```
<CFSET new_cat =
objXMLDOM.createNode(1,"category","")>
```

Once we have a node we can add it to an existing node using the APPENDCHILD method. In the following example we get the book node and then append our new category node to it.

```
<CFSET book_node =
objXMLDOM.SelectSingleNode("/book")>
<CFSET
book_node.appendChild(new_cat)>
```

When opening an XML file that you’ve added a new node to, you may notice some strange formatting. The new tag is added onto the end of the existing tag without any spacing. New nodes are actually separated by newline characters as specified in the XML standard, but in a windows environment most editors won’t show the newline character so everything will appear to be on one line.

The delete is similar to the edit. We find the node we want to delete and the node it belongs to in the hierarchy. This time we use the removeChild method to remove the category node from the book node.

```
<CFSET cat_node =
objXMLDOM.SelectSingleNode("/book/cat
egory")>
<CFSET book_node =
objXMLDOM.SelectSingleNode("/book")>
<CFSET
book_node.removeChild(cat_node)>
```

Code Walk-Through

We have the basic tools we need, so let’s take a look at a sample application. Our sample consists of six ColdFusion pages. That’s a lot of pages, but they are fairly simple and each one presents a different principle. The first page is called list.cfm. This page shows a list of XML files we can edit and allows us to select one to edit.

The xmledit.cfm page displays most of the book fields in a form and allows us to change them. This demonstrates searching and displaying information from an XML document.

The next page is the xmledit_action.cfm page. This page handles the form submission from xmledit.cfm, and it demonstrates how to update nodes in an XML file. The category is handled differently from the other fields because it can occur multiple times within the XML.

We have an addcat.cfm page that allows you to add a new node to the XML file. The editcat.cfm allows you to change an existing node and demonstrates some of the advanced searching capabilities of Xpath. The final page is delcat.cfm, which allows you to delete a category by removing a node from the XML file.

• *list.cfm*

This page is pretty straightforward. It uses cfdirectory to get a list of all the XML pages in the specified directory. It displays these files as links to xmledit.cfm with the file name passed in on the URL. Passing file paths over the URL is generally a poor security practice, but we’re keeping things simple for this example. We also display the size of the XML file for good measure (see Listing 3). (Listings 3–8 can be found on the ***CFDJ*** Web site, www.sys-con.com/coldfusion/sourcec.cfm.)

• ***xmledit.cfm***

In this page we load our XML file. We create a form to edit our data. We put the file name in a hidden field to pass onto the action page. We use the select single node to get most of our data. You can see several different searching methods working in this page. Our category search can return more than one node, so we used cfloop to loop over the collection that is returned. For each category we generate a link to the catedit.cfm and catdel.cfm pages. We also provide a link to add a new category (see Listing 4).

• ***xmledit_action.cfm***

This page gets the form data from the xmledit.cfm page and updates our XML document. We load the document like we did in the xmledit.cfm page. We find the nodes using the same search as before, but this time instead of outputting the node text in our form we set the node data to the form data. Once the data has been changed we use the SAVE method to write our changes back to the file, as seen in Listing 5.

• ***addcat.cfm***

There isn't a separate action page for the category manipulation pages. The same page is used to generate the form and process it by checking to see if there's data to be processed. For the add form we pass along the document in the hidden field and give the user a single text box to enter the name of a category. The form is processed in a similar manner to the other pages we've seen. First create the XML and load the XML file. In this case, create a new node with the CREATENODE method. Once we have the node we assign our data to it using the text property as we did in the xmledit_action.cfm page. Now we have our new node, but we need to add it into our structure. We want to add it to our main book node so we get the book node and use the APPENDCHILD method to add it to the book node. The changes are complete, so we save them back to the file and send the user back to the xmledit.cfm page where the new category will now show up (see Listing 6).

• ***editcat.cfm***

For the edit category we are changing an existing node, so we need to save the file name and the original node data in hidden fields. We provide the user with a text box to enter the new category text for the node. On the processing side we load the data as usual. This time, however, we have to find the correct node to change. We're going to use the square brackets in our XPath search to find the correct node. In the code we use the original category name we passed in to find the category we want to edit. We replace the text of the node with the new category text and save the file. The user is then sent back to the xmledit.cfm page (see Listing 7).

• ***delcat.cfm***

For the delete category we also pass in the name of the category to be deleted. We place the document and category name in hidden fields and ask the user to verify the delete. If the user verifies the delete, we process the form; if not, we redirect them to the xmledit.cfm without making any changes. By now you know the drill, we load up the XML file. The delete is similar to the edit. We find the node to delete and the book node. This time we use the removeChild method to remove the category node from the book node. We save our data and return to the xmledit.cfm page (see Listing 8).

Some Things to Add

One thing not dealt with in the sample application is adding and deleting the XML files. Since we're dealing with files, you can use the CFFILE tag to delete any files you don't want. To create a new file you could create all the nodes and append them together, but there's an easier way. You can create an XML template file with the correct structure, but with all the nodes blank. When you want to create a new XML file you can make a copy of this file and then send it to the editor. This saves us the work of creating a new add XML file by reusing our edit.

Conclusion

XML is an important technology for the future. In a medium such as the Internet where content is king, XML is an enabling technology that will allow organizations to maximize the potential of their content. ColdFusion can be used to effectively manipulate XML and to incorporate XML into sophisticated Web applications. The ColdFusion application presented in this article has only scratched the surface of XML's potential.

Start with these basic tools and concepts, and keep learning. Welcome to the world of XML!

About the Author

Kelly Brown is the CTO of AboutWeb (www.aboutweb.com), an Internet solutions provider in the Washington, DC, area. He has a BS and MS in computer science and is a Microsoft-certified systems engineer.

kbrown@aboutweb.com

Listing 1

```
<?xml version="1.0"?>


<!ELEMENT title (#PCDATA)>
<!ELEMENT pages (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT author (firstname,lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>



<!ELEMENT book (title, pages, price,
   ISBN, author, category*)>
```

Listing 2

```
<?xml version="1.0"?>
<!DOCTYPE book SYSTEM "book.dtd">
<book>
 <title>Computer Book</title>
 <pages>100</pages>
 <price>19.95</price>
 <ISBN>123456789</ISBN>
 <author>
  <firstname>Kelly</firstname>
  <lastname>Brown</lastname>
 </author>
 <category>Computer</category>
 <category>Education</category>
</book>
```

CODE LISTING

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

2001

JDJEDGE conference & expo

With

Cf ColdFusion fasttrack

September 23-26, 2001 is your most valuable educational opportunity of the year. Don't Miss It!

Register NOW for best rates!

The Depth and Breadth of Education to Advance Your Professional Skills

The JDJEdge 2001 Conference & Expo provides your best opportunity to understand how Java technologies can solve enterprise challenges.

Four information-packed JDJEdge Tracks featuring leaders in Java Technologies

Track 1 J2ME – Micro Edition & Wireless
Cutting-edge sessions for software engineers and hardware specialists working on wireless solutions.

Track 2 J2SE – Standard Edition
General Java programming for corporate programmers developing full Java applications, including several introductory sessions.

Track 3 J2EE – Enterprise Edition
Advanced sessions for software architects, Web programmers, corporate developers and consultants developing server-based applications.

Track 4 Working with *I*-Technology
Technical and management sessions for business analysts, corporate systems managers, architects, project managers and CIOs .

JAVA DEVELOPER'S JOURNAL JDJ READERS' CHOICE AWARD

JDJ Readers' Choice Awards, the Oscars of the Software Industry, is the world's most widely participated industry award program. Winners will be recognized at JDJEdge 2001 International Java Developer Conference & Expo in New York.

Who Should Attend

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

The only event backed by SYS-CON Media and *Java™ Developer's Journal*

JDJEDGE 2001 Features

- Unmatched Keynoters and Faculty
- Over 150 Intensive Sessions and Tutorials
- The Largest Java Expo on the East Coast Participation by Invitation Only
- BEA WebLogic™ FastTrack to Certification
- IBM WebSphere™ FastTrack to Certification
- Macromedia ColdFusion™ FastTrack to Certification
- Sun Java™ FastTrack to Certification

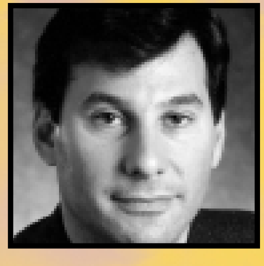
www.sys-con.com/JDJEdge/

JAVA AND JAVA-BASED MARKS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF SUN MICROSYSTEMS, INC. IN THE UNITED STATES AND OTHER COUNTRIES. SYS-CON PUBLICATIONS AND SYS-CON EVENTS ARE INDEPENDENT OF SUN MICROSYSTEMS.

Keynote Panel

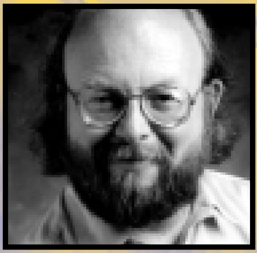
LYNCH
President
Macromedia

BARATZ
CEO, Zaplet, Inc.
Former President, JavaSoft



ROSS
Founder, JavaLobby
Panel Moderator

LITWACK
CEO, SilverStream

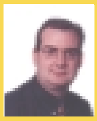
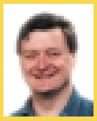
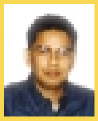
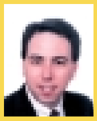
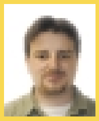


SCOTT
CEO, PointBase, Inc.
Cofounder, Oracle

DIETZEN
CTO
BEA

GOSLING
Creator of Java
VP & Fellow, Sun Microsystems

ADVISORY BOARD

 J. Milbery	 S. Phipps	 A. Sagar
J. Westra	 S. Rhody	 A. Williamson

KIESSLING
CEO
Sitraka

Media Sponsors:
Java Lobby
XML Journal
Java Developer's Journal
WebServices Journal
Wireless Business & Technology
Linux Business Week
WebTechniques
Programmer's Paradise
ColdFusion Developer's Journal
WebSphere Developer's Journal

SYS-CON MEDIA

Owned & produced by
SYS-CON EVENTS
135 Chestnut Ridge Road,
Montvale, NJ 07645
201 802-3069
Fax: 201 782-9651

Using XML Scalable Vector Graphics with CFML

Create dynamic charts and diagrams

BY
CHRISTIAN SCHNEIDER

This article describes how to use Scalable Vector Graphics (SVG), the amazing new graphic format, with CFML to create dynamic charts and diagrams.

SVG is the W3C standard format for scalable graphics based on XML (see www.W3C.org). Yes, that's right, graphics based on XML; for example, see the following code:

```
<svg width="250" height="300">
<circle cx="120" cy="120" r="50"
   style="fill:red; fill-opacity:
     0.7"/>
<circle cx="80" cy="40" r="30"
   style="fill:blue; fill-opacity:
     0.5"/>
</svg>
```

This SVG snippet renders a vector graphic 250x300 pixels holding both a red-filled circle with a radius of 50 pixels at the coordinates 120, 120, and an opacity level of 0.7, and a smaller blue circle. Save this snippet as test.svg and embed it in an HTML page using the following code:

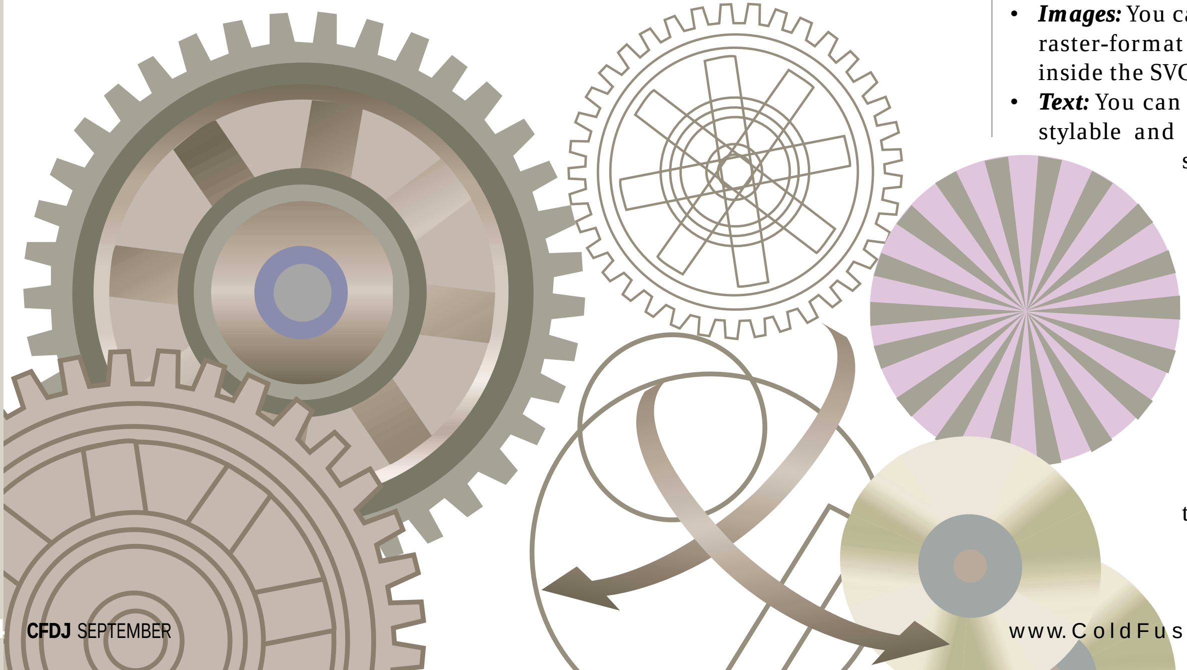
```
<h3>This is my HTML page holding
   the SVG</h3>
<embed src="test.svg"
type="image/svg+xml"
 name="mySvgTest"
 width="250" height ="300"></embed>
```

How will we be able to see this image when opening the HTML page in a browser? Since SVG is a W3C standard, the major browsers will incorporate this format sooner or later into their future releases. Of course, you don't have to wait that long if you install an SVG viewer plug-in that's capable of rendering SVG graphics in your browser now, until this native browser support is available. Some plug-ins for the most common browsers (Internet Explorer and Netscape) are available that comply to W3C's SVG format, including one from Adobe (www.adobe.com/svg).

What Benefits Do We Gain Using SVG?

As SVG is a vector format, it renders on the screen, prints with a high quality and sharpness, and its file size is small compared to other formats such as GIF or JPG. The usage of vectors also makes it "zoomable" on the client without quality loss (just right-click an SVG image, or use CTRL and SHIFT-CTRL to control the zoom with your mouse). The following is a list of some of SVG's coolest features:

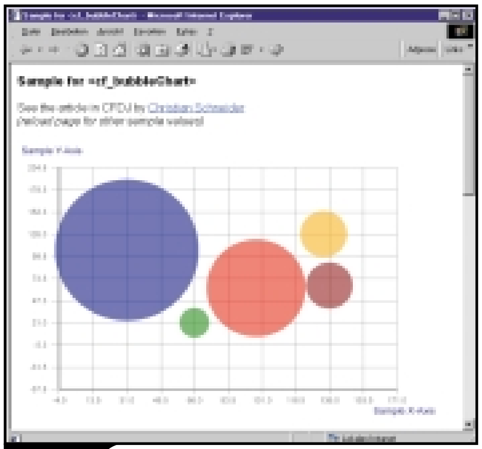
- ***Shapes:*** You can use a lot of basic shapes such as lines, circles, ellipses, rectangles, and polygons, as well as easily define and use your own shapes.
- ***Colors:*** A full color palette that allows you to easily define smooth gradients and transitions in all shapes and directions. You can even use opacity levels to define any grade of transparency for all shapes and texts.
- ***Images:*** You can embed the usual raster-format images (like GIF) inside the SVG if you need to.
- ***Text:*** You can add text that's fully stylable and transformable, but still keeps text that's searchable and easily changeable. There's a full palette of fonts and you can even use your own fonts in an SVG image.
- ***Styling:*** You can use Cascading Style Sheets (CSS) to add lots of styles to your shapes and objects.



MACROMEDIA

www.vue.com/macromedia

- ***Filters:*** You can also define custom filters and apply them as a style. Filters include 3D-effect filters and spotlights.
- ***Animation and interaction:*** All properties of an SVG element – as well as all CSS styles applied to such an element – are easily animated based on a time frame and events (e.g., "mouseOver" or "click"). This can produce flying animations, morphing shapes, scaling and rotating shapes, and hiding/showing objects with just a few simple tags.
- ***Paths:*** You can define any kind of path and bézier curves within an SVG image on which text and other SVG elements can easily be rendered.
- ***Linking:*** All SVG elements can be linked to URLs using the <a> tag within SVG. This even works with different frame targets.



Sample for <cf_bubbleChart>

See the article in CFDJ by Christian Schneider
(reload page for other sample values!)

FIGURE 1: Sample view of the SVG charting component

- ***Scripting:*** All properties and the XML–DOM tree are scriptable using JavaScript or ECMAScript from within the SVG's source or even from within the embedded HTML page's source (using Internet Explorer) and vice versa. This means you can handle an event fired within an SVG image in the embedded page or trigger some changes in the SVG based on an event raised in that page.
- ***XML:*** This is the biggest feature since the whole format is just a plain XML document. No proprietary binary formats, just XML allowing you to dynamically generate the SVG's source on the server-side, making it suitable for on-the-fly database-driven charts, for example. And for those folks still wishing these files were a compressed binary, you can optionally compress SVG files with gzip to get the smallest file sizes.

Developing a Charting Component: Bubble Charts

To provide you with a working example of using SVG with CFML, I've developed a custom tag (<cf_bubbleChart>) that renders a bubble chart, similar to the chart in MS Excel; this kind of chart has a value for the *x*-axis, the *y*-axis, and the radius of each bubble, which enables you to present three data rows in one chart. See Figure 1 for a screenshot of this custom tag in action (note that you must install the SVG viewer plug-in since current browsers don't yet implement SVG natively).

What is the main concept behind this charting custom tag? It's dual-mode since it consists of the custom tag outputting the <embed> tag to embed the SVG data, and a second part that actually creates the SVG data dynamically. This is achieved by the custom tag entering a URL as the <embed> tag's source attribute, which points to a "loader" file for the custom tag. This "loader" file then invokes the custom tag again to generate the dynamic SVG data stream.

When invoking the charting tag <cf_bubbleChart> within a page, the "loader" file places all attributes into the SESSION scope (<cflock>ed of course!), so the invocation of the tag in "SVG generation mode" from the aforementioned "loader" file still knows all parameters to generate the appropriate SVG data. Using the session scope here is quite straightforward since all attributes provided at the invocation of the custom tag are still available when generating the SVG data stream on the server.

I've decided to place the data into the session scope keyed by a random Universally Unique ID (UUID) to allow multiple instances of the charting tag within one page (hence within one session), without the instances overwriting each other. Using a UUID as the key (which is handed over as a URL parameter in the <embed> tag's source) prevents all invocations of the charting tag in one page from being rendered the same while they have different attributes.

A More Detailed Look

Now let's look at this charting component. The working example consists of four files: Application.cfm, index.cfm, bubbleChartLoader.cfm, and bubbleChart.cfm. (All files are available for download from ***CFDJ***'s Web site at www.sys-con.com/coldfusion/sourcec.cfm.) I'll also publish the files to Macromedia's Tag Gallery.

Application.cfm

This file creates a normal session environment for this example using <cfapplication>.

index.cfm

This is our sample file showing how to use the custom tag <cf_bubbleChart>. It sets up a few random bubble values to show in the bubble chart.

bubbleChartLoader.cfm

This file is the aforementioned "loader" file for the custom tag <cf_bubbleChart>, which simply takes a UUID as a URL parameter and calls <cf_bubbleChart> with that UUID.

bubbleChart.cfm

This file is the custom tag <cf_bubbleChart>. When I'm discussing SVG later on I recommend you look at some examples and the specification provided at www.W3C.org and www.adobe.com/svg. This information is important when this custom tag generates the SVG code, since explaining SVG in detail is unfortunately out of this article's scope.

The custom tag first checks if a UUID was passed as an attribute. If so, it loads all attributes from the session scope (where they were stored before) using that UUID, otherwise it expects all attributes directly. After setting the <cfparam>s for the default values of some attributes, the tag decides (by checking if the UUID was provided or not) if it should first create the HTML <embed> tag to embed the SVG chart or the SVG data stream requested from within the <embed> tag. When the first mode is the case, that is, writing the <embed> tag to embed the SVG, <cf_bubbleChart> stores all parameters into the session using a random UUID, which is appended to the URL. This URL of the <embed> tag's source is pointing to the aforementioned "loader" file that calls <cf_bubbleChart>, which loads all attributes from the session scope and generates the SVG data stream.

Generating the SVG data stream is quite easy since SVG is plain XML. After the <?xml version="1.0" standalone="no"?> directive, the DTD must be referenced using the following snippet:

```
 <!DOCTYPE svg PUBLIC "-
//W3C//DTD SVG 20001102//EN"

"http://www.w3.org/TR/2000/CR-
SVG-20001102/DTD/svg-
20001102.dtd">
```

Following this is the <svg> tag, which describes the actual SVG. Since it's plain XML you can simply use <cfoutput>, <cfloop>, and all other CFML tags (even <cfquery>) to make things dynamic. After a few calculations on how to draw and scale the elements and the *x* and *y* axis for the dynamic values, this article's custom tag first draws the chart area, followed by the axis titles and the grid, and finally the bubbles as animated circles. Animation in SVG is easy since it has the genius <animate> tag that allows us to define what properties should become animated (here the radius making the bubbles "fly in" toward the user). From a styling aspect, SVG is also simple to handle since all styles are applied using CSS, which can also be easily manipulated from within CFML.

As I mentioned before, look at the SVG specification and the tutorials (see the aforementioned links) to see what's possible with this open standard from the W3C. Unfortunately, until the major browsers support this standard natively with XHTML, we're forced to use the viewer plug-in, but its installation is simple and smooth so it won't be a problem for most users. And psst..., I've heard rumors that the next release of Acrobat Reader (which many Internet users have, since it allows them to view PDF files) will automatically include the SVG viewer plug-in. So we can even start using SVG before the browsers support it natively.

By the Way

To gain even greater control of displaying alternatives when the viewer plug-in is not available in the client's browser, use the HTML <object> tag to embed the SVG. This lets you specify alternative HTML code to show (in Internet Explorer at least) when the plug-in isn't available. The <embed> tag though works in all browsers! Adobe has prepared a JavaScript for doing browser-independent checking for that plug-in (see www.adobe.com/svg).

There's a lot of work going on around the SVG W3C standard, including the plug-in work from Adobe and others, a Java SDK being developed at Sun, and toolboxes and converters being developed by different companies, including a visual SVG editor (like PaintShop Pro) from Jasc (www.jasc.com). The smartest tool I've seen there is the "Batik" open-source project from Apache (http://xml.apache.org/batik), which includes a Java-based Rasterizer for converting SVG on-the-fly to GIF or JPG images. This can be used for those browsers that don't have the viewer plug-in installed yet, so they receive an image of the SVG instead.

By using the CFML tag <cfobject> to invoke the Java Rasterizer object on the server-side, you can easily serve dynamically generated images to the browser. The coolest thing is that you have to code the graphic only one time for both view types: either serve the SVG stream directly to the browser (having the plug-in available) or use <cfobject> with the Rasterizer to dynamically convert this SVG stream into an image and serve it using <cfcontent>. The HTML <img> tag for doing this would look like <img src="mySvg2-ImageGenerator.cfm?-someparams=somevalues"> where mySvg2-Image-Generator.cfm uses <cfobject> to access the Rasterizer.

ABOUT THE AUTHOR

Christian Schneider is a Macromedia Certified Advanced ColdFusion and Web site developer. He has over four years of intensive experience developing CF-based intranet applications for banks and logistic corporations.

MAIL@CHRISTIAN-SCHNEIDER.DE

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

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

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

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

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

web services
web designer
Stephen Kilmurray stephen@sys-con.com
web designer
Purva Dave purva@sys-con.com
web design intern
Carol Auslander carol@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

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

CFDJ FEATURE

BY **PAUL J. ELISII** AND **SERGE OHOTIN**

Using XSLT to Deliver ColdFusion Applications… Anywhere

While developing an application in Wireless Markup Language (WML) for Wireless Application Protocol (WAP) devices, I realized just how many different types of wireless devices exist globally and how different the protocols and languages have to be from device to device. In a previous article, "ColdFusion in the Palm of Your Hand" (***CFDJ***, Vol. 2, issue 4), I showed how ColdFusion applications can be written and deployed to a wireless Palm VII. In this article I compare and contrast writing that same application in WML for WAP devices.

Taking the Palm VII application and rewriting it in WML for WAP devices wouldn't be overly difficult, but what about all the other devices and protocols such as HDML and XHTML? Rewriting the application for all those various devices and then supporting all the code bases would be quite overwhelming. Ideally, if there was a single markup language that supported all wireless devices, you could just support that single language. However, with the proliferation of more and more protocols and languages it's not very realistic.

There's no questioning the fact that supporting all these wireless devices is important. A recent IDC Report estimates that by 2002 (see Figure 1) there will be more wireless devices capable of browsing the Internet than wired browsers and by 2004 it will be close to double. Worldwide, the ability to support wireless devices will be critical where wireless device penetration will far exceed personal computer penetration.

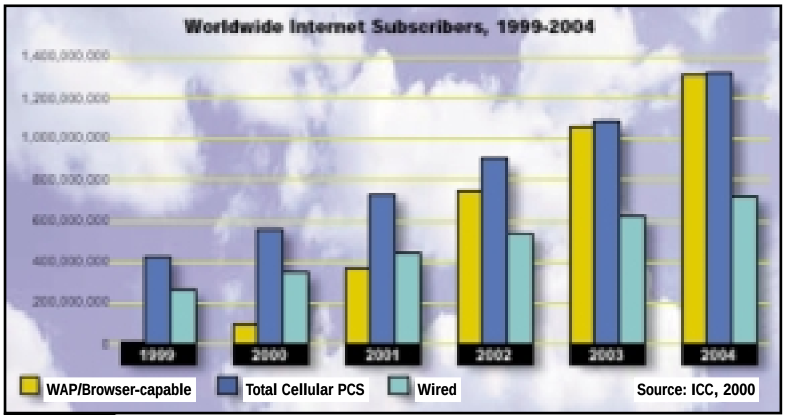


FIGURE 1: Worldwide Internet subscribers

Add support for additional devices
without rewriting the application

With a proliferation of wireless devices accessing the Internet, applications will need to support these devices just as applications must support various browser versions. You wouldn't dream of developing an Internet application that supports only IE or Netscape, so we should be writing applications that support both HTML browsers and wireless devices.

Even though it's not realistic to support a single markup language for all wireless devices, there is a way for you to have a single application that does this. It can be accomplished using XML and XSLT.

This article illustrates how to write a single application that uses XSLT to output its data to HTML for browser access *and* WML for wireless phone access. The generated XML content will be transformed into each of the desired markup languages (see Figure 2).

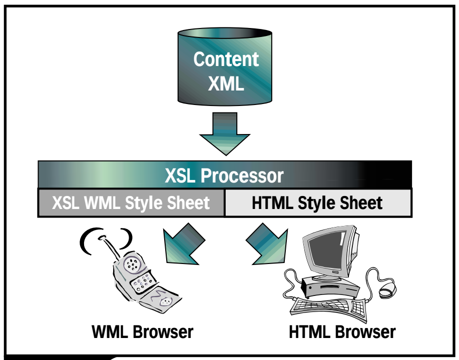


FIGURE 2: XSLT application architecture

The XSLT process takes both an XML and an XSL document as input and, using an XSLT processor, transforms the XML into a different form of XML, HTML, WML, or any other desired output.

The XSL document specifies the rules regarding how the XML is translated. An XSL document is similar to a Cascading Style Sheet (CSS), but much more powerful. XSL is really a language specification in which you can use loops and so forth to translate XML data.

If your application creates HTML as output, like most traditional applications, you can use CSS to take an HTML document and specify how that HTML will be presented. However, if your application creates XML as output, you have many more options. You can apply a CSS to the XML to present the data; you can use XSLT to transform the output to other XML, HTML, WML, etc.; or you can use XSLT to transform the output into XSL-FO (XSL-Formatting Objects), which allows you to present the data in additional formats such as a PDF.

Basics of XSL (XSLT and XPath)

Two of the main components that make up the language elements of XSL include XML Path Language (XPath) and XSLT. Most XSL documents contain elements of XPath and XSLT to transform XML documents.

To actually execute the transformation an XSLT processor is required. This processor takes the XML and XSL and then parses to create the desired output. This process is standardized by the W3C and there are many good XSLT processors available on the Internet such as MSXML, Saxon, and Xalon. A list of XSLT processors can be found at www.wrc.org and can usually be freely downloaded. For the examples in this article we use the Saxon XSLT processor. It's Java-based, easy to use, and a complete implementation of the standard.

Within an XSL document you'll use both XPath and XSLT to translate XML.

XPath allows you to navigate within the XML document tree. For example, child::* will match all the children of a context node that's currently being processed. The XPath language also contains predicates (filters) and functions that can create very powerful capabilities of parsing XSL documents.

XSLT allows you to select and manipulate the data elements that you can use to navigate XPath. It allows you to create templates, specify output, and control the flow of processing in an XSL document. One example of an available function is the count() function, which could be used to parse through an XML document and output all the customer records that have at least two orders, for example:

```
<xsl:template
match="Customer[count(//Order)>1]">
 <xsl:value-of select="Product"/>
</xsl:template>
```

There are entire books dedicated to XPath and XSLT. Two that were useful in the writing of this article were the *XSLT Programmer's Reference* by Michael Kay (author of the Saxon parser) and *Professional XSL* by Kurt Cagle, et al. Both books are published by Wrox Press Ltd.

"Support for wireless mobile devices will increase rapidly in the next two years, and the demand for applications that support these devices will increase accordingly"

Sample Application – Contact Manager Lookup

In this article I review a contact management application that queries a database for a list of contacts that match the desired query request.

The two main controlling ColdFusion files in the application consist of contacts_form.cfm and contacts_result.cfm, which can be found on the ***CFDJ*** Web site, www.sys-con.com/coldfusion/sourcec.cfm. In addition, two XML stylesheets are used that transform the XML result set to HTML and WML – wddx_to_html.xsl and wddx_to_wml.xsl, respectively.

For ease of discussion, I created a fairly simple application that takes some input from the user, performs a query, returns that result to XML, and then transforms the XML into the appropriate output for the desired device.

The contacts_form.cfm source code contains an HTML form for HTML devices and a WML form for WAP mobile devices. The <cfif> tag first checks to see if this is a device that supports WML. If it does, the WML form is built, otherwise the HTML form is built. First, the HTML form will be examined along with the output, then after a brief overview of WML, the WML form and subsequent output will be reviewed.

We could have used XSLT to transform the user input forms from XML to the appropriate HTML or WML, but since the forms are statically built and quite simple, it didn't make sense to do so. This is something that you may want to consider in more complex applications.

Support for HTML Devices

The HTML form that's built to query the user for input is quite basic. As the code in contacts_form.cfm illustrates, the form uses a standard HTML form and Input tag to gather user input, then passes that input as a form field to the contacts_result.cfm page.

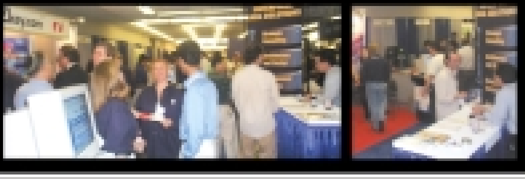
Displaying the results of the query is where things get much more interesting. To display the matching contacts from the database, the single ColdFusion file, con-

Essential Technologies for Today's B2B Integration Challenges

Web Services EDGE

PRE-REGISTER ONLINE TODAY **SAVE $300**

SEPTEMBER	New York, NY	OCTOBER	Santa Clara, CA
web services EDGE conference & expo EAST COAST	24-25	web services EDGE conference & expo WEST COAST	24-25
JDJ EDGE conference & expo	23-26	XML EDGE conference & expo 2001	22-25



each venue features:

- *Over 100 Information-Packed Sessions*
- *Two Distinct Conferences for Just One Registration Fee*
- *Pioneers and Visionaries*
- *Leading Vendors Presenting New and Problem-Solving Products*

ColdFusion fast track
SEPT. 24-25, 2001
TWO
CONCENTRATED DAYS OF LEARNING TO JUMPSTART YOUR PROFESSIONAL CREDENTIALS!

Participating Companies, Sponsors and Exhibitors as of June 1, 2001

TogetherSoft, SilverStream, sitraka software, sonic software, aligo, KENAMEA, shinka technologies, SAVAJE, INFERDATA CORPORATION, bea, InCert, SEAGULL, Data Quad, idoox, CAPE CLEAR, ParaSoft, MERANT, PointBase, hp invent, pingtel, TopCoder, NOVA LABS, ROGUE WAVE SOFTWARE, Bestning, Rational, DataMirror, CodeHound.com, OASIS, ORACLE, DATA JUNCTION, ENTEGRITY Solutions, TOWER, xml spy, TALKING BLOCKS, WHITEHILL, ADVANTEL, XML.ORG, XMLCities, XML JOURNAL, Programmer's Paradise, COLDFUSION Developer's Journal, WebSphere, WebServices JOURNAL, WebLogic, webtechniques, JAVA DEVELOPER'S JOURNAL, SD Times, wireless

Secure Your Seat Now!

Two Easy Ways to Register

1 On the Web: **www.SYS-CON.com**

2 By Telephone: **201 802-3069**

Be sure to visit www.SYS-CON.com for travel information. Reserve Your Hotel Room Early!

SESSION TOPICS, SPEAKERS AND TIMES ARE SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

THE XMLEDGE 2001 INTERNATIONAL CONFERENCE AND EXPO IS IN NO WAY RELATED TO XMLEDGE BY MOBILEQ, INC., A SERVER SIDE SOFTWARE PLATFORM FOR A COMPLETE WIRELESS INTERNET INFRASTRUCTURE.

JAVA AND JAVA-BASED MARKS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF SUN MICROSYSTEMS, INC. IN THE UNITED STATES AND OTHER COUNTRIES. SYS-CON PUBLICATIONS AND SYS-CON EVENTS ARE INDEPENDENT OF SUN MICROSYSTEMS.

tacts_result.cfm, is used to query the database, convert the ColdFusion query to XML, and then transform that XML into either WML or HTML using an XSLT processor.

The actual process of transforming the XML is completed with a series of Java calls from lines 34–56 of contacts_result.cfm. The <CFOBJECT> tag instantiates the Java object and then the methods are called to perform the actual transformation. All the Java code was embedded directly into the ColdFusion page in this example to illustrate how this can be done completely with ColdFusion. We've also implemented this as a pure Java Web service that can be run from JRun or another Java application server and called from ColdFusion.

The first section of the code checks to see if it's a WML device that's requesting the page. If it's not, the code defaults to HTML. The XSL and content type for HTML is set up and the form field, contactName, is prepared for the query statement.

Next the query is executed and the results are stored in the contacts ColdFusion query object, which is then converted to WDDX via the simple ColdFusion tag, <cfwddx>. Since the XSLT processors will operate on any compliant XML code, I decided to go from WDDX directly into HTML and WML as opposed to going into a more pure form of XML first.

Traditionally, XML output is in the following form of name and value pairs.

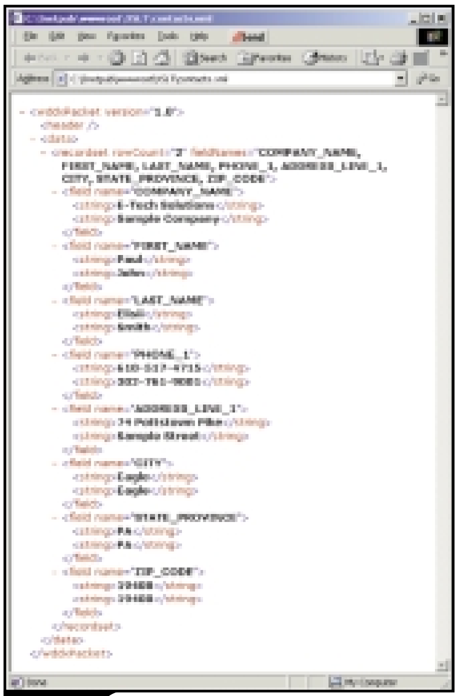
```
<record1>
<company><E-Tech Solutions>
<name><Paul Elisii>
</record1>
<record2>
<company><Sample Company>
<name><Sam Smith>
</record2>
```

This makes writing the XSL stylesheet quite straightforward and easy. However, WDDX organizes its data a bit differently, which makes writing the XSL stylesheet a bit more complicated.

Figure 3 displays the WDDX that we're converting. As you can see, each field node contains all the values in the result set, then the next field, and so on.

When we output the results we want them grouped by each individual record, so we create a looping mechanism in the XSL stylesheet code that regroups the fields in each record set. It'll be very useful to refer back to this WDDX as we examine the XSL stylesheet code.

Also, in writing the XSL stylesheet it was important to stay generic and not refer to any specific fields within the WDDX packet. By creating it so that it can generically take WDDX and convert it to



```
- <wddxPacket version="1.0">
    <header />
  - <data>
    - <recordset rowCount="2" fieldNames="COMPANY_NAME,
        FIRST_NAME, LAST_NAME, PHONE_1, ADDRESS_LINE_1,
        CITY, STATE_PROVINCE, ZIP_CODE">
      - <field name="COMPANY_NAME">
          <string>E-Tech Solutions</string>
          <string>Sample Company</string>
        </field>
      - <field name="FIRST_NAME">
          <string>Paul</string>
          <string>John</string>
        </field>
      - <field name="LAST_NAME">
          <string>Elisii</string>
          <string>Smith</string>
        </field>
      - <field name="PHONE_1">
          <string>610-517-4715</string>
          <string>302-761-9081</string>
        </field>
      - <field name="ADDRESS_LINE_1">
          <string>74 Pottstown Pike</string>
          <string>Sample Street</string>
        </field>
      - <field name="CITY">
          <string>Eagle</string>
          <string>Eagle</string>
        </field>
      - <field name="STATE_PROVINCE">
          <string>PA</string>
          <string>PA</string>
        </field>
      - <field name="ZIP_CODE">
          <string>19480</string>
          <string>19480</string>
        </field>
      </recordset>
    </data>
  </wddxPacket>
```

FIGURE 3: WDDX of contacts query

WML and HTML, I was able to add or remove fields from my query and not have to change the corresponding XSL stylesheets. The trade-off is that you can't be as specific with some of the formatting. In most real-world applications you'll have to be more specific about the XSL to get the formatting you're looking for. The WML example XSL document illustrates this.

The source code that illustrates the XPath and XSLT code used to transform the WDDX packet into HTML is found in the wddx_to_html.xsl file (also found on the ***CFDJ*** Web site). The XSL stylesheets for HTML and WML are very similar – they both parse through the WDDX packet in the same way. The differences are in how they output HTML- and WML-specific content.

The first template in the XSL (lines 5–13) gets processed on the child node of the root. In the case of the WDDX packet, this processes the <data> section. Since there's only one <data> section in the WDDX packet, this gets processed only one time. This is where the HTML <body> and <table> tags are created and where a call to process the other templates is made.

The first template processed is template match="recordset". This template parses through the field nodes creating the HTML content. The first challenge in parsing through the WDDX packet is to create a looping mechanism that loops from one to the number of rows in the packet. A simple way to accomplish this is to loop through the child nodes of the first child record of <recordset>, see the following:

```
<field name="COMPANY_NAME">
 <string>E-Tech Solutions</string>
  <string>Sample Company</string>
</field>
```

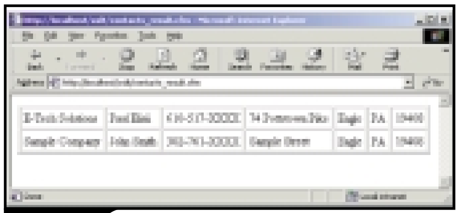
This looping construct loops through the data so we can extract the values for each of the data columns for the first row of data, the second, and so on.

Within the loop for each row of data in our WDDX packet we first get the value-of position() and store it in a variable. Then we begin our <tr> since we'll have one row in our table for each row of data in the WDDX packet. We have a second loop that loops through each column of data and then end the </tr>.

Within the second loop we take the $row variable and pass it into a third template called *field* that outputs the actual data to HTML. In our example, this template gets called 14 times (two rows of data with seven columns).

The field template (<xsl:template name="field">) is where the actual data elements are output into the HTML stream. The child node of the <field> node in the WDDX packet is the <string> node, which in our case contains the data that outputs the particular column and row it's called with. As we'll see in the WML example, special formatting will be performed in this template depending on which column is being processed.

The HTML version of the output is displayed in Figure 4. This is basic output to illustrate the basic concept of transforming XML into HTML.



E-Tech Solutions	Paul Elisii	610-517-XXXX	74 Pottstown Pike	Eagle	PA	19480
Sample Company	John Smith	302-761-XXXX	Sample Street	Eagle	PA	19480

FIGURE 4: HTML query results

A nice things about creating the XSL generically is that simply changing the query to add a fax number, for example, will change the output without changing any of the XSL code.

Support for WAP Devices

Before discussing the WML code in this sample application, I'll present a brief overview of WML.

WAP is a protocol for developing wireless applications. Its two main components are WML and WMLScript. They're similar in concept to HTML and JavaScript, but with some key syntactical and functional differences.

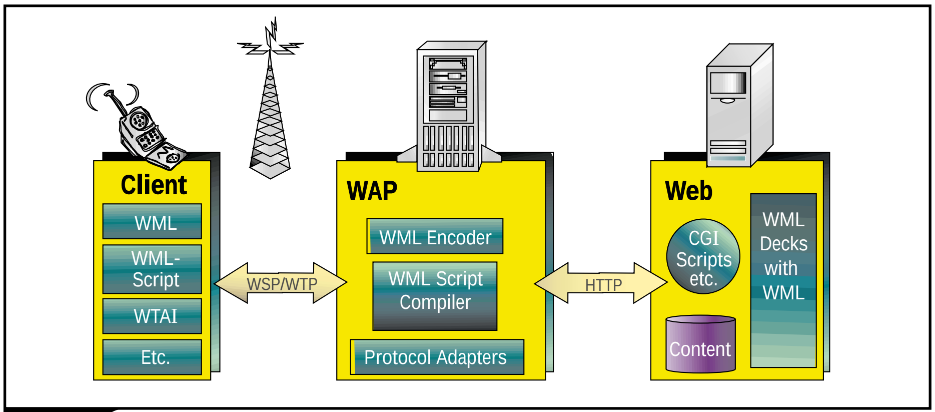


FIGURE 5: Mind map with one circuit app expanded

Two great sources for information and tutorials on WAP and WML are www.wap-forum.org and www.wap.net.

The basic architecture of WAP consists of a wireless client device (phone or PDA) that communicates via a wireless communication protocol to a WAP gateway. That WAP gateway then uses protocol adapters and compilers to communicate with a Web server, which communicates with an application server (in our case ColdFusion) (see Figure 5).

As a variation on this theme application servers that are coming onto the market have a built-in WAP gateway as part of their single-bundled product.

WML and WMLScript

WML and WMLScript are the languages used for WAP applications. WML, a form of XML, instructs wireless devices on how to display content. WML pages operate as cards in a deck, and a single page is referred to as a deck. Each card in the deck corresponds to a single screen on your phone. You can actually perform an HREF call from one card to another within a deck to navigate the client through your application.

The following is a simple example of a WML application:

```
<?xml version="1.0" encoding="utf-
8"?>
<!DOCTYPE wml PUBLIC "-
//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml_1.1.
xml">

<wml>
 <card>
  <p>
   Hello Wireless World!
  </p>
 </card>
</wml>
```

Many tags within WML are unique and specific to the language; however, other HTML and WML tags are the same. You have to be cautious, however, because not *all* devices that support WAP support *all* the tags available.

A great reference on WML and WMLScript can be found at http://developer.openwave.com/resources/index.html. In addition, you can also download the Openwave SDK, which allows you to test your WAP applications without owning a wireless device. It can be downloaded from http://developer.openwave.com/download/index.html.

The SDK installs pretty easily and allows you to download a wide variety of phone configurations/skins that emulate different WAP-capable phones. I downloaded the Mitsubishi T250 skin, which is the exact WAP phone that I'm currently using.

WAP Version of the Application

From the WAP emulator I can enter the URL (localhost or on the Web) just like a browser, and I can run the application just like it would run on a the actual WAP device. Running the exact same contacts_form.cfm page that was run from the browser produces the result shown in Figure 6.

On WAP-capable phones the actions that can be executed are usually on the upper-left and right-hand sides of the phone with corresponding keys. On the Mitsubishi phone the two buttons on the left- and right-hand sides of the Menu button control the actions.

To run the application via the phone emulator use your mouse to click on the corresponding keys, spelling out your words on the keyboard (much easier!).

The application functionally works the same way it did on the browser. I can enter the last name or company name that I'm searching for on the form, then click the left soft-key button to execute the Find action. That action executes a new page to display the results.

The output on the WAP device doesn't use a table and is formatted a bit differently. I decided to make the company name bold and to build in the ability to scroll the results by using the Next action on the phone. As you can see in Figures 7 and 8, the first page is returned and hitting Next displays the subsequent contacts by scrolling through the individual cards in the deck.

The XSL code for the WML version is contained in the wddx_to_wml.xsl file (see the ***CFDJ*** Web site). The structure and flow of the XSL for the WML output is very similar to the HTML version. Both parse through the WDDX packet the same way.

The first difference you'll notice is that the document type is different. The document type allows the WAP device to recognize this as WML output.

For the main document tag we use <wml> instead of <html>, and then for each record output we create a new card instead of a new row in our HTML table. To create the <card> tag with additional and dynamic elements, we use the XSLT <xsl:element> tag that starts on line 16 and continues to line 32. Each card in a WML deck must have a unique name, so the card is created as a combination of "contact" and the current position() being processed, which produces the following output.

```
<card id="contact1">
```

Now the remainder of the card is built. Each column in the current contact is processed by the same field template that was used in the HTML XSL. The XSL used for the WML is more specific with manipulating the data elements. For example, the company name is bolded by using an <xsl:choose> on the element name and then performing different formatting for each element. If the @name is COMPA-

FIGURES 6, 7, 8: WAP input form (6), WAP iquery results (7,8)

NY_NAME, I use the <strong> tag to make the output bold. If the the @name is either CITY or STATE_PROVINCE, we use the <xsl:text> tag to insert a blank space. With all other fields we just add a <br/> tag for a new line. The WML output produced is the following:

```
  <p>
    <strong>E-Tech
Solutions</strong><br/>
    PaulElisii<br/>
    610-517-XXXX<br/>
    610-518-1582<br/>
    74 Pottstown Pike<br/>
    Eagle PA 19408<br/><br/>
```

When we come back to the lines after the <xsl:for-each> tag loops through the columns, we'll add an action for this card. Again, we use the xsl:element tag from lines 26–31. We'll build a WML <do> accept action with a label of Next. When this action is processed, it executes an HREF to go to the next card in the deck. We have to identify which card we're going to, and in this case, it will be the next contact, contact2. On line 30 we concatenate "contact" to the current position()+1 to produce the following result:

```
<do type="accept" label="Next">
    <go href="#contact2"/>
```

The <xsl:element> tags are then terminated. This produces the following tags to complete the card for contact1.

```
</do>
  </p>
 </card>
```

Summary

Support for wireless mobile devices will increase rapidly in the next two years, and the demand for applications that support these devices will increase accordingly. Wireless application support will soon become a natural part of all Internet applications, as it is in Netscape and IE browsers.

Using XSLT, applications can be built to support multiple devices and allow developers to quickly add support for additional devices without having to rewrite the application.

Technologies and Resources

- ***ColdFusion***

All examples were built using ColdFusion 5.0. However, no explicit features of ColdFusion 5.0 were used. All the examples ran under other versions of ColdFusion without difficulty.

- ***IIS 5.0***

All examples were run using IIS 5.0. A modification had to be made to IIS to allow it to recognize WML devices. The following content type had to be added: text/vnd.wap.wml

- ***UP SDK***

The SDK was downloaded from http://developer.openwave.com/download/index.html and all examples were run with the Mitsubishi T250 configuration/skin.

- ***Saxon XSLT Processor***

Version 6.2.2 of the Saxon processor was downloaded from http://users.iclway.co.uk/mhkay/saxon. The class files were copied to a directory in the class path so they could be called from ColdFusion.

About the Authors

Paul J. Elisii is the CEO/CTO of e-Tech Solutions, Inc., a Philadelphia-based e-business development company and a Macromedia Premier Partner. Paul is active in developing and researching emerging technologies for e-Tech Solutions.

Serge Ohotin is a developer with e-Tech Solutions, Inc. He was extremely helpful in performing research and in developing some of the sample applications for this article.

pelisii@etechsolutions.com

sohotin@etechsolutions.com

CFXtras.com is The Component Resource For ColdFusion Developers.

Buy custom tags from CFXtras.com and sell your custom tags too. We handle all the details, and you see the profits in your account. View more detail on the custom tags below at http://www.cfxtras.com.

CFXtras

2simplifi Online Commerce from 2simplifi Web Solutions
2simplifi Online Commerce is a complete ColdFusion e-Store Application. The application features the ability to apply site wide discounts, tax rules, shipping cost ranges, dynamic product rotation and a wish list for prospective shoppers. $282.00
Component ID: 113

Podium V4 from Polix, LLC
Podium is an open source bulk emailing application for CF 4.5+/SQL7. Delivers 200 million emails in a 24-hour period, without native CF mail tags. Rich feature set, customizable, free phone support. $4799.00
Component ID: 53

2simplifi All Jobs from 2simplifi Web Solutions
2simplifi All Jobs - New Features: FAST APPLY - 1 Click Job Apply for existing Job Seeker Members & Confidential Resume Postings. $282.00
Component ID: 74

CFX_Zip from EmTek Systems
CFX_Zip is a ColdFusion add-on that allows you to enhance your ColdFusion applications by providing full support for industry standard ZIP files. $49.00
Component ID: 23

CFX_Spell from EmTek Systems
CFX_Spell is a complete ColdFusion spell checker engine that allows CF developers to provide server side spell checking within their applications. Add-on dictionaries and languages available. $149.00
Component ID: 24

Remote Developer from Network Web Applications Inc
Remote Developer is an online IDE with server administration tools that allow you to administer and develop a web site from any location. $199.00
Component ID: 129

BTCalendar from Basstech Consulting Group
BTCalendar is a highly configurable calendar application that has been designed to be implemented quickly and easily into any ColdFusion application. It is capable of integrating with any database table used to store date and/or schedule–related information. $59.99
Component ID: 82

webmaster@cfxtras.com • 704.408.6186

Missed an issue?

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

COLDFUSION Developer's Journal

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

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

***CFDJ* The Complete Works**
~~Reg $79.99~~
Online Only
$71.99

XML JOURNAL

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

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

***XML-J* The Complete Works**
~~Reg $59.99~~
Online Only
$53.99

JAVA DEVELOPER'S JOURNAL

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

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

***JDJ* The Complete Works**
~~Reg $119.99~~
Online Only
$71.99

COLLECT ALL 3
JDJ
XML-J
CFDJ
~~REG. $259.97~~
ONLINE
ONLY $175.99

Order Online and Save 10% or More!

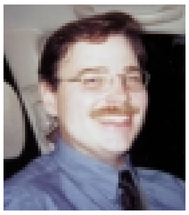
www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SAVE 10% ONLINE JDJStore.com SAVE 10% ONLINE JDJStore.com SAVE 10%

Activedit 2.5 by CFDev.com

REVIEWED BY
CAREY
LILLY



Some time ago, when Homesite 1.0 was still my editor of choice (yes, *that* long ago), I had a problem with a client. A small-town newspaper editor wanted a wider audience – the Web.

Naturally, she knew nothing about HTML or FTP, and she wasn't about to learn. I thought, "Create a database." I could use an HTML form to collect the text and, voilà, instantly updated pages. But how could she format the text without learning <br> and <b>?

The boss nixed the idea as too expensive and time-consuming. I probably wasted enough time converting her text to do the database option twice. But I'm not paid to make the Big Decisions.

Actually, I couldn't get the problem out of my mind. A client with no other knowledge (except how to get online and how to use a word processor) needed to be able to create and edit HTML documents on a remote server. The only solution is a Web-based WYSIWYG editor that can save in HTML.

Problem solved. Activedit 2.5 from CFDev.com does the trick. You get a fully functional HTML editor that you just plug into a form.

Installing Activedit

If you have access to your own server, or if your Web space provider is nice enough to install it, you'll be in business fast. *Note:* To use the spell-check feature, you'll need a Java Virtual Machine (the documentation recommends the Java 2 Runtime Environment, available at http://java.sun.com/products/).

Even if you don't have access to your server's custom tags folder, you can still use Activedit. Simply upload the activedit.cfm file to the folder where you'll be using it. You'll also need to upload the /inc folder that contains support files.

In my test I used the nonserver method at my host. As far as I could see, there was no discernible difference in performance.

Using Activedit

Activedit is a CF custom tag that you include in any CF form page. It creates a text editor inside your form. Activedit, at its simplest, looks like this:

```
<cf_activedit fieldname="con-
tents" inc="inc/"></cf_activedit>
```

To explain what you're seeing: "fieldname" is simply that – the name of the form field. That fieldname gets passed, HTML code and all, into your form handler; "inc" is the folder containing Activedit's components and image files. If you want to include some predefined input, it can be inserted via a CF variable. Other attributes allow you to modify the toolbar or enable and disable different options.

From this point, assuming you don't need to allow image uploading or spell checking, you can use the basic tag as if it were a <textarea> form tag. Add an appropriate CFFILE tag in your form handler, like so:

```
<cffile action="write"
file="#destination_dir#\con-
tent.html" nameconflict="over-
write" output="#form.contents#">
```

Now you have a perfectly usable HTML editor inside your form that writes to a static HTML page. Of course, you can also save the results into a memo field in your database or to an e-mail. You may want to include a validator to extract any potentially harmful code. I was able to paste a little JavaScript into my form. Setting "alloweditsource=no" would probably take care of it.

On the other hand, your more savvy clients may appreciate being able to add tables and hyperlinks.

Unfortunately, space doesn't permit me to go into all the details of this product. I haven't even touched on the Java-based spell checker or image uploading.

Who can best use this? Honestly, if you maintain a Web site, whether your own or a few dozen for your clients, Activedit can dramatically alter the way you do business. Imagine a Web site like the one I mentioned at the outset: a newspaper editor needs weekly (maybe daily) updates. Imagine she can edit her site online and you never have to hear from her (except for the checks). Or imagine sending formatted HTML e-mails using your client database. Imagine a Web site where users can browse and create and maintain their own personal Web pages without ever calling or e-mailing you.

Is Activedit worth it? If you want hassle-free content management and reduced customer hand-holding, it's the way to go. But to me, just the "wow" factor would be worth the money.

VITALS

Activedit 2.5
CFDev.com

Address: 181 Genesee Street
Suite 500
Utica, NY 13501

Web: www.CFDev.com

Phone: 800 791-1916

Test Environment: NT4 server with IIS and CF4.5

Pricing: I tested an open source version with spell checker for $475. The basic ColdFusion version is $99 ($175 with spell checker).

ABOUT THE AUTHOR
Carey Lilly is an associate with a Web site–development firm based in the New York area. He has been developing with ColdFusion since 1997 and has 10 years' experience with relational databases.

CAREY@WORLDCONTACT.COM

SPECIAL INTRODUCTORY OFFER

SAVE $31*

HURRY, DON'T DELAY! OFFER EXPIRES: DECEMBER 31, 2001

Premiering... this *fall*

subscribe Now!

FOR FAST DELIVERY

Go Online and Subscribe Today!

The World's Leading Independent WebLogic Developer Resource

Helping you enable inter-company collaboration on a global scale

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

FOR WLS DEVELOPERS BY WLS DEVELOPERS
WebLogic
DEVELOPER'S JOURNAL
weblogicdevelopersjournal.com
An introduction to...
WebLogic Server Clustering
Web Logic fast track
Welcome to WLDJ
Avoiding Performance Pitfall with Entity EJBs
ASK BEA
TIPS & TRICKS
JNDI Initial Contexts
TECH SUPPORT

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

SYS-CON MEDIA

ColdFusion and XML-RPC Part 1 of 2

Connect applications built in different technologies

BY RONALD WEST

Application development has changed dramatically in the past few years. Applications are now more robust, encompass many types of technologies, and integrate into frameworks that are not Web-enabled.

With the changes that occur in the strength of the applications and in the technology, it's important to have a means of integration. As new technologies arise it's easy to forget about older, less extensible technologies.

We need application development that can share data across different platforms and technologies. We need to package data to be handled by disparate applications and a means of transmission.

Enter XML-RPC

XML-RPC encapsulates data into XML packets that are sent from one server to another. This creates a system that allows two technologies to share data without knowledge of the other's proprietary API. The data is packaged in a universal format so that each side only needs to understand what "kind" of data they're working with (is this a structure with keys *x*, *y*, and *z*, or is this an array with elements in position 1–5). The data structure is the only concern and the two systems can handle it in their own separate yet equally powerful ways.

"Combining two or more applications that have been built from separate technologies produces enormous benefits"

Combining two or more applications that have been built from separate technologies produces enormous benefits. Take two applications that offer similar services to the same clients. Combine the applications and you create a suite of services that allows both companies to market communally – a win-win for all parties involved.

Applications that offer similar services to a particular marketplace are often considered competitors since they frequently share the same clients. When two organizations share the same market space and the same clients, it could also be viewed as an opportunity for them to work together.

If the underlying principles of application deployment technologies are similar, business practices can be enhanced with the manipulation and presentation of information; thus two applications, previously viewed as incompatible, can now work together. In addition, the effort required to coordinate the two applications is greatly reduced. The two organizations need to work on a uniform data format that can be shared. Existing processes can handle the data manipulation and presentation, and there's no need to abandon supported technologies or hire personnel.

In the case of Internet applications, the users receive a best-of-breed set of applications that have passed through the "proof-of-concept" stages and generally worked out all the kinks. Therefore the users of the system are empowered with a more robust set of applications that allows them to do their job in a more efficient manner.

Now, the organizations that have purchased technologies of a certain breed are not tied down to that platform and can easily utilize new technologies as they become available. In turn, they feel that the applications that have been purchased and integrated are not obsolete as they can be effortlessly connected to future applications.

Both organizations that previously created a single application have effectively doubled their resources. They no longer need to plan out the business logic, provide development resources, and wait through the testing periods for a proven product. Instead the organizations can locate market-proven applications that extend their current application and simply integrate them into their own. This saves all the work that was required to create the integrated applications (content experts who supported the design and implementation process that built the business logic the application was structured from).

Now let's get to the fun part – the solution.

Defining the Systems Involved

The system I'm working with is comprised of a ColdFusion application that shares data with a JRun application, which in turn communicates with a PHP application. The definition of each system is provided later. The key is, it really doesn't matter what the systems are. I'll try to be as generic as possible as this type of connectivity can be used in almost any vertical market; the application I'm working with is workers' compensation.

web services EDGE conference&expo™

October 24–25, 2001

XMLEDGE conference&expo 2001

October 22–25, 2001™

Santa Clara Convention Center Santa Clara, CA

FREE Registration

Register for one...attend both

See them First at XMLEdge2001

Fundamentally Improving the Speed, Cost & Flexibility of Business Applications

Many New XML Products Will Hit the Market in Q3 of 2001

Opening Keynoter

Charles Goldfarb

The Father of XML

SYS-CON MEDIA

Owned & produced by SYS-CON EVENTS

WEB SERVICES CEO PANEL

Litwack CEO SilverStream	**O'Connor** President Sonic Software	**Kutay** CEO AltoWeb	**O'Toole** Chairman Cape Clear	**Slama** CEO Shinka Technologies	**Morris** CEO IONA Technologies

www.SYS-CON.com

Platinum Sponsor: Gold Sponsor: ADOS Silver Sponsor: SilverStream Corporate Sponsors: Association Sponsors: OASIS XML.ORG

Media Sponsors: XML Journal, Java Developer's Journal, WebServices, WebLogic, WebSphere, ColdFusion, wireless, SD Times, Programmer's Paradise

A ColdFusion application was developed that records certain demographic information about injured workers and determines through a series of questions which types of tasks are required to help manage this claim. For example, one of the tasks could be *Fraud Investigation*.

Instead of creating an application that would allow access to information that's necessary to complete the investigation, we'll work with an organization that has the application already built. We'll integrate our applications so that information can be shared directly from ours to theirs, utilizing certain critical business processes without having to learn a great deal about them. In our example we'll work with the ColdFusion application server, a CF template, a servlet, and the JRun application server. We'll discuss one specific data module, checkSuburb, and work our way through the process step-by-step and explain how an application written in CF can share data with applications written in almost any other technology.

First we take the data that checks whether the Australian State and the Suburb is a valid match (checkSuburb module). The data consists of a structure with several keys. These keys contain string data that's used to determine a match (is the State a part of the given Suburb?). If you define the data that will be shared before deciding how it will get from one place to another, you'll understand the full capabilities of XML-RPC; it's not the structure of the data in transit, it's merely the structure of the data before and after transport. It's similar to a teleportation system. The data is in one chamber and consists of certain properties. When the data is teleported to the other chamber, all the properties and structures remain intact. We don't care what happens to the data when it goes between the two chambers, we simply care that the end result is a data object that retained its value.

Establishing a Connection with JRun and the Servlet

The ColdFusion application server can communicate directly with a JRun application running on any platform. The servlet tag sends structured CF data to JRun via a TCP/IP request along a specific port to the servlet, which will be processed by JRun.

```
<cfservlet code=MyServlet.class
jrunProxy=127.0.0.1:8081
timeout=30
debug=true>
</cfservlet>
```

The code attribute signifies the class file that you'd like to exchange data with, and the jrunProxy is the IP and port for the JRun external Web server listener. The other two attributes are fairly self-explanatory.

The cfservlet tag, which is placed in a ColdFusion template and processed like a cfmodule or a custom tag, can pass and receive data. Due to the architecture, any session, application, or server variables that the servlet needs to work with must be passed in.

The cfservlet tag can pass data to a servlet two ways – by value or by reference. For Java users this concept won't be new. However, for those non-Java developers I'll briefly explain the difference as it applies to this particular model.

To pass data by value you're allowing the servlet to access a variable and inherently its value at any particular time. Change the value of the variable and the servlet receives the new value. This is useful for data variables that we don't want the servlet to interact with or modify. For instance, we may send someone's address to a servlet that sends out automatic mailings. The servlet needs the data to perform the task, but the ColdFusion template doesn't want the servlet to modify any of the data. The cfservlet tag has an associated tag named cfservletparam. This tag is embedded within the opening and closing cfservlet tag and is used to send the data to the servlet. In the case of data sent by value, a sample cfservletparam tag might look like this:

```
<cfservletparam
 name=myData
 value=#variables.myData#>
```

The name attribute is the name of the variable sent and the value is obviously the value. The process of sending data by value is akin to sending data through a form – you have a name value pair that's sent through. For instance, in Listing 1 we have a variable named *data* that contains a structure with two keys, State and Suburb, each with a string. The variable data is sent to the servlet for processing as a name-value pair. I'll discuss the actual exchange in Part 2 of this article.

Note that we may send different "types" of data – strings, numbers, dates. Table 1 offers a conversion chart that maps ColdFusion and Java variable types. It's important to understand the difference between a string and an integer. For example, in ColdFusion it's possible to do the following:

```
<cfset myNum="5">
<cfoutput> #evaluate("#myNum# *
   2")#</cfoutput>
```

It looks as though myNum begets a string with a value of 5

"We don't care what happens to the data when it goes between the two chambers, we simply care that the end result is a data object that retained its value"

Type	In Java
int	java.lang.Integer
double	java.lang.Double
bool	java.lang.Bool
date	java.util.Date
string	java.lang.String
array	java.util.Vector
structure	java.util.Hashtable
ColdFusion query test	com.Allaire.util.Recordset (a WDDX-supplied utility class)

TABLE 1 Conversion chart

because of the double quotes; however, ColdFusion treats this as a string and an integer, which allows it to be evaluated in an arithmetic function. In Java a string can't have arithmetic evaluations performed on it, so if passing integers, it's good to keep them distinguished. *Note:* The documentation specifies which type of data is sent via an additional attribute named *type*. However, combining the name, value, and type attributes throws an error.

To utilize the strengths of Java and have a servlet modify data from a CF template, we need to send the data by reference. Essentially, we allow the servlet to receive the data in its original format, do some work on it, and send the new data back to CF. To send the data by reference we'll use the same cfservletparam tag that we used when we sent data by value, however, this time we'll use different attributes. Therefore, a cfservletparam tag used to pass data by reference will look similar to the following:

```
<cfservletparam
 name=myData
 variable=theData>
```

The name attribute in this case will be the name of the variable to be sent to the servlet tag. The variable attribute represents the ColdFusion variable that will be modified by the servlet tag. In Listing 1 we establish a blank variable, "result", that we pass through to the servlet by reference. The result variable will be used by the servlet to send data back to the CF template. Again I'll discuss this exchange in Part 2.

Summary

So far we've established that it's possible to connect to a JRun server and work with a Java servlet using the cfservlet tag. The cfservlet tag has a subtag, cfservletparam, that allows us to send data to the servlet by value or by reference. In Part 2 I'll discuss how that data is transmitted and the JRun architecture.

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

RWEST@ACESTECHNOLOGIES.COM

Listing 1

```
<!---- // this will send a check suburb request

data structure:

Structure with following keys:
-) Suburb ( String )
-) State ( String )

notes:

variations:

 // ---->

 <cfscript>
 data=structNew();

 Suburb="Bondi";
 State="NSW";

 structInsert(data, "Suburb", Suburb);
 structInsert(data, "State", State);

 methodCall="checkSuburb";

 result = structNew();
 result2 ="";

 </cfscript>

<cfservlet code="Connect" jrunproxy="#JRunProxy#"
debug="Yes" timeout="15">
  <cfservletparam name="methodCall"
value="#methodCall#">
 <cfservletparam name="data" variable="data">
 <cfservletparam name="result" variable="result">
 <cfservletparam name="result2" variable="result2">
 </cfservlet>
```

CODE LISTING

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

RED HORIZON

www.sourceaction.com

BY BEN PARSON

Building VoiceXML Applications

You've used ColdFusion to build forms that send e-mail and build database-driven applications, and maybe even played with some WDDX. Now you're probably looking for a new and different way to use the skills you've gained. VoiceXML could be just the ticket.

As a Web developer, I'm always interested in new technologies and finding different ways to apply the knowledge I've gained. A few months ago I discovered VoiceXML and my curiosity was piqued.

This article is a basic introduction to VoiceXML. It'll get you started – and make you dangerous. Excellent resources on the Internet, included at the end of the article, will take you further. I won't teach you anything you don't already know about ColdFusion; in fact, the ColdFusion examples are simple. Instead, my intent is to expose you to VoiceXML development and show you how you can use ColdFusion to implement dynamic voice applications.

What Is VoiceXML?

If you're not already familiar with the XML in VoiceXML, there's an excellent primer on XML at the TellMe Web site. (See the Resources section for the address.)

So where does the voice part come from? VoiceXML allows you to build applications with a voice interface. We've all called a technical support line and been asked to browse through common menu options using our TouchTone phone. You can build applications like that with VoiceXML, but it's nothing new. That technology is as old as TouchTone phones!

The real advantage to VoiceXML is that it brings the technology to people who are traditionally more familiar with building Web applications. Since it's based on XML, VoiceXML will look familiar to you as a ColdFusion developer. Using VoiceXML, you can build interactive voice applications using voice recognition and synthesized speech. You can even build applications that use voiceprint identification. Remember the movie *Sneakers?* "My voice is my passport, verify me."

Using ColdFusion to implement dynamic voice applications

VoiceXML Platforms

To run your voice application, you need to have it hosted by a VoiceXML platform. Two of the most common are the TellMe Studio (http://studio.tellme.com) and the BeVocal Café (http://cafe.bevocal.com). Both provide free access to develop and host one application. When you get to the point of wanting to deploy your application, you can discuss pricing with them based on your needs. The examples in this article are based on the TellMe platform.

Hello World VoiceXML Application

As always, the first thing you have to write is the obligatory Hello World application. I've taken the liberty of including the sample code from the examples at TellMe networks to show you this (see Listing 1). More about TellMe later.

The <vxml> tag is the VoiceXML equivalent of the <html> tag. You begin and end your VoiceXML document with it. After stating some metainformation, the code begins with the <form> tag.

You're already familiar with forms from HTML, but they're used a little differently here. In VoiceXML a form is a container for fields that handle interactions with your users. The difference is simple: in HTML you use forms as a way to input data to your application. In VoiceXML forms are used both to prompt for data input and to present the output.

A VoiceXML form is actually closer to being a Windows development form than one used for Web development. Here you'll simply use a form as a container to output the "Hello World" phrase to the user.

The <block> tag sets up an executable context, telling the interpreter to present information to the user. If the code reads <block>Hello, world!</block>, the interpreter would have used text to speech to output this to the user. But that would be cheesy. For our high quality application you don't want some scratchy computer-generated voice.

Instead, you're going to use the <audio> tag, a tag that's used to output a .wav file to the user or to output text to speech. In this case you're specifying a .wav file to be played (http://resources.tellme.com/audio/misc/hello.wav), but if the specified file doesn't exist, you'll use text to speech to output the string "hello world". You do this by enclosing that text between the open and close of the <audio> tag.

After this you pause (the application), and then use the <audio> tag to say "goodbye" to the user. By not specifying the path to a .wav file here, you're telling the interpreter to use text to speech. After pausing again, use the <goto> tag to return the user to the main menu of the TellMe service.

How Does ColdFusion Fit?

The Hello World application is a basic one that simply outputs some data to the user. To provide real value, you need to tie a VoiceXML application into your database. You can use ASP, PERL, or any number of Web development technologies to do this, but I believe ColdFusion is best suited to the task. It has a proven track record for rapidly building database-enabled Web applications, and its strengths apply here as well.

A VoiceXML application is hosted on a platform like those of TellMe or BeVocal. However, these platforms are able to call out to your Web server to get code dynamically. As long as your ColdFusion application outputs valid VoiceXML, everything works great. Your application can call a ColdFusion page that looks up some data based on user input and returns the results to the platform in VoiceXML.

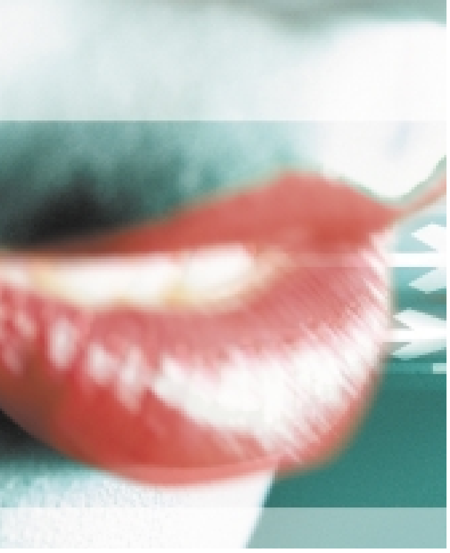
Building a Sample Application

To see how this works, we'll build a sample application that will prompt a user for the name of a U.S. state. Once that's done, we'll return some specific information about that state. This will let us show a few different ways in which ColdFusion can interact with the VoiceXML platform.

VoiceXML Basics

You need to know a few common terms to start building voice applications. First, you should know what a "grammar" is.

Have you ever used a voice recognition application? When you first start with these applications, you have to go through a



"Remember the movie *Sneakers*? 'My voice is my passport, verify me'"

training period in which the application learns how you say certain words so it can best recognize what you're saying. In most cases you won't have that luxury when you build a voice application. So how can you use speech recognition with any degree of accuracy?

In VoiceXML a grammar is a list of options that the application will listen to and allow to be spoken as input. For example, if you were to ask a user to say a favorite month, you'd build a grammar that accepts the values January, February, March, and so on. The system would accept only the values you provide in your grammar. Anything else spoken by the user would be ignored. This way, your application can use voice recognition technology without having to go through the training phase, yet still have a high degree of accuracy.

Both the TellMe and BeVocal platforms use the Nuance Grammar Specification Language (GSL). You can learn the details from their sites or on the Nuance site (www.nuance.com). Nuance also provides a free developer's program you can apply for.

Let's look at a sample grammar to get started (see Listing 2). Grammars are placed inside the <grammar> tag. The CDATA section protects the following section from being recognized as markup by the parser. Your grammar description is placed inside that block.

You're telling the application that the words contained inside the brackets of the grammar are possible options for the user to speak. If you look at the first line of the sample grammar, these options are "dtmf-1" and "massachusetts". The state names are not capitalized intentionally because your grammars should be lowercase. You know that "massachusetts" is a state name, but what is "dtmf-1"?

DTMF stands for dual-tone multi-frequency and is basically the TouchTone that you normally associate with traditional phone applications.

You're saying that the user can either say the number "one" or press it on the phone. They could also say "Massachusetts" and it would return the same value. In the curly braces we can see the value that will be returned from the grammar. Here it's the number one, which happens to be the ID of the state "Massachusetts" in the database. This way, you can do a lookup in the database using the primary key to find the detailed information on the state.

Notice the difference in the entries for New Hampshire and Rhode Island. When you're expecting a phrase from the user rather than a single word, you need to put it in parentheses. By replacing the space with an underscore in the grammar, you allow the user to slur the state name together as if it were one word. You still need to give users the option in parentheses of saying it correctly as two words as well.

Dynamic Grammars

Now let's get into some of the good stuff. There's another way to reference a grammar without including it between <grammar> tags. You can link to an external grammar, much like referencing stylesheets in HTML. The syntax for this is:

```
<grammar name="StateList"
src="http://www.yoursite.com/gram-
marfile.gsl"/>
```

It's beneficial to include grammars like this because the file you include could be any type of file as long as it outputs a valid grammar. Here's where you get to start using ColdFusion. You need to duplicate the sample grammar in Listing 2 dynamically using the database of states. You can download the sample database and source code from ***CFDJ***'s Web site (www.sys-con.com/coldfusion/sourcec.cfm).

As in XML, every tag in VoiceXML must have both an opening and a closing tag, as in <grammar></grammar>. However, some tags don't take information between the tags, or the information between tags is sometimes optional. In these cases you can use shorthand to open and close the tag. In the preceding example the grammar tag closes itself with a forward slash so that <grammar></grammar> is the same as <grammar/>.

Listing 3 is an example of a ColdFusion page that will output your dynamic grammar. You start with a basic database query, selecting the data from the states table. The next section of code would be a normal CFOUTPUT with one exception: you have to handle the case of state names that have more than one word.

This example checks for a space in the state name. If one is found, it outputs the state name as one word, replacing the spaces with underscores for those of us who slur. Then the state name is still output as a phrase for those of us who enunciate our words.

Dynamic grammars can get complex and can include thousands of entries – the more you have, the longer it will take your application to recognize commands from the user. However, the GSL language is extensive, and you can tune your grammars to handle these situations.

Imagine writing a grammar for an application that allowed users to say, "What appointments do I have on Tuesday?" or "Read me the subjects of my unread e-mail from yesterday." While complex, you can build grammars capable of handling these situations. The cause for concern for the developer is determining the level you want to go to in allowing user input.

To make your grammar easier to write but less user-friendly, the "What appointments do I have on Tuesday?" grammar could be written to accept the word *appointments* and then prompt users with a second grammar to choose the day that interests them. If you'd like more details on grammars, read "VoiceXML: Grammars for VoiceXML Applications" by H. Seth (***XML-J***, Vol. 2, issue 5). See the resources at the end of this article for more information.

Building the Form

Now that you've built your dynamic grammar, you need to build the form that uses the grammar to prompt the user and get input. You'll do this by using a VoiceXML form with one field (see Listing 4). In the example, just after the metatag, a variable of "StateID" is declared that will remain persistent across both forms in the application. After that, the form is created and given the name "StateLookup".

Special Introductory Offer SAVE $13.89*

WebServices JOURNAL

The world's leading independent Web Services technology 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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

PREMIERE ISSUE

WebServices JOURNAL

www.wsj2.com

SYS-CON Events presents

From the Editor by Jack Martin pg. 5

Guest Editorial by Ajit Sagar pg. 7

Case Study by Robert Diamond pg. 70

Success Stories by Brady Flowers pg. 92

Tips & Techniques by Sean Rhody pg. 94

WSJ News pg. 98

Interview by Robert Diamond pg. 104

IMHO by Jason Westra pg. 110

Feature: Is It Really XML's Killer App? *Real-world Web Services are here to stay* — Brady Flowers 22

Security: Authentication, Authorization and Auditing *Support for any kind of e-business* — Jack Martin 30

Wireless: Enable Your Web Services with WAP *Finding common ground* — Sean Rhody 38

Tutorial: How to Develop and Market Your Web Services *A means for generating more traffic* — Ajit Sagar 46

WSDL: Definitions and Network Endpoints *Open standards-based technology makes the difference* — Israel Hilerio 54

Feature: Deploying EJB in a Web Services Environment *Run on any platform* — Robert Diamond 62

Web Services and Swing: An Introduction *High-performance translations* — Jason Westra

Feature: Implementing SOAP-Compliant Applications *Exploring technology*

SYS-CON MEDIA

www.wsj2.com

SYS-CON MEDIA

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 Web Services.

Next, a form field called "StateID" is created that will prompt users for the state they're interested in. The link to the dynamic grammar is inserted. The <audio> prompt is now used to ask users for a specific state.

Remember that in a production application you should always have professionally recorded audio for any static messages you present to the user. Even some dynamic content can be prerecorded for a more professional touch. We're going to go with the text to speech, so our <audio> tag doesn't include an "src" reference.

After the <prompt> tag, there are a few other options that the parser will go to depending on what happened in the prompt. These tags are event handlers that are relatively self-explanatory, but we'll go over them anyway. The code in the <nomatch> section is run when the user's input doesn't match any of the entries in the grammar. The user is told that the application didn't understand what was said, and is reprompted for the same information.

The code in the <noinput> section is run if the user didn't say anything or if the application couldn't hear what the user said because of too much background noise. It will also reprompt the user if this occurs.

The <help> tag is special in that the user can say "help" whenever prompted for information. I say "special" because it's something the application will recognize even though it wasn't explicitly allowed in the grammar. This is supported as long as you put code in the event handler for it.

The code under the <filled> section runs when the application recognizes the user's input and returns the value specified in the grammar. Here it will be the ID of the state, and the "StateID" variable will be set to the value returned here. The <goto> tag is then used to jump to another form named "StateDetailLookup" that will make an external call to another ColdFusion page for a database lookup.

This form begins by declaring an executable block of code. After doing this, a local variable of "StateID" is declared and sets its value to the "StateID" returned from the grammar. The submit tag is then used to make the call out to our ColdFusion code to obtain a new VoiceXML document via HTTP. Similar to including a grammar, this could be either a static file hosted somewhere else or it could be a dynamic ColdFusion page.

You can pass variables to your pages by using the GET or POST methods of HTTP. Here the parameter is passed in the query string as a GET request. The method is specified, and the "namelist" parameter is set to "StateID". Here only one value is being passed in the "namelist" field. If there were more values, they would be included as a space-separated list. The application closes with the </vxml> tag to complete this portion.

Dynamic VoiceXML

Let's go into some more ColdFusion development by looking at Listing 5, a dynamic result to the external VXML call. The code begins with a simple CFQUERY, looking up the information from the "StateCaps" table based on the "StateID" field passed in the URL.

The VoiceXML document is then built just as if it were built on the platform itself. It starts with the <vxml> tag and goes on from there. A new form called "result" is created that will output the results to the user. There is some error-checking code to make sure that data is actually returned before the results of the query are output to the user.

"The real advantage to VoiceXML is that it brings the technology to people who are traditionally more familiar with building Web applications"

At the end of this is a <goto> tag that sends the user back to the last anchor. If you recall, in the "StateLookup" form created in Listing 4, the "anchor" attribute was set to "true". This is the last place an anchor was set, so this is where the application returns.

We could mix ColdFusion code and VXML throughout our document, just as we would with HTML or WML. The only difference between these technologies is the format of the output. Most of us have developed different versions of the same code for various reasons, whether to support multiple platforms (WAP phones, or Palm or Windows CE devices) or even just to support different Web browsers. The same techniques we used then could be used to build a ColdFusion application to support both Web and voice browsers.

What Could I Build with VoiceXML?

By now you're probably thinking about which of your ColdFusion applications you could port over to VoiceXML. There's probably a good place for you to use your skills and enhance applications as well. Let's go through some samples.

One thing you could build is an employee list. You could have a dynamic grammar that lists all employees' names as options and gives their detailed information as a result. Or how about building an order- and shipment-tracking application that could be implemented as the hold dialog when customers call in? You could then look up their most recent orders and shipments using their ANI information and read them back the status. (*ANI* stands for Automatic Number Identification and while different from Caller ID, it offers similar functionality.)

Another possibility would be to build a VoiceXML front end to your e-mail package, allowing your users to access their e-mail from any phone. You could get really fancy and tie it into their calendars as well. If you really want a fun project, sign up with the BeVocal Café and start playing with the Voiceprint Identification. Use that to authenticate your regular users.

Sharing the Inspiration

This article's primary purpose is to give you a basic understanding of VoiceXML and spark your interest enough to build an application. If you'd like more information and development resources, see the links at the end of the article. Also, if you're going to Macromedia DevCon this year, they'll have a session entitled "Delivering Dynamic, Voice-Driven Applications Using VXML." If you do feel inspired to build a voice application, let me know how it's going by e-mailing me. I'd love to hear about what you've done.

Resources

1. *TellMe Studio:* http://studio.tellme.com/
2. *BeVocal Cafe:* http://cafe.bevocal.com/
3. *Nuance:* www.nuance.com
4. *VoiceXML Forum:* www.voicexml.org/
5. *From the W3C:* www.w3.org/Voice/#implementations)
6. Seth, H. (2001). "Tools for Developing VoiceXML Apps." ***XML-Journal***, Vol. 2, issue 3.

About the Author

Ben Parson, a senior Web developer for PowerQuest Corporation, focuses mainly on e-commerce, Web development, and database design. He has been building Internet database applications for the past seven years.

ben.parson@powerquest.com

Listing 1: Hello World Application

```
<?xml version="1.0"?>

<!--
Tellme Studio Code Example 101
Copyright (C) 2000 Tellme Networks, Inc. All Rights
Reserved.

THIS CODE IS MADE AVAILABLE SOLELY ON AN "AS IS" BASIS,
WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITH-
OUT LIMITATION,
WARRANTIES THAT THE CODE IS FREE OF DEFECTS, MERCHANTABLE,
FIT FOR A
PARTICULAR PURPOSE OR NON-INFRINGING.
-->

<!DOCTYPE vxml PUBLIC "-//Tellme Networks//Voice Markup
Language 1.0//EN"
"http://resources.tellme.com/toolbox/vxml-tellme.dtd">

<vxml>
<meta name="scoping" content="new"/>
<form>
<block>
<audio
src="http://resources.tellme.com/audio/misc/hello.wav">
 hello world
</audio>
<pause>500</pause>

<audio>
 good bye
</audio>
<pause>500</pause>

<goto next="_home"/>
</block>
</form>
</vxml>
```

Listing 2: Sample Grammar

```
<grammar>
 <![CDATA[
  [
  [dtmf-1 massachusetts] {<option "1">}
  [dtmf-2 new_hampshire (new hampshire)] {<option "2">}
  [dtmf-3 connecticut] {<option "3">}
  [dtmf-4 rhode_island (rhode island)] {<option "4">}
  ]
 ]]>
</grammar>
```

Listing 3: Dynamic Grammar Using ColdFusion

```
<cfquery name="StateCaps" datasource="States">
 select State, ID from StateCaps
</cfquery>

[
<cfoutput query="StateCaps">
 [<cfif FindOneOf(#State#, " ", 1) gt 0>
   #Replace(LCase(State), " ", "_", "ALL")#
(#LCase(State)#)
  <cfelse>
   #LCase(State)#
  </cfif>] {<option "#id#">}
</cfoutput>
]
```

CODE
LISTING

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

CFXTRAS
www.cfxtras.com

PACIFIC ONLINE
www.paconline.net

COLDFUSION Developer's Journal

home advertise subscribe contact SYS-CON MEDIA

<dot.com>

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

<magazine>

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

<content>

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

<conferences>

- coldfusion edge fast track new york, ny sept 23–26

<search cfdj>

Search

<cfdj specials>

Collect All Three!

JDJStore.com
$175.99

JDJ, XML, CFDJ - ALL THREE!!! The Complete Works

<bestsellers>

1. Goetz's Net Tools Pro Kit from Lawrence Goetz $19.99
2. JRun Studio 3.0 from Macromedia $449.99
3. Homesite v4.5 Win 32 from Macromedia $93.99
4. ColdFusion 5.0 Server Enterprise $4,095.00
5. ColdFusion 5.0 Server Professional $1,295.00

What's Online

www.sys-con.com/coldfusion

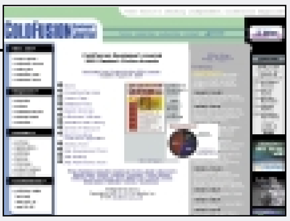
September **2001**

CFDJ Online

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

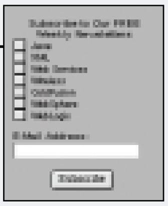
Readers' Choice Awards – Vote Now

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



Subscribe to Our Free Weekly Newsletters

Now you can have the latest industry news delivered to you every week. SYS-CON newsletters are the easiest way to keep ahead of the pack. Register for your *free* newsletter today! There's one for ColdFusion, Java, XML, Web Services, and Wireless. Choose one or choose them all!



Subscribe to Our FREE Weekly Newsletters
- Java
- XML
- Web Services
- Wireless
- ColdFusion
- WebSphere
- WebLogic

E-Mail Address:

Subscribe

JDJ Store CD Special

The complete library of ***CFDJ***, ***JDJ***, and ***XML-J*** articles are available on CD at a special price for a limited time.

Order now and have more than 1,000 articles on hand for research and review. There are features, how-tos, product reviews, tips and tricks, interviews, IMHOs, and more!

CFDJ topics include Tips from Ben Forta, Custom Tags, ColdFusion and Java, Server Stability, Site Performance, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Application Frameworks, Error Handling, Wireless ColdFusion, Ask the Training Staff, Authentication, Monitoring, Smart Objects, A Beginner's Guide to CF, Safe Scripting, JavaScript, and Load-Balanced Servers.

This exclusive package normally sells for $260, but it can now be yours for only $175.99 for a limited time. Order today and save!

Search ColdFusion Jobs

ColdFusion Developer's Journal is proud to offer our 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.

QUICK POLL

What is Your Favorite New Feature of ColdFusion 5.0?
- User-Defined Functions
- Query of Queries
- Application Deployment
- Integrated Charting
- Performance Improvements

Vote

Results

Click for a FREE 30-day trial! AltoWeb

click here to DOWNLOAD e-TEST suite empirix

INTERMEDIA.NET NT web hosting FREE setup: code CFWEB

AbleCommerce FREE TRIAL

CFDynamics QUALITY ColdFusion HOSTING

conceptware

IS YOUR WEBSITE MISSING IN ACTION

Podium v4 can! Free demo at CFXtras.com

FREE Domain Names FREE ECommerce Software

Cold Fusion Hosting virtualscape

DOWNLOAD COLDFUSION

Letters to the Editor. . .

Plaudits for Christian Schneider

Thank you for the article "Live Monitoring of User Sessions" [***CFDJ***, Vol. 2, issue 8]. I've implemented the live monitoring system on my company's Web site. I wanted to enhance this functionality by showing the name of a page where each user is. I know it can be done in OnRequestEnd.cfm page, but I can't find any variable that will give me the name of the page. Can you throw some light on this?

Girish Bhagia
gbhagia@Envestnet.com

Thank you for your feedback. It's nice to hear that my article is accepted that much ;-).

To track the page name, open up a structure indexed by user IDs (of ip-addresses) and insert an OnRequestEnd.cfm "#CGI.SCRIPT_NAME#" into them (giving you the name of their page). However, don't forget to clear out timed-out users from this structure. You can get a list of all available variables by opening the expression builder inside Studio (using CTRL-SHIFT-E)and scrolling down to the variables section.

Hope that helps.

Chris
mail@Christian-Schneider.de

I'm a new subscriber to ***ColdFusion Developer's Journal*** and recently came across your article "Live Monitoring of User Sessions." I've been looking for tools to manage and monitor the users of our Web site and found your article fascinating! Since we have a clearly defined user base – it's for employees only – our concern is having too many users hanging onto our pages without logging out. I've often thought that someone should put together a management tool that would not only allow us to monitor the session variables but also other information regarding the activities of the user.

Michael Toepfer
Michael.Toepfer@unh.edu

I've enjoyed your article "Instant Messaging for Your Apps" [***CFDJ***, Vol. 2, issue 11] about session tracking and instant messaging. I was wondering if you were willing to share your instant messaging forms with me in order to understand it all and possibly "tweak" it.

Richard Liendecker
rliendecker@lmslogistics.com

Plaudits for Bruce Van Horn

I look forward to your "Ask the Training Staff" column. Sometimes the most simple (fundamental) questions are answered in a short paragraph. Three out of the four questions you listed in the February issue [Vol. 3, issue 2] solved some of my problems. That makes this a valuable source of information. I hope ***CFDJ*** continues this column. Keep up the good work!

Robert W. Stampfle
rstampfle@dqe.com

Keep Up the Good Work

First let me say that I really enjoy ***ColdFusion Developer's Journal***; it's always full of useful information and help.

When I get to Robert Diamond's editorial page I'm always struck by his picture. I must be getting older than I think. Is that really him, today or several years ago? I assume he gets asked that question frequently.

Keep up the good work with ***CFDJ***.

Richard Cox
coxri4@mail.northgrum.com

Editor's Note: *Periodically a company whose product we review has some points they wish to contend.*

KM Tools Responds

I would like to clarify a couple of points made by Carey Lilly in his review of cfx_kmSuite [***CFDJ***, Vol. 3, issue 6].

1. Regarding "potentially hazardous system-access functions" – the hazardous functions are grouped in a separate, optional module: cfx_kmsystem. This means it doesn't have to be installed on public servers, thus protecting exposed servers from abuse.
2. "Some functions of the suite duplicate existing CF functions." The LIST functions are "alternate" versions, not duplicate. They were written to address specific problems with the existing CF LIST functions. This is clearly stated at the top of the module documentation for cfx_kmList (www.hoptechno.com/kmtools/).
3. In my view the real strength of cfx_kmSuite is the comprehensiveness and consistency of the functions. It would have been nice if the article pointed out that cfx_kmSuite contains more than 120 functions. This is a big advantage over using a random assemblage of freeware, commercial, and ad-hoc custom tags.
4. Finally, we are always responsive to suggestions from users for additions to the suite.

Phil Dunn
www.hoptechno.com/kmtools/

CFDYNAMICS

www.cfdynamics.com

Your Own Magazine

Do you need to differentiate yourself from your competitors?
Do you need to get closer to your customers and top prospects?
Could your customer database stand a bit of improvement?
Could your company brand and product brands benefit from a higher profile?
Would you like to work more closely with your third-party marketing partners?
Or, would you simply like to be a magazine publisher?

SYS-CON Custom Media is a new division of SYS-CON, the world's leading publisher of Internet technology Web sites, print magazines, and journals.

SYS-CON was named America's fastest-growing, privately held publishing company by *Inc. 500* in 1999.

SYS-CON Custom Media can produce inserts, supplements, or full-scale turnkey print magazines for your company. Nothing beats your own print magazine for sheer impact on your customers' desks... and a print publication can also drive new prospects and business to your Web site.

Talk to us!

We work closely with your marketing department to produce targeted, top-notch editorial and design. We can handle your distribution and database requirements, take care of all production demands, and work with your marketing partners to develop advertising revenue that can subsidize your magazine.

So contact us today!

East of the Rockies,
Robyn Forma,
robyn@sys-con.com,
Tel: 201-802-3022

West of the Rockies,
Roger Strukhoff,
roger@sys-con.com,
Tel: 925-244-9109

SYS-CON CUSTOM MEDIA

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

SYS-CON MEDIA

RePrints

SAVE 16% Off the annual cover price

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Receive 12 issues of *CFDJ* for only $89.99! That's a savings of $17.89 off the annual subscription rate. Visit our site at *www.sys-con.com* or call 1-800-513-7111 and subscribe today!

ANNUAL NEWSSTAND RATE
~~$107.88~~
YOU PAY
$89.99
YOU SAVE
$17.89 Off the Annual Newsstand Rate

Here's what you'll find in every issue of *CFDJ*

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

Increase your CF knowledge at
CF Advisor
COLDFUSION Developer's Journal
Getting CF Flash
CFDJ Feature: Creating Intelligent
404 Pages
HTML E-Mail: Harnessing the Power

The World's Leading Independent WebSphere Developer Resource

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE NOW AND SAVE $31.00 OFF THE ANNUAL NEWSSTAND RATE

ONLY $149 FOR 1 YEAR 12 ISSUES REGULAR RATE $180 FOR 12 ISSUES

SUBSCRIBE NOW AND SAVE $31*

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

www.WebSphereDevelopersJournal.com

SYS-CON MEDIA WebSphere DEVELOPER'S JOURNAL

www.wbt2.com

www.javadevelopersjournal.com

www.xml-journal.com

www.coldfusionjournal.com

www.powerbuilderjournal.com

www.webspheredevelopersjournal.com

www.wldj.com

www.wsj2.com

subscribe online www.sys-con.com or call 800 513-7111 SYS-CON MEDIA

wireless | java | xml | coldfusion | powerbuilder | websphere | weblogic | web services

Next Month...

ETesting
Tips and testing tools for debugging
by Charles Arehart

Untrusted Data Sources
Protecting your ColdFusion applications
by Jackson Moore

A Cold Cup O' Joe *Part 6*
Java CFX debugging
by Guy Rish

Using MS-SQL Stored Procedures with ColdFusion
Really useful…and not as difficult to work with as you might think
by Ian Rutherford

COLDFUSION Developer's Journal

Don't miss the October issue!

CFDJnews

PaperThin Releases CommonSpot 2.5; Partners with AJJA I.T.C.

(*Boston, MA*) – PaperThin, Inc., launched its newly enhanced ColdFusion-based content management application, CommonSpot Content Server 2.5.

The new customer-driven features include an ASP/hosting solution, increased security functions, and multilanguage support.

PaperThin also announced a partnership with AJJA Information Technology Consultants, Inc. As a PaperThin partner, AJJA will distribute CommonSpot in Canada, and provide business and training solutions for existing and new CommonSpot users.
www.paperthin.com
www.ajja.com

Macromedia Sitespring Available

(*San Francisco, CA*) – Macromedia, Inc., introduced Macromedia Sitespring, a new Web-based application for managing the Web site production process.

Macromedia Sitespring offers an integrated, server-based approach to team collaboration, file management, and client communications. A new product extension allows Sitespring users to see their project tasks from within Macromedia Dreamweaver and Dreamweaver UltraDev.
www.macromedia.com

O'Reilly Releases *Programming ColdFusion*

(*Sebastopol, CA*) – *Programming ColdFusion* by Rob Brooks-Bilson covers everything users need to know to create effective Web applications with ColdFusion, and includes numerous examples that programmers can use for their own applications. The book starts with ColdFusion basics and quickly progresses to topics such as sharing application data, accessing databases, and maintaining state information. It also explores more advanced topics such as creating custom tags, sharing data with WDDX, and calling external objects.

For more information see www.oreilly.com/catalog/cold fusion/index.htm

Sybex Releases *Mastering ColdFusion 5*

(*San Francisco, CA*) – Sybex Inc. is distributing *Mastering ColdFusion 5* by Arman Danesh, Kristin Aileen Motlagh, and Raymond Camden. This book provides comprehensive coverage of all the new and improved features. The accompanying CD includes evaluation copies of ColdFusion 5.0 for Windows, Linux, and Solaris; JRun Server 3.1; and all the example code from the book.

www.sybex.com/ sybex-books. nsf/booklist/2979.

SYS-CON Media Acquires CFAdvisor

(*Montvale, NJ*) – **SYS-CON Media** has acquired the assets of ***CFAdvisor***, a leading online ColdFusion e-zine. "***CFAdvisor*** has been a well-respected online ColdFusion resource since its inception three years ago," said Fuat Kircaali, founder and CEO of **SYS-CON Media**.

"We will continue building ***CFAdvisor*** and maintain its world-class editorial content and resources for ColdFusion developers."
www.sys-con.com

CFDev.com Releases @JMail; ASP Version of Activedit 2.5

(*Syracuse, NY*) – CFDev.com, a division of NETDesign Inc., introduced @JMail, a ColdFusion CFX tag that replaces CFMAIL for bulk e-mails, mailing lists, newsletters, and more. It's capable of sending up to 1,000 e-mails per minute, and works on all ColdFusion supported platforms.

The company also released Activedit v2.5 for the ASP environment. This tool enables Web developers and integrators to increase the efficiency of their Web publishing and content management capabilities.
www.cfdev.com

Upcoming Events

ColdFusion Edge 2001 Fast Track/JDJEdge 2001 International Java Developer Conference & Expo
September 23–26, 2001
New York, New York
www.sys-con.com/coldfusion-edge/

Fusebox Tutorial Conference
October 20, 2001
Orlando, Florida
www.cfconf.org/fusebox2/

Macromedia DevCon 2001
October 21–24, 2001
Lake Buena Vista, Florida
www.cfconf.org/DevCon2001/

ADVERTISER INDEX

ADVERTISER	URL	PH	PG
ABLECOMMERCE	WWW.ABLECOMMERCE.COM	360.253.4142	2
ACTIVEPDF	WWW.ACTIVEPDF.COM	888.389.1653	4
CFDYNAMICS	WWW.CFDYNAMICS.COM	800.422.7957	59
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX.HOST	9
CFXTRAS	WWW.CFXTRAS.COM	704.408.6186	57
COLDFUSION EDGE 2001	WWW.SYS-CON.COM/COLDFUSIONEDGE	201.802.3069	12,13
CORDA TECHNOLOGIES	WWW.CORDA.COM	801.805.9400	3
DTN FINANCIAL SERVICES	WWW.FINWIN.COM	800.652.2291 X.2259	11
EVOLUTIONB	WWW.EVOLUTIONB.COM/CASESTUDIES		63
FUSEBOX	WWW.FUSEBOX.COM	212.929.7644	17
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	23
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	64
JDJEDGE CONFERENCE & EXPO	WWW.SYS-CON.COM/JDJEDGE	201.802.3066	32,33
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	45
MACROMEDIA	WWW.MACROMEDIA.COM/GO/DEVCON01		21
MACROMEDIA	WWW.MACROMEDIA.COM/DOWNLOADS	888.939.2545	29
MACROMEDIA	WWW.VUE.COM/MACROMEDIA	888.460.8679	35
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	57
FALL INTERNET WORLD	WWW.INTERNETWORLD.COM	203.559.2866	19
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	16
RED HORIZON	WWW.SOURCEACTION.COM		51
SYS-CON MEDIA REPRINTS	WWW.SYS-CON.COM	201.802.3026	61
WEBLOGIC DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/WEBLOGIC		47
WEB SERVICES EDGE 2001	WWW.SYS-CON.COM/WEBSERVICESEDGE	201.802.3066	41
WEB SERVICES JOURNAL	WWW.SYS-CON.COM/WEBSERVICES		61
WEBSPHERE DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/WEBSPHERE		47
XMLEDGE CONFERENCE & EXPO	WWW.SYS-CON.COM/XMLEDGE	201.802.3066	49
YES SOFTWARE	WWW.CODECHARGE.COM	650.754.9810	41

EVOLUTIONB

www.evolutionb.com/casestudies

INTERMEDIA.NET

www.intermedia.net