U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

May 2002 Volume: 4 Issue: 5

Full Conference Program INSIDE page 42

SYS-CON MEDIA

ColdFusion CFMX – A First Look

by Ben Forta page 6

ColdFusion Developer's Journal

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)
by **SYS-CON Publications, Inc.**

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

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

CFMX is the Word…

BY **ROBERT DIAMOND**

As a jaded computer user and developer, it's hard to excite me with a simple new release. As a magazine editor, with products and demos flying across my desk, it becomes even harder. I've seen many products and releases come and go over the past few years, and it's been a very interesting journey. I've seen what I thought were great products fail miserably and a few rather mediocre ones that are now widely adopted because they managed to fill a special niche.

It's rare that a product comes along that really excites me – that I'm playing with a new feature and my mind is going a mile a minute thinking of ways to use it – that my imagination just goes wild over the potential I think something has.

I'm happy to say that this is one of those rare occurrences. With that glowing introduction, welcome to the May issue of ***CFDJ***, where we start our ground-breaking official coverage of ColdFusion's next release. Actually, it doesn't feel quite right to refer to it simply as "ColdFusion's next release." Having participated in the beta program for a number of months, I've found it to be so much more. The hype we've been hearing for so many months has finally been lived up to, and the offhand phrase doesn't give Macromedia – and the very talented developers who created the new product – as much credit as they deserve. In fact, if you're a ColdFusion developer (and since you're reading *CFDJ*, I think it's safe to assume you are), it's going to be more than enough to knock your developer socks right off.

The normal life-cycle of a software product tends to be a year, with new releases/upgrades annually. For Macromedia, despite the release of ColdFusion 5, ColdFusion MX has been under construction for nearly two years. It therefore represents more than the natural evolution of the product, but rather ColdFusion Reborn, or the next (r)evolution of ColdFusion.

The Java platform…CF's now larger seat on the fast-moving Web services train… .NET support: it's got it all, and more. Macromedia has lit a fire under ColdFusion, and as the details about the product emerge, the Allaire/Macromedia merger is finally living up to its promise. The move that some questioned when it first took place, not understanding what was taking place behind the scenes, is now proving itself. Much credit for the hard work and vision goes to Macromedia's leaders, like Kevin Lynch and CTO Jeremy Allaire, as well as to the now fully integrated product development teams.

Our cover story this month, by **Ben Forta**, explores some of CFMX's many new features. And we'll be going into greater depth in the subsequent issue once Macromedia has made the full product announcement.

The current issue doesn't just give you a nice look at CFMX, though; it's filled with a lot more. "Query Custom Tags in CF" by **Reuben Poon**, a new face here at *CFDJ*, offers tips on simplifying your code. "Using JSP Custom Tags" by **Charlie Arehart** lets you in on a good way to use one of the features in the new CF. **Christian Schneider** shows you how to build a WAP-based e-mail interface. **Hal Helms** writes about recursion, and **Tom Muck** reviews the Click Portal Server from Intrafinity, whose product helps companies operate. Part 2 of **Philip Chalmers**'s article explains how to use CF to solve session management problems. Last, but certainly not least, CFDJList manager **Simon Horwith** writes about some of the issues of the day – this month, SQL Server 2000.

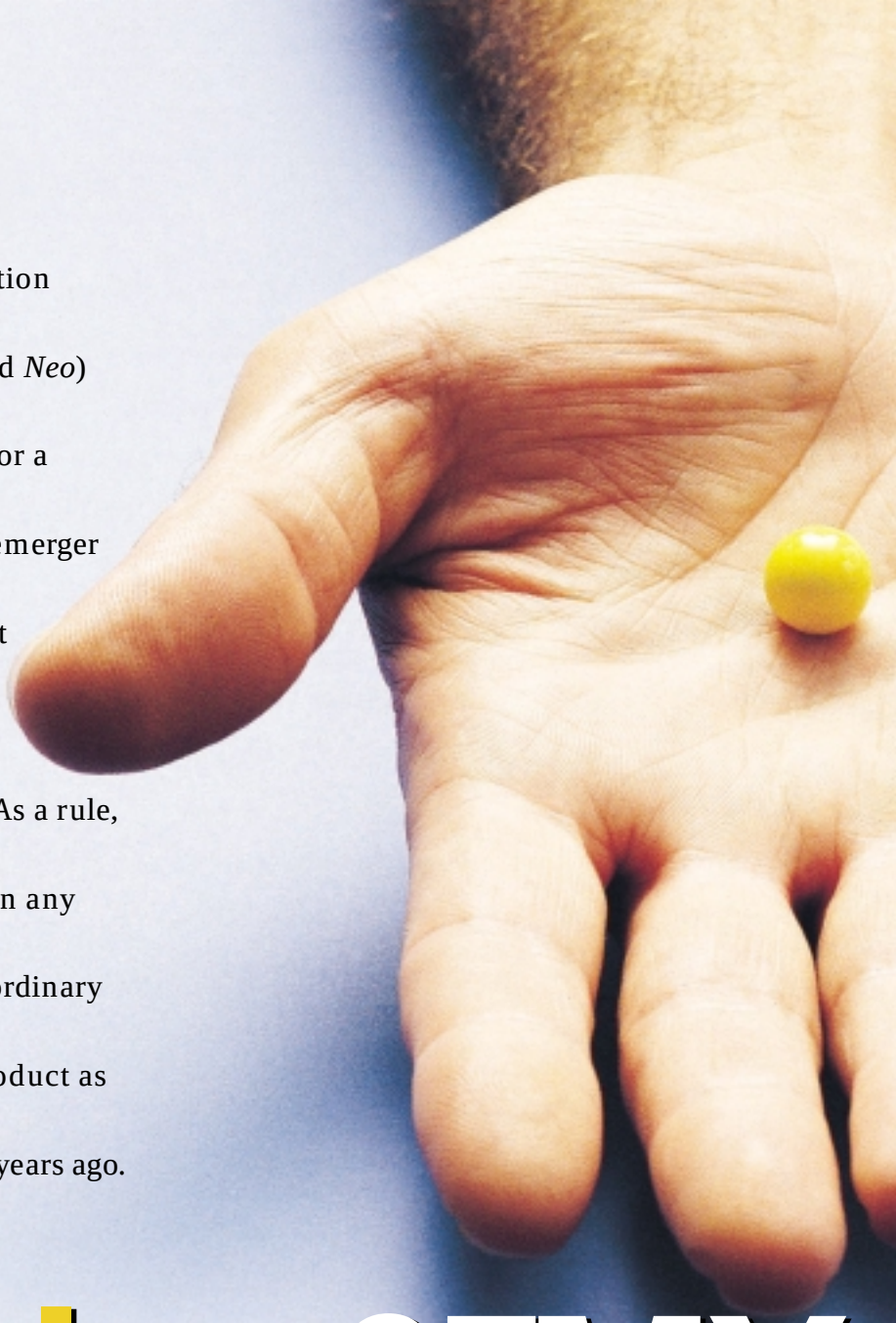
ABOUT THE AUTHOR

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

ROBERT@SYS-CON.COM

ColdFusion Feature

By Ben Forta



It finally happened: the next-generation ColdFusion (previously code-named *Neo*) is in public beta, and is scheduled for a mid-2002 release. We (Macromedia and premerger Allaire) have been talking about this product for a long time, and we demonstrated it first publicly at DevCon 2001 in Orlando. As a rule, future products aren't discussed publicly in any detail (if at all), but ColdFusion MX is no ordinary product – indeed, it's as revolutionary a product as the original ColdFusion was close to seven years ago.

ColdFusion CFMX

So here, with special permission from the powers that be at Macromedia, it is my privilege to give you a first look at ColdFusion MX.

A Brand-New ColdFusion

First and foremost, I must state quite emphatically that CFMX is not just another product upgrade. It truly is a brand-new ColdFusion, built from the ground up, taking into account everything we've learned since creating the very first application server (before that term was even coined). Why did we do this?

ColdFusion 5 is a solid and reliable product. It is without doubt the best ColdFusion we've ever created. It is fast, it is stable, it is secure, it is efficient – and most important, it solves real problems right now. At the same time, developers are finding themselves running into limitations. The fact is, the current product is being used to solve problems that didn't even exist when it was first created, and developers are now asking for features that are simply pushing the limits of the current ColdFusion.

Furthermore, despite the rise and fall of such important buzzwords as *intranets*, *extranets*, *B2C*, *B2B*, *P2P*, and so on, one thing has become blatantly clear: the Internet has proved itself a viable and valuable application platform. With this recognition has come the need for application-development, platform, and infrastructure standards – standards beyond the scope of

– A First Look

ColdFusion itself. These standards (primarily J2EE and .NET) didn't exist when ColdFusion came into being, but they do now. Thus it has become important for ColdFusion to embrace and leverage all they have to offer.

Having read the previous two paragraphs, you should fully understand why we built a brand-new ColdFusion – redesigned, reengineered, and rebuilt from the ground up to do a whole lot more and be a whole lot more capable than its predecessor.

(As a side note, many have been questioning Macromedia's commitment to ColdFusion and the ColdFusion community. It's worth noting that no prior version of ColdFusion has had the kinds of resources that have gone into building CFMX – not just budget-wise, but also the number of developers, the time in development, the amount of QA time, and more. Macromedia is making it very clear that ColdFusion is an incredibly strategic product, and one it is deeply committed to.)

The Power of Java, the Simplicity of ColdFusion

Now you know why we built a brand-new ColdFusion. CFMX is built on top of underlying Java technologies – this has been widely known for quite a while. But what hasn't been as clear is why we did this. After all, Java is slow, right? And it's difficult, isn't it? Oh, and isn't Java a ColdFusion competitor?

Let's go off on a slight tangent. "Java" means different things to different people, but at its simplest Java is two very different things:

- ***Java is a language:*** It is a compiled object-oriented language that promises compiled code portability. It's a much lower-level language than CFML, and is designed to solve a very different set of problems.
- ***Java is a platform:*** J2EE is a complete specification for everything from virtual machines, database support, legacy application integration, and mobile device support to directory service and security services, messaging and transactions, and building reusable components. J2EE is rich, powerful, and, until now, inaccessible to most.

Java the language is not that exciting to ColdFusion developers. For the types of applications built with ColdFusion you'd never want to use Java anyway; it would be a lot more work with no real upside at all.

But Java the platform – now that *is* compelling…until you realize you'll be coding in Java, which is anything but rapid. Imagine the raw power of Java, accessible with the ease of use and simplicity of ColdFusion – that's CFMX.

Java difficult? Not via ColdFusion. On the contrary, using CFMX, the Java platform is finally accessible to the masses. Java is as simple as CFML.

Competitors? No, ColdFusion and Java aren't competitors at all; they're totally different products. Even Java the language and CFML the language aren't competitors because they're designed to solve totally different problems. And CFMX bridges the two worlds.

As for slow….Well, first of all, most of us associate Java with poor performance because of experiences with applets in the late '90s (an experience best forgotten). Having said that, if you were to compare native C or C++ code against Java code running within a Java Virtual Machine, then yes, the Java code would likely be a bit slower than the C or C++ code. But ColdFusion was never C/C++ anyway; the ColdFusion server itself was, but the CFML code you write is not. Your code is interpreted by a C/C++ engine, but the code itself isn't C/C++.

CFMX functions quite differently – it is a true compiler. Your CFML code is compiled into Java bytecode, and compiled code is processed far quicker than interpreted code. So, while there might on occasion be merit to the argument that Java is slower, CFMX is decidedly not.

What does it all mean? As a ColdFusion developer, you have been fortunate enough to be able to build world-class applications quickly and easily. Now you can take those applications to the next level, harnessing all the power of Java while retaining the ColdFusion experience. You'll be able to use:

- JSP tags and tag libraries
- JavaBeans and EJBs
- Java APIs
- And much more

The following code snippet is one I used at DevCon in Orlando to demonstrate JSP tag library use in CFMX. The tag library, downloaded from a public Internet site, creates two-dimensional graphics, like buttons. As you can see, using one new tag, <CFIMPORT>, JSP tag libraries are usable and highly accessible:

```
<!--- Load JSP tag library --->
<CFIMPORT TAGLIB="/WEB-INF/lib/advisor2d.jar"
          PREFIX="2d">
<!--- Default text --->
<CFPARAM NAME="ATTRIBUTES.text"
         DEFAULT="">
<!--- Create image --->
<2d:imageSubmit bgPaint="image-images/buttonbg.png"
    text="#ATTRIBUTES.text#"
    textAlign="RIGHT-BOTTOM"
    paint="red"
    vpadding="5"
    hpadding="5"
    font="15"
    scope="session" />
```

The <CFIMPORT> tag imports a JSP tag library and associates a prefix with it (so it may be used later). Specific tags may then be called using prefix:name syntax. It's as simple as that. Every JSP tag library out there is now available to you as ColdFusion developers. The power of Java, the simplicity of ColdFusion.

The reverse is true too. With CFMX, Java developers have an alternative to endless listings of low-level code to display database data or generate dynamic, data-driven e-mail.

The power of back-end Java, without sacrificing what makes ColdFusion ColdFusion: simplicity and ease of use.

Usable XML

XML has been overhyped. It's not the end-all, it won't solve all problems, and it won't, in and of itself, make the Internet a better place. But XML is valuable in that it can make data exchange and sharing simpler. I say *can* and not *does* because more often than not, reading and writing XML data, walking trees, accessing nodes, and so forth is anything but trivial.

Unless you're using CFMX. New to ColdFusion is a set of tags and functions designed to make working with XML data as simple as using CFML. You can read XML text into ColdFusion objects and convert them back again, create and modify XML objects, extract data from XML objects using XPath expressions, and even apply XSL transformations.

Even better, the ColdFusion XML object is a form of structure, so everything you already know about creating and manipulating structures applies to XML objects too. In fact, most of the CFML struct functions can be used to manipulate XML objects.

As an example, the following code retrieves a database query and then generates XML output containing the retrieved data:

```
<!--- Get user list --->
<CFQUERY DATASOURCE="dsn"
         NAME="users">
SELECT UserID,
       FirstName,
       LastName,
       Email
FROM Users
</CFQUERY>

<!--- Create XML object --->
<CFXML VARIABLE="UserRecs">
  <UserRecs>
    <!--- Loop through users --->
    <CFOUTPUT QUERY="users">
      <!--- Write user --->
      <user>
        <UserID>#UserID#</UserID>

<FirstName>#FirstName#</FirstName>

<LastName>#LastName#</LastName>
        <EMail>#EMail#</EMail>
      </user>
    </CFOUTPUT>
  </UserRec>
</CFXML>
```

Similar operations may be used to read inbound XML data, search for specific values, and more. The following snippet reads and parses an uploaded XML document and then extracts a single attribute, saving it to a ColdFusion variable:

```
<!--- Read uploaded XML file --->
<CFFILE ACTION="read"

FILE="#CFFILE.ServerFileName#"
        VARIABLE="uploadedxml">
<!--- Parse read XML file --->
<CFSET uploadeddoc=XmlParse(uploadedxml)>
<!--- Extract UserID --->
<CFSET accountNum=uploadeddoc.user.
XmlAttributes.UserID>
```

As you can see, CFMX makes working with XML data as simple as CFML itself. But XML support is only one of the new features in CFMX. So why do I bring it up? Read on.

Components and Services

Web services are all the rage now. Web services are simply the next generation of distributed computing, a mechanism by which to create and use bits of applications (or parts of applications) anywhere, even remotely. In services lingo, Web services are *produced* (made available for use), and then *consumed* (used) by other applications.

Web services aren't a ColdFusion invention. In fact, Web services are one of the few technologies that just about every major player in the space is backing and supporting. .NET uses services extensively, and J2EE applications support services too. This broad industry support means that Web services are going to become very important. They're going to succeed.

In addition, Web services employ several key technologies, all of which are open and standard:

- HTTP
- XML
- SOAP (Simple Object Access Protocol)
- WSDL (Web Services Description Language)
- UDDI (Universal Description, Discovery, and Integration)

I'm not going to cover Web services in detail here, but look for lots of coverage in the future. For now I'll work under the assumption that it's agreed that Web services are the next generation of application development.

CFMX makes consuming Web services incredibly simple. In fact, all it takes to consume a service is a single CFML tag (there are actually several ways to invoke Web services, both tag based and <CFSCRIPT> based).

Even more exciting, CFMX makes creating and producing Web services just as simple.

New to CFMX are ColdFusion Components (CFCs, for short), which are a bit like custom tags in that they facilitate code reuse, but they do much more than custom tags. Consider the following:

- CFCs support multiple methods (actions) in a single component.
- CFCs provide structured parameter validation.
- CFC methods can return data.
- CFCs may be invoked locally or remotely.
- CFCs may be consumed by applications and environments other than ColdFusion.
- CFC methods may be secured (using roles and access levels).
- CFCs support reuse via inheritance.
- CFCs may be introspected (they self-document).

And I've saved the best for last…

- CFCs can be accessed as Web services.

CFCs are a new type of ColdFusion file (they even use a new file extension, CFC). They're created using new tags (with self-describing names like <CFCOMPONENT>, <CFFUNCTION>, <CFARGUMENT>, and <CFRETURN>) and can use any and all CFML – the same code you'd use in your applications right now.

Using CFCs you can build truly reusable code – structured, tiered, abstracted, even distributed. And all just using CFML. They're the new building blocks that will (and should) change the way you build ColdFusion applications. And for development teams, CFCs provide an ideal mechanism for distributing the development process across different developers and skill sets, allowing developers to concentrate on what they do best and then making their code accessible to other developers.

Once you've created a CFC, it can be made available as a Web service by adding a single attribute to the method definition. That's it. CFMX does the rest (including generating the WSDL automatically on the fly).

You're going to hear lots on CFCs from me and others in the upcoming months.

Powering Rich Clients

Everyone has heard of Macromedia Flash, which is installed in some 98% of all browsers. Running on mobile devices, new phones, and even game machines, the Flash player is just about everywhere. Yet mention Flash, and most people think of clicking Skip Intro. But that has been changing. CFMX, and the just-released Flash MX, will force developers to completely rethink Flash and its use.

Many of you have seen (or created) Flash movies, often animation clips or widgets (like form controls). But what about Flash as an application client? With the browser (player) so accessible, with portability, with minimal download times, with rich UI control, with the ability to create a much better user experience, Flash becomes compelling as a true application client. (Again, apps like these were demoed at DevCon in Orlando.)

Why is this happening now? Well, there are two primary reasons:

1. For many developers Flash has been somewhat inaccessible. Developers are the ones who create applications (as opposed to movies), and developers think in code and scripts; concepts like time lines, tweening, and even color palettes are foreign to them. Even development environments that use lots of pop-up boxes and palettes (as opposed to being code-centric) make die-hard developers a little weak in the knees. So Flash MX has been designed to appeal to developers too, offering new and improved ways to write and implement code-driven Flash.
2. The process of passing data back and forth between Flash on the client and a back-end server hasn't been exactly trivial or painless. In fact, there's no special integration between ColdFusion 5 and Flash 5. They can be used together, just as other application servers and technologies can. But with CFMX ColdFusion becomes the ideal back end to power Flash on the client.

Which brings us back to components. I mentioned earlier that CFCs could be invoked remotely, outside of ColdFusion. Using ActionScript and a few lines of code, Flash applications can invoke CFCs as if they were local objects. ActionScript code simply refers to the object locally. The fact that it's at the other end of a data connection is transparent and irrelevant…it just works.

With Flash made more accessible to developers, and with sophisticated yet simple support embedded right within CFMX, Flash truly can become the next-generation client for complete (and very rich) applications.

Summary

CFMX is nothing short of revolutionary. From its reengineering on top of the J2EE platform, its XML support, and its support for components and thus Web services and .NET to its Flash MX integration, the ColdFusion development team has built a product that will change the way you think about application development. And this is just the start of it. They've added lots of other features too, including many enhancements to existing features. In the future I'll be delving into lots of CFMX specifics, so stay tuned.

About the Author

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

BEN@FORTA.COM

Handling **Recursion**

For tidier apps, eliminate duplicate code

BY
HAL HELMS

Recursion is a wonderfully geeky word that comes from the Latin word *recursio*, meaning "to run back."

A recursive procedure (or function, or subroutine) is a procedure that calls itself to do part of the work. The ancients showed the concept of recursion as a snake eating itself.

For a more modern consideration, here's what the dictionary says:

recursion (ri-'kər-zhən) : see recursion

[*insert groan here*] Recursion in ColdFusion is often done by means of a custom tag. The technique itself is very helpful and can provide functionality that would be difficult to achieve if we didn't have it.

A Case for Recursion

I want to take a look at this technique this month, since it's a topic that often frustrates, even frightens, developers. Let's examine an application written for the megacorporation FuseboxersRus. They've decided to create a human resources application to help the many Fuseboxers that work for the company. They want sections related to employee benefits, company information, internal job openings, and training opportunities.

After discussions and prototyping, we determine that the application will have a simple two-tier architecture. The home circuit will be HumanResources. Underneath it will be circuits for Benefits, FuseboxersRus, Openings, and Training (see Figure 1).

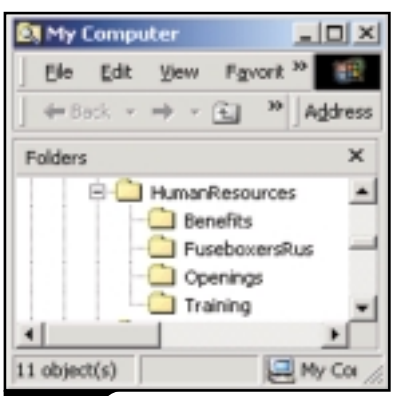


FIGURE 1: Directory structure

FIGURE 2: Main HR page

Further, while each section should have its own home page, the HR department wants an application home page to look like Figure 2. This single home page, generated by the fuseaction HumanResources.main, actually needs information from each circuit. The stock quote will come from the FuseboxersRus circuit, the job openings from Openings, the training opportunities from Training, and the benefits from Benefits.

Using Recursion to Eliminate Duplicate Code

Duplicating code in both home and individual circuits is bad technique. Instead, let's have each circuit respond to fuseactions specifically designed to return information to the home circuit. For the stock quote the Company circuit responds to the fuseaction request, Company.stockQuote. The job openings will be handled by Jobs.random3. The fuseaction, Training.upcoming, will provide the information on training opportunities, and we'll get a list of random benefits by executing the fuseaction Benefits.random5 (see Figure 3).

With cases like this – and there are many such cases in creating real-world applications – the best method is to call the application recursively. In effect, we want to suspend processing of the current fuseaction for a moment while we ask the application to return the information we need to complete the waiting fuseaction.

We can do just that with Fusebox 3. Listing 1 is a snippet from the FBX_Switch.cfm file in the home circuit.

Each <cfmodule> call spawns a separate process, giving us the effect of suspending the original fuseaction while we go out and get the needed information.

The code for the fuseaction is straightforward. The only thing different about it is that it returns information in the caller scope, which is needed so that custom tags can communicate with the pages that called them. If we have code that will be invoked both through a custom tag and normally, this code snippet (placed in the home circuit's FBX_Settings.cfm file) will automatically adjust the scope of the code:

```
<cfif Fusebox.isCustomTag>
 <cfset Scope = "Caller">
<cfelse>
 <cfset Scope = "Variables">
</cfif>
```

FIGURE 3: Main HR page showing fuseactions

> **Duplicating code in both home and individual circuits is bad technique”**

The variable Fusebox.isCustomTag is part of the application programming interface provided by Fusebox 3. Using this technique you can create applications that are more powerful, tidier, and require less code.

[This article is excerpted from the book *Discovering Fusebox*, by Hal Helms and John Quarto-vonTivadar.]

@ HAL@FUSEBOX.ORG

Listing 1

```
<cfcase value="main">
<cfmodule template="#Fusebox.rootPath##self#"
fuseaction="Benefits.random3">
<cfmodule template="#Fusebox.rootPath##self#"
fuseaction="Company.stockQuote">
<cfmodule template="#Fusebox.rootPath##self#"
fuseaction="Training.upcoming">
<cfmodule template="#Fusebox.rootPath##self#"
fuseaction="Jobs.random5">
<cfset XFA.jobInfo = "Jobs.info">
<cfset XFA.benefitInfo = "Benefits.info">
<cfset XFA.trainingInfo = "Training.info">
<cfset XFA.stockInfo = "Company.info">
<cfinclude template="dsp_Main.cfm">
</cfcase>
```

CODE LISTING

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

ABOUT THE AUTHOR

Hal Helms (www.halhelms.com) is a Team Macromedia member who provides both on-site and remote training in ColdFusion and Fusebox. He also publishes a free Occasional Newsletter, available at www.halhelms.com.

Query Custom Tags in ColdFusion

Using QCTs just may make your development life easier

BY
REUBEN
POON

What's the one element almost all ColdFusion-based applications rely on? A database.

Utilizing a database is one of the most fundamental and important aspects of writing ColdFusion applications.

If you're still embedding your queries inside ColdFusion templates, or you don't have a standardized method of accessing a database, this article is for you.

Using Query Custom Tags

So why should you use a query custom tag (QCT)? QCTs provide an abstracted interface to the database, reduce duplicate code inherent with inline queries, ease delegation of work, and allow for individual testing of each query. You can chain them together and build more complex queries out of simpler QCTs. With a set of QCTs you can mimic a database interface layer that your ColdFusion templates can interact with to retrieve or modify data (see Figure 1).

Some of you may be thinking, "This only creates more work for me," or "What benefit does this give me?" Well, the benefits of using an interface (or abstraction) layer are similar to those that would prompt you to use a function or create a class. These benefits include simplifying logic, isolating change, and controlling access to the database.

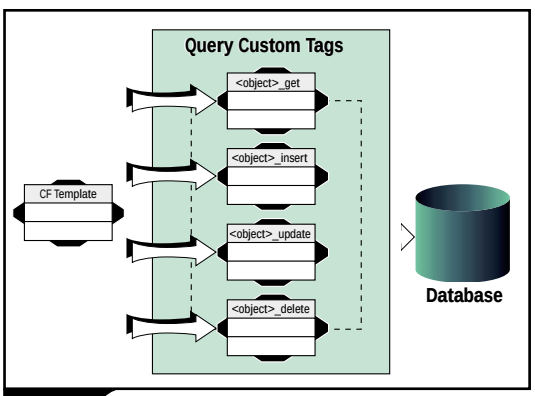


FIGURE 1: Overview

Using a QCT simplifies your code. You can hide complexity you don't need to think about. Instead of seeing many lines that insert some data into the database, all you see is a custom call with the appropriate data. It's clear what's happening based on the name of the custom tag call:

```
<CFMODULE TEMPLATE="ct_customer_insert_qry.cfm">.
```

This custom tag is inserting a customer. When you perform multiple database actions, it's easier to decipher several custom tag calls than a tangled spaghetti-plate of code when reading it later (or when someone else reads it). Code that's easier to understand increases readability and maintainability, which reduces errors.

Enclosing queries within a custom tag isolates change. When you have many inline queries that do the same thing and you need to change anything, you now have to change all those queries. With a QCT you minimize the places needed to change code. Further, changes to the database are hidden. For example, if you rename a column in the database, you can alias that column in your QCT and users of the QCT are abstracted from the details.

With this abstraction inherent with QCTs, you can also control the access to the database. If the person who builds the QCTs is different from the person who builds the display templates, the QCTs can contain business logic that controls access to the database without the display templates even knowing about it. You now also have a central point of control: it's convenient to have only one place to look to make similar changes, such as caching all your data retrieval queries. Or if you'd like to implement a draft/publish system for your data, you can add that logic to your QCTs.

Even without all the benefits of abstraction that QCTs bring, one of the biggest pluses is code reuse. When you write inline queries – those that are embedded directly in the ColdFusion template – you rewrite that query for every page. By putting all your queries into custom tags, you can reuse a query anywhere in your application. Many times these inline queries differ only slightly from page to page. One query might return all rows in a table, for example, whereas another returns only one. Both queries can be rolled into one QCT that performs a simple "get" of that table's data.

With QCTs the work of building an application can be easily delegated among developers. One person can work on building the display templates while another works on building the database interface through the QCT. Delegation applies not only to developers, but also to the actual code. You create a transaction QCT that collects only data and delegates work by calling other QCTs.

Because the queries are part of a custom tag, unit testing is much easier through this defined interface and you can test every query you write. Now you can write a test harness for each query and run through tests to make sure it's doing what you think it is. Of course, even if all your unit tests pass, you can't guarantee that your entire application works; however, if even *one* unit test fails, you'll know for a fact that your application is broken.

You may still be asking why use a custom tag. Wouldn't you be able to gain these benefits by including the query with the <CFINCLUDE> tag?

The answer is No. Includes don't actually hide the details of what goes on inside that file. When you create a custom tag, everything that goes on inside the tag is hidden to the caller. Any variables created aren't accessible and you don't have the side effects inherent when variables are shared.

Custom tags provide a clear and consistent interface for queries. Rather than relying on implicit variable passing (like setting an ATTRIBUTE-scoped variable), or stating in the documentation that such and such a variable must exist and the query will be returned in this variable name, custom tags allow precise control of the input and output of any information necessary for the query. When you see a QCT call, you know exactly what's being passed to the custom tag and, to a lesser extent, what's returned.

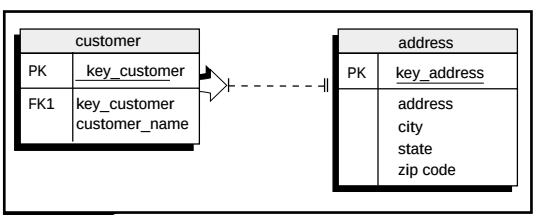


FIGURE 2: Database schema

Users of the QCT can also specify how they'd like the information returned (if necessary). This way the QCT won't randomly clobber a variable because the hard-coded variable name to return was already used for something else. With a custom tag you can perform complex and conditional logic based on the input information. This also allows for multipurpose queries – returning one record when a key is passed or returning all records when no key is passed.

Implementing Query Custom Tags

Okay, QCTs sound like a good thing to you. How do you implement them? Building a QCT consists of four steps:

1. Building the custom tag itself
2. Accessing the information passed to the custom tag
3. Setting up the query
4. Returning the results to the caller (if necessary)

The first step consists of building the custom tag. One thing all our custom tags include is a descriptive header. This consists of basic information such as the custom tag name, author, creation and modified dates, description of the custom tag, and a detailed listing of all the attributes (see Listing 1). Another thing is to make sure the code is executed only once if both a beginning and an ending tag are used:

```
<CFMODULE TEMPLATE="ct_cus-
tomer_get_qry.cfm"></CFMODULE>
```

This is done by wrapping all of our code within this IF statement:

```
<CFIF ThisTag.ExecutionMode EQ
"start"></CFIF>
```

One thing that makes life easier is setting up a template for a generic custom tag (we even have templates for generic QCTs). Now we can start setting up the code to localize and check for required custom tag attributes.

Our second step is to pass information into the custom tag. We do this by using custom tag attributes. For a custom tag that retrieves records, we expose an optional attribute for the key of the table. If the key is passed, we use it and return only the one record for that key. If no key is passed, we return all records. Extending this technique, you can create a single QCT that performs many actions (this is especially useful for queries that search the database). We now scope the attribute to our local (VARIABLES) scope (see Listing 2). Another attribute that we pass to QCTs is "query_name" (see Listing 3). This allows QCT callers to specify what variable name they'd like their query results placed in. Many developers set up global constants for data such as the data source. Our QCTs allow the application data source to be used or a new data source to be passed into the custom tag (see Listing 4).

Now that we have our required information, we can move to the third step and set up the query (see Listing 5). Since this query will retrieve data, we check to see if the key was passed in. If it was, we select that one row. Otherwise we select all the rows. Remember to use the Val function to make sure that malicious or malformed data can't harm the database. This way a database command sent to the QCT posing as a key value won't be performed.

We use the query_name attribute that the QCT callers specify and make this the name of our query. By using the CALLER scope, we make this query available to the caller's scope. The entire QCT can be seen in Listing 6.

Uses

One way to utilize QCTs is to chain them together in a cascading hierarchy. Creating a QCT that delegates functionality to several smaller QCTs results in a more modular, object-based view of the database. For example, suppose you're storing some customer information represented in the simple database schema depicted in Figure 2. When adding a new customer to the database, you must enter information into two tables. We'll chain two insert QCTs to a transaction QCT, as shown in Figure 3. Users now need to call only one QCT, which clarifies and simplifies the process for the caller.

```
<CFMODULE TEMPLATE="ct_cus-
tomer_object_insert_qry.cfm"
 CUSTOMER_NAME="#VARIABLES.cus-
tomer_name#"
 ADDRESS="#VARIABLES.address#"
 CITY="#VARIABLES.city#"
 STATE="#VARIABLES.state#"
 ZIPCODE="#VARIABLES.zip-
code#"></CFMODULE>.
```

Using Query Custom Tags with Fusebox

For those of you who use the Fusebox methodology for development, query custom tags are similar to the query and action fuses. The logic driving how QCTs came about probably differs from the flavor that Fusebox brings, but the resulting techniques are similar.

Two primary ways of thinking separate QCTs from the Fusebox style. One is that QCTs were designed to act as an abstraction layer between ColdFusion code and the database. The second is that QCTs were designed to be directly chained together to build up complex relational database schemas based on individual insert, update, and delete QCTs.

That said, I believe that QCTs can be integrated into Fusebox applications. The first distinction between QCTs and Fusebox really lies in just how you think about what you're doing. The second distinction is implementation based, but can be implemented in Fusebox by simply calling a different Fuse or, in Fusebox 3, by nesting circuits.

So if you're a Fusebox developer, try thinking about the fuses that interact with the database a little differently and see whether QCTs will benefit your development.

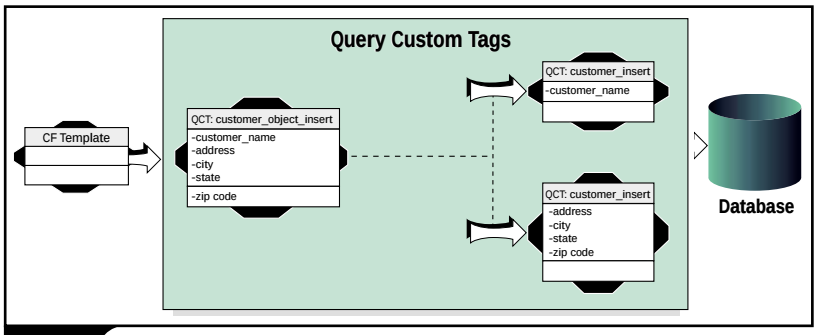


FIGURE 3: Cascading insert

First we create two QCTs that actually do the work of inserting data into the database (see Listings 7, 8). Notice how each insertion QCT returns the key of the inserted row. This makes it easy to chain QCTs together by grabbing inserted keys and passing them to the next QCT in the cascade. We now create a transaction QCT that will collect the required information, organize all the data, and delegate the work to other QCTs (see Listing 9). The transaction QCT grabs the inserted address key and passes it to the customer insert QCT. Note that the data collection QCT both locks and starts a database transaction to wrap both insert QCT calls.

For more complex table structures, the transaction QCT may need to collect data by using child tags.

Conclusion

Now that you know how query custom tags can benefit you and you understand how to use them, give it a shot. By writing QCTs and chaining them together to cascade actions, you'll abstract the interaction with the database, promote code reuse, ease delegation of work, and allow more granular testing of code. After you've used QCTs for a while, you'll be amazed by how much easier and simpler to write – and to debug – your code becomes.

ABOUT THE AUTHOR

Reuben Poon has developed Web applications and sites for more than four years and currently works at PINT, Inc. (www.pint.com). He is ColdFusion certified and has helped write books such as JavaScript: The Complete Reference.

@ REUBENP@PACBELL.NET.

Listing 1

```
<!--- ================================
TAG NAME:    ct_customer_get_qry
AUTHOR:      PINT
CODER:       Reuben Poon
CREATED:     02.24.02
MODIFIED:    02.24.02

FUNCTION:    Returns a query containing customer information.

NOTES:       Passing in a customer key will narrow the results
             of the query.

ATTRIBUTES:
             NAME:      customer_key
             VALUE:     (int)
             REQUIRED: NO
             DEFAULT:   not used to narrow query results

             NAME:      query_name
             VALUE:     (string)
             REQUIRED: no
             DEFAULT:   "qCUSTOMER"

             NAME:      APPLICATION.datasource.name
                        OR datasource_name
             VALUE:     (string)
             REQUIRED: yes

USAGE:
             <CFMODULE
               TEMPLATE="ct_customer_get_qry.cfm"
               QUERY_NAME="qCUSTOMER">
========================================== --->
```

Listing 2

```
<!--- CUSTOMER_KEY --->
<CFIF IsDefined( "ATTRIBUTES.customer_key" )>
  <CFSET VARIABLES.is_customer_key_defined =
    true>
  <CFSET VARIABLES.customer_key =
    ATTRIBUTES.customer_key>
<CFELSE>
  <CFSET VARIABLES.is_customer_key_defined =
    false>
  <CFSET VARIABLES.customer_key = 0>
</CFIF>
```

Listing 3

```
 <!--- QUERY_NAME --->
<CFIF IsDefined( "ATTRIBUTES.query_name" )>
  <CFSET VARIABLES.query_name =
    ATTRIBUTES.query_name>
<CFELSE>
  <CFSET VARIABLES.query_name =
    "qCUSTOMER">
</CFIF>
```

Listing 4

```
<!--- APPLICATION.DATASOURCE.NAME --->
<CFLOCK TIMEOUT="40" SCOPE="APPLICATION"
  TYPE="READONLY">
<CFIF IsDefined( "ATTRIBUTES.datasource_name" )>
  <CFSET VARIABLES.datasource_name =
    ATTRIBUTES.datasource_name>
<CFELSEIF
  IsDefined( "APPLICATION.datasource.name" )>
  <CFSET VARIABLES.datasource_name =
    APPLICATION.datasource_name>
<CFELSE>
  <CFTHROW MESSAGE="The Attribute DATASOURCE_NAME
    or Application variable
 APPLICATION.DATASOURCE_NAME
 must be defined.">
</CFIF>
</CFLOCK>
```

Listing 5

```
SELECT
  customer_name , address , city , state , zipcode
FROM
  customer , address
WHERE
  customer.key_address = address.key_address AND
  <CFIF VARIABLES.is_customer_key_defined>
    customer_key =
   #Val( VARIABLES.customer_key )# AND
  </CFIF>
  1 = 1
```

Listing 6

```
<!--- Suppress lots of white space --->
<CFSETTING ENABLECFOUTPUTONLY="YES">

<!--- ==========================================
TAG NAME:    ct_customer_get_qry
AUTHOR:      PINT
CODER:       Reuben Poon
CREATED:     02.24.02
MODIFIED:    02.24.02

FUNCTION:    Returns a query containing customer information.

NOTES:       Passing in a customer key will narrow the results
             of the query.

ATTRIBUTES:
             NAME:      customer_key
```

```
            VALUE:    (int)
            REQUIRED: NO
            DEFAULT:  not used to narrow query results

            NAME:     query_name
            VALUE:    (string)
            REQUIRED: no
            DEFAULT:  "qCUSTOMER"

            NAME:     APPLICATION.datasource.name
                      OR datasource_name
            VALUE:    (string)
            REQUIRED: yes

USAGE:
            <CFMODULE
              TEMPLATE="ct_customer_get_qry.cfm"
              QUERY_NAME="qCUSTOMER">
============================================== --->

<CFIF ThisTag.ExecutionMode EQ "start">
  <!--- ----------------------------------------
  BEGIN Localize Variables
  ---------------------------------------- --->

  <!--- CUSTOMER_KEY --->
  <CFIF IsDefined( "ATTRIBUTES.customer_key" )>
    <CFSET VARIABLES.is_customer_key_defined =
    true>
    <CFSET VARIABLES.customer_key =
    ATTRIBUTES.customer_key>
  <CFELSE>
    <CFSET VARIABLES.is_customer_key_defined =
    false>
    <CFSET VARIABLES.customer_key = 0>
  </CFIF>

  <!--- QUERY_NAME --->
  <CFIF IsDefined( "ATTRIBUTES.query_name" )>
    <CFSET VARIABLES.query_name =
    ATTRIBUTES.query_name>
  <CFELSE>
    <CFSET VARIABLES.query_name = "qCUSTOMER">
  </CFIF>

  <!--- BEGIN Standard Query Variables --->
  <CFSCRIPT>
  VARIABLES.datasource = StructNew();
  </CFSCRIPT>

  <!--- APPLICATION.DATASOURCE.NAME --->
  <CFLOCK TIMEOUT="40" SCOPE="APPLICATION"
    TYPE="READONLY">
  <CFIF IsDefined( "ATTRIBUTES.datasource_name" )>
    <CFSET VARIABLES.datasource.name =
    ATTRIBUTES.datasource_name>
  <CFELSEIF
    IsDefined( "APPLICATION.datasource.name" )>
    <CFSET VARIABLES.datasource.name =
    APPLICATION.datasource.name>
  <CFELSE>
    <CFTHROW MESSAGE="The Attribute DATASOURCE_NAME
    or Application variable
    APPLICATION.DATASOURCE.NAME must be defined.">
  </CFIF>
  </CFLOCK>
  <!--- END Standard Query Variables --->

  <!--- ----------------------------------------
  END Localize Variables
  ---------------------------------------- --->

  <CFQUERY NAME="CALLER.#VARIABLES.query_name#"
    DATASOURCE="#VARIABLES.datasource.name#">
    SELECT
      customer_name ,
      address , city, state, zipcode
    FROM
      customer , address
    WHERE
      customer.key_address = address.key_address AND
      <CFIF VARIABLES.is_customer_key_defined>
        customer_key =
    #Val( VARIABLES.customer_key )# AND
      </CFIF>
      1 = 1

    <!---
    NOTE: Normally a view to simplify this query.
    --->
  </CFQUERY>

</CFIF>

<!--- UnSuppress lots of white space --->
<CFSETTING ENABLECFOUTPUTONLY="NO">
```

Listing 7

```
<!--- INSERTED_ADDRESS_KEY_VARIABLE --->
<CFIF IsDefined
  (
  "ATTRIBUTES.inserted_address_key_variable"
  )>
  <CFSET VARIABLES.inserted_address_key_variable =
  ATTRIBUTES.inserted_address_key_variable>
<CFELSE>
  <CFSET VARIABLES.inserted_address_key_variable =
  "VARIABLES.inserted_address_key">
</CFIF>

<CFQUERY NAME="address_insert"
  DATASOURCE="#VARIABLES.datasource.name#">
```

```
  <!---
  This query is MS SQL specific.  Oracle
  implementations will be slightly different and may
  require two CFQUERY calls
  --->
  SET NoCount On
  INSERT INTO
    address
    (
    address ,
    city ,
    state ,
    zipcode
    )
  VALUES
    (
    '#VARIABLES.address#' ,
    '#VARIABLES.city#' ,
    '#VARIABLES.state#' ,
    '#VARIABLES.zipcode#'
    )
  Set NoCount Off
  SELECT inserted_address_key = scope_identity()
</CFQUERY>
<CFSCRIPT>
"CALLER.#VARIABLES.inserted_address_key#" =
  address_insert.inserted_address_key;
</CFSCRIPT>
```

Listing 8

```
<!--- INSERTED_CUSTOMER_KEY_VARIABLE --->
<CFIF IsDefined
  (
  "ATTRIBUTES.inserted_customer_key_variable"
  )>
  <CFSET VARIABLES.inserted_customer_key_variable =
    ATTRIBUTES.inserted_customer_key_variable>
<CFELSE>
  <CFSET VARIABLES.inserted_customer_key_variable =
    "VARIABLES.inserted_customer_key">
</CFIF>
<CFQUERY NAME="customer_insert"
  DATASOURCE="#VARIABLES.datasource.name#">
  <!---
  This query is MS SQL specific.  Oracle
  implementations will be slightly different and may
  require two CFQUERY calls
  --->
  SET NoCount On
  INSERT INTO
    customer
    (
    key_address ,
    customer_name
    )
  VALUES
    (
    #VARIABLES.address_key# ,
    '#VARIABLES.customer_name#'
    )
  Set NoCount Off
  SELECT inserted_customer_key = scope_identity()
</CFQUERY>
<CFSCRIPT>
"CALLER.#VARIABLES.inserted_customer_key#" =
  customer_insert.inserted_customer_key;
</CFSCRIPT>
```

Listing 9

```
<!--- INSERTED_CUSTOMER_KEY_VARIABLE --->
<CFIF IsDefined
  (
  "ATTRIBUTES.inserted_customer_key_variable"
  )>
  <CFSET VARIABLES.inserted_customer_key_variable =
    ATTRIBUTES.inserted_customer_key_variable>
<CFELSE>
  <CFSET VARIABLES.inserted_customer_key_variable =
    "VARIABLES.inserted_customer_key">
</CFIF>
<CFLOCK TIMEOUT="500">
<CFTRANSACTION>
<CFMODULE TEMPLATE="ct_address_insert_qry.cfm"
  ADDRESS="#VARIABLES.address#"
  CITY="#VARIABLES.city#"
  STATE="#VARIABLES.state#"
  ZIPCODE="#VARIABLES.zipcode#"
  INSERTED_ADDRESS_KEY_VARIABLE=
    "VARIABLES.inserted_address_key">
<CFMODULE TEMPLATE="ct_customer_insert_qry.cfm"
  ADDRESS_KEY="#VARIABLES.inserted_address_key#"
  CUSTOMER_NAME="#VARIABLES.customer_name#"
  INSERTED_CUSTOMER_KEY_VARIABLE=
    "VARIABLES.inserted_customer_key">

</CFTRANSACTION>
</CFLOCK>

<CFSCRIPT>
"CALLER.#VARIABLES.inserted_customer_key#" =
  VARIABLES.inserted_customer_key;
</CFSCRIPT>
```

CODE LISTING

The code listing for this article is also located at

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

By Christian Schneider

Developing a WAP Based E-mail Interface

An introduction to WAP/WML possibilities using ColdFusion

This article is about developing an application for wireless devices with WAP support. For this I've chosen to implement a WAP-based e-mail client.

On the Web, using HTML, this is quickly done with ColdFusion. This should also be true for WAP, I thought. Whether or not it turns out to be true…well, just read on.

Okay, it did turn out to be easy, since ***CFDJ*** has published several articles about the design and pitfalls of WAP development in the last few months. I strongly suggest that you also take a look at the two-part article "Developing Wireless Apps with ColdFusion" by Charlie Arehart (***CFDJ***, Vol. 2, issues 8, 9).

To start developing a real-world WAP application, you need to know the basics of WAP and WML, such as knowing that a WML "page" is called a "card," and a "deck" is a collection of "cards" being transferred all at once. You'll also need intermediate skills in ColdFusion, especially an understanding of the <CFPOP> and <CFMAIL> tags we're going to use in this application. But if you're a novice, don't be discouraged if these terms mean nothing to you. You'll get to know them all during the course of this article.

I recommend that you download and install the mobile phone simulator UP.SDK from www.openwave.com (formerly phone.com) on your PC (see the referenced Arehart article for more details) so you can test this application without a WAP-capable device.

Take a look at Figure 1, which diagrams the application's flow. To better understand the diagram, I've provided a number of screenshots (see Figures 2–6). As I explain the flow in the next

paragraphs, you can directly link the screenshots to the different states in the UML diagram.

As you can see in Figure 1, after providing the login information, the mobile user can either check for mail or compose new mail (see also Figures 2 and 3).

When checking for new mail, the CF server uses the provided login information (stored into the session) to access the user's mail account via <CFPOP> and displays the result as a listing of all mail headers (see Figure 4). After selecting a message, the details are shown in a separate card (see Figure 5), where the user can decide either to open the mail or to delete it from the server. After opening an e-mail, the user can read it and then either return to the inbox for more mail, delete it from the server, or send a reply (see Figure 6).

When you look at the source code on the Web (at www.sys-con.com/coldfusion/sourcec.cfm), you'll see that I first coded this application in HTML for the Web browser to be sure of its functionality, then modified the HTML prototype into a WAP/WML solution for wireless devices. I originally thought I'd better not do this since some concepts of application flow that work well in HTML would certainly not work in WML and vice versa, but after I finished the prototype, the change to WML was easier than I expected.

Naturally, this kind of prototyping only works when you're thinking of WML all the time and concentrating on the flow of your mobile phone application. But with this in mind, you can quickly ensure that your application's features are working, since testing and debugging in an HTML-based Web browser is much easier than with WML on WAP devices (you all know how to debug in the HTML/Web browser environment, but I'm sure not all of you have a good way to track errors using a mobile phone).

Once the HTML prototype was completed, I felt a little sad about dropping it after having ported it on WML, so I had the templates check for WAP devices or HTML-capable browsers to decide which version to send to the client. This was done with just a few lines of simple code:

```
<!--- Test of WAP-Devices (Auto-Detect) --->
<cfif CGI.HTTP_ACCEPT CONTAINS "text/vnd.wap.wml">
      <cfset isWap = true>
<cfelse>
      <cfset isWap = false>
</cfif>
```

Here we check the CGI variable CGI.HTTP_ACCEPT for the MIME types the client browser accepts. If the client accepts "text/vnd.wap.wml", you can be sure it's a mobile WAP device; if it doesn't, most likely it's the usual HTML-capable Web browser. Having placed this simple check inside the Application.cfm file (see Listing 1), we can tell whether the user is "surfing" with a mobile WAP phone or a standard Web browser. As you'll see when you look at the other code listings, most pages check for this flag at the beginning and serve different content depending on whether it's WML or HTML. For more complex applications that behave differently when accessed via WAP than they do via a conventional Web browser, you might think of writing two sets of templates (at least for presenting the user interface) instead of just checking within each template. For simpler applications, such as the example in this article, the first method – deciding within the templates – is just fine.

Design Considerations for WML

A primary concept of WML design you need to understand is that WAP devices have a very small screen; consequently, a WML "deck" is divided up into a collection of "cards" that can contain both the user interface and the content. So instead of presenting a large table listing all new mail on the server with all the details (sender's address, date, etc.), which would be good for HTML-capable Web browsers, it's better to split this information into several cards within one deck for WML.

The benefit of designing a card that lists only the mail's subjects, linked to the appropriate card holding the headers (sender, date, etc.) within the same deck, is a more comfortable user interface without any performance loss since all the headers are transferred within the same deck and therefore available locally without any server round-trips. By hiding detailed information on subsequent cards and presenting something akin to an index card on top, you can design a good WML deck.

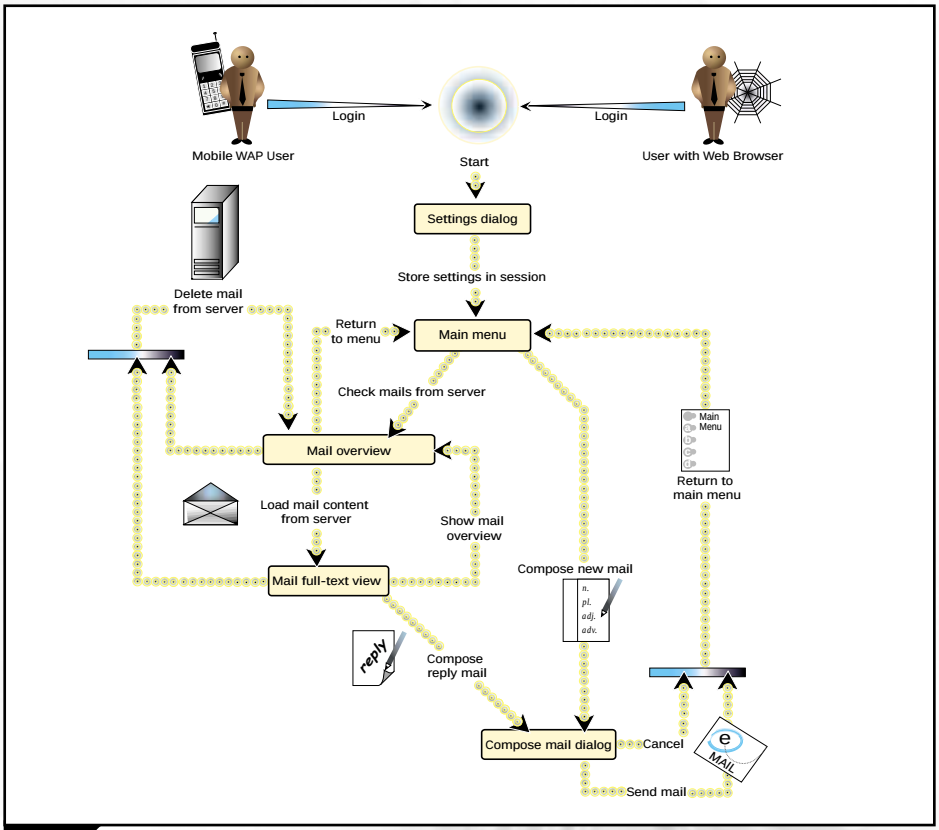


FIGURE 1 State diagram of the application

On the other hand, putting too much detail in one deck exceeds the maximum (which is about 2 kilobytes per compiled deck sent from the WAP gateway). This is why I put just the header details in the deck, not the actual mail content, which could be quite a lot of text. After viewing the header details, a user can choose whether to fetch the content or continue browsing other e-mail.

On the subject of design considerations, mobile phones of course have no PC keyboard, although I sometimes wish they did – I hate typing in long text using the cumbersome keypad. Okay, this is a bad example, I thought, since e-mail often involves a lot of typing (especially when composing new mail) – but that's really up to the user: a mobile phone can certainly check new mail, and sometimes handle short text. But when using this example I strongly advise you not to ask for the mail account data with every login. Instead, provide your users with a database-based storage of their account data. As most of you (including me when I coded the example app) change the mail account settings during testing, asking the user to type in all settings at the start is okay. I also did this to keep things as simple as possible, and tried to focus on the WAP-related issues, not on storing login data into a database table (which is definitely not new to you).

Session Management on a Mobile Phone

Before going into the details of the code, I'd like to talk a bit about sessions. In general, ColdFusion provides the developer with a nice Web application framework including session management. If you haven't much experience in this area, session management is handled by the CF server via the use of cookies. Each user (browser) gets a session cookie set with a unique ID (CFID and CFTOKEN) that's sent back to the server with every request within the Web application. Alternatively, when cookies aren't supported on the client's browser, you can still keep the session management aware of its clients by passing the unique ID value of the client through every URL link your Web application has. This means you'll then have to append #SESSION.UrlToken# at every URL-like link, form target, or client-based JavaScript redirection. That way the CF server keeps informed about its clients even when no cookies are supported.

All this is true if you're dealing with the traditional Web and major browsers. But in terms of microbrowsers running on a mobile device that's not directly connected to your Web server (on mine it's connected through a WAP gateway server of the network supplier), things could be different. Fortunately, the WAP design group (www.wapforum.org), which consists of all major players and companies in the WAP world, has thought about this aspect too, and most WAP gateways keep track of cookies on behalf of the mobile devices and send them back to the server on each subsequent request (as normal browsers would). But to be on the safe side, I suggest you append the #SESSION.UrlToken# to each link in WML (you'll see this often throughout the listings).

Talking about links, WML is strictly XML; thus you'll have to watch out for all special characters, including the "&" sign we often use when passing parameters through a URL link. To put it simply, what would normally look like http://www.domain.com/page.cfm?param1=value1&param2=value2&param3=value3 must be written in WML, that is, http://www.domain.com/page.cfm?param1=value1&param2=value2&param3=value3. To make it easy to keep track of all XML-relevant special characters, you could also use the following line to output the link via the CF4.5 XmlFormat() function:

```
#XmlFormat("http://www.domain.com/
page.cfm?param1=value1&param2=value2&
param3=value3")#
```

Getting Data Back from the WAP Device to the Application

Before I start describing the source code, there's one point left to discuss: how to pass input data back from the mobile device to the application. In traditional Web programming this is done through forms, with form fields passing values either as GET or POST requests to the action page (the latter is done more often, I suppose), as well as by using URL parameters appended to the action page's URL as the query string (the part after the "?") that emulates GET requests.

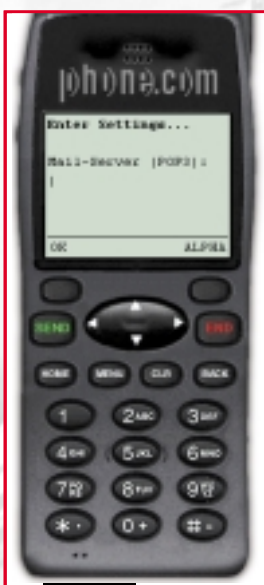


FIGURE 2: Enter the mail server settings

FIGURE 3: Main menu of application

In WML you can submit the input values either as a POST request (using <postfield>s) or add them to a URL link (the equivalent of the GET method). As some gateways don't let <postfield>s pass values correctly (note that the simulators handle them correctly when in HTTP direct mode with no WAP gateway involved), I suggest you use the URL "get" variant, at least for now. A more detailed description of this can be found in Part 2 of Arehart's article, referenced earlier.

One nice feature of WML when dealing with form fields is that each value entered into a deck is directly available (in all cards within that deck) as a client-side WML variable.

```
<wml>
  <card id="card1">
    <do type="accept"><go
      href="#card2"/></do>
    <p>
      Password:
      <input name="lastName"
        type="text"/>
    </p>
  </card>
  <card id="card2">
    <p>
      You entered: $lastName
    </p>
  </card>
</wml>
```

In this snippet from a WML file the second card directly repeats what the user has entered on the first card's input field. So all input data is directly available in client-side WML variables for further use within the deck. We use these variables to attach the form field's input values as URL parameters to a link to the action pages.

Looking at the Code

Some of you may think this is too much information, coming as it does before the details of the source code. The advantage of this rather long introduction is that now you're aware of the more interesting aspects of WML development as you face it for the first time. Consequently, understanding the source code should be quite easy. So let's begin.

This application (see Figure 1 for the state diagram) consists of 11 listings that can be downloaded from www.sys-con.com/coldfusion/sourcec.cfm.

1. **application.cfm:** Initializes session management
2. **index.cfm:** Asks user for e-mail account settings
3. **initAccount.cfm:** Stores account settings in user's session
4. **mainMenu.cfm:** Wrapper including menu.cfm
5. **menu.cfm:** Snippet that's included wherever menu should be presented
6. **checkMail.cfm:** Checks user's mail server for e-mail and lists headers for overview
7. **openMail.cfm:** Loads and opens full text of e-mail (eventually parceled out to multiple decks if too large)

FIGURE 4: Listing of new mails

FIGURE 5: Detail view of a mail's header

FIGURE 6: Mail body of an opened mail

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
vp, business development
Grisha Davida grisha@sys-con.com
chief operating officer
Mark Harabedian mark@sys-con.com

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

editorial
executive editor
M'lou Pinkham mpinkham@sys-con.com
editor
Nancy Valentine nancy@sys-con.com
managing editor
Cheryl Van Sise cheryl@sys-con.com
associate editors
Jamie Matusow jamie@sys-con.com
Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com

production
vp, production & design
Jim Morgan jim@sys-con.com
lead designer
Cathryn Burak cathyb@sys-con.com
art director
Alex Botero alex@sys-con.com
associate art directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Aarathi Venkataraman aarathi@sys-con.com
assistant art director
Tami Beatty tami@sys-con.com

sys-con events
vice president, events
Cathy Walters cathyw@sys-con.com
conference manager
Michael Lynch mike@sys-con.com
sales executives, exhibits
Richard Anderson richard@sys-con.com
Michael Pesick michael@sys-con.com

customer relations/JDJ store
manager, customer relations /JDJ store
Anthony D. Spitzer tony@sys-con.com

web services
web designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
content editor
Lin Goetz lin@sys-con.com

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

8. **composeMail.cfm:** Presents user interface for composing new e-mail or reply mail
9. **sendMail.cfm:** Sends composed mail
10. **deleteMail.cfm:** Deletes specified mail from user's mail server
11. **error.cfm:** Invoked when error was caught with <CFTRY> and <CFCATCH>

Before describing each file in detail, I'd like to introduce the header that's necessary when using WML with CF tags:

```
<cfsetting showdebugoutput="No">
<cfcontent type="text/vnd.wap.wml"
 reset="Yes"><?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD
 WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1">
     <p>Hello World</p>
  </card>
</wml>
```

The <cfsetting> tag disables any debug output on the page to make sure your WML doesn't become invalid XML when debug output is sent to it at the end of the request. Next you'll see <cfcontent type="text/vnd.wap.wml" reset="Yes">, which is for setting the MIME type of the page to "WML", telling the client, "Yes, I'm really WML." The same is true with the XML and doctype declaration, which is a must for every WML page.

Now let's take a more detailed look at the source files:

***application.cfm* (Listing 1)**

What we're doing here is enabling session management and checking for session timeouts and for a WAP/WML or HTML client (see the first snippet above).

***index.cfm* (Listing 2)**

This file (and all subsequent files) checks the REQUEST.isWap flag to determine what to send to the client, WML or HTML. As this file asks for a user's e-mail account settings, we have one card with many <input> fields for entering the POP3 and SMTP servers as well as the username and password for POP3 access. The more interesting part is where we set an action on one of the phone's buttons (here, the "accept" button, which is often directly beneath the display). You'll notice that I assigned a URL to that button, which is output from the CF variable #tmpTarget#. This variable is set up inside the <cfscript> block to become a URL string, sending every <input> field's value as a URL parameter by way of client-side WML variables. The content of #tmpTarget# looks like this (with the CFID and CFTOKEN replaced by the unique values of the current session):

> **"To start developing a real-world WAP application, you need to know the basics of WAP and WML. You'll also need intermediate skills in ColdFusion"**

```
"initAccount.cfm?CFID=1&CFT0KEN
=95042334& initPop3=$initPop3&
initPop3Port=$initPop3Port&
....."
```

Finally, after all "&"s have been escaped by XmlFormat(), the link is written into the <go> WML tag for assigning links to actions. That way the file initAccount.cfm gets all settings as URL parameters.

***initAccount.cfm* (Listing 3)**

This file is passed in the account settings either as URL or FORM variables originating from either the WML or the HTML interface. After being captured by this template, they're stored in the session scope for further use. Finally, the menu.cfm file is included to show the menu.

***mainMenu.cfm* (Listing 4)**

This file is nothing more than a wrapper including menu.cfm, which is convenient when a link should display only the menu content and nothing more. Then you simply set the link to mainMenu.cfm.

***menu.cfm* (Listing 5)**

This file is the real menu, which is included wherever a menu is needed on a page. It contains links to checkMail.cfm, compose Mail.cfm, and index.cfm.

***checkMail.cfm* (Listing 6)**

This file is more interesting. Here all headers are fetched from the user's mail server using <CFPOP> and stored in the ColdFusion query variable "qryMailHeaders". In WML a <select> list is populated with the e-mail subjects taken from the query. Each <option> tag jumps to a detail card containing all mail headers using the onpick="#cardName" attribute. Note that I escape all strings with Xml-Format() to avoid any conflicts when an e-mail contains special characters. After the

card holding the <select> list has been written, a card displaying the mail's headers (such as sender, date, and subject) is generated for each mail. Each detail card also has links to open the e-mail, delete it from the server, or return to the main menu.

***openMail.cfm* (Listing 7)**

As the name suggests, this file opens the mail, using <CFPOP> to fetch the entire e-mail, including the body. It won't fetch attachments since you can't use them on mobile phones – at least not yet. Next, the body of the e-mail is cropped after about a thousand characters and split over multiple decks, if necessary, to avoid a deck that exceeds the maximum size. Finally, the "back to inbox," reply, and delete links are written. While the delete link just calls deleteMail.cfm with the mail's message ID as a URL parameter, the reply link looks more interesting since it passes the sender and subject as escaped URL parameters to composeMail.cfm.

***composeMail.cfm* (Listing 8)**

This file acts as the user interface for writing new mail or replies to mail. This double feature is possible through optional FORM/URL parameters holding the sender as well as the subject when acting as a reply mailer. The sender and the subject are populated into the corresponding <input> fields. The link to sendMail.cfm is written the same as it is in index.cfm (see Listing 2).

***sendMail.cfm* (Listing 9)**

This file simply takes all parameters necessary for sending e-mail (such as receiver, subject, etc.) either as FORM parameters when submitted from HTML or as URL parameters when originating from a WML deck. After the mail has been sent using <CFMAIL>, the main menu is shown.

***deleteMail.cfm* (Listing 10)**

This file works much like openMail.cfm except that instead of loading the mail using <CFPOP>, it deletes it from the server using <CFPOP action="DELETE" ...>.

***error.cfm* (Listing 11)**

This file is invoked within a <CFCATCH> block used in critical sections where the mail server is involved. Whenever an error occurs with the mail server (most often a typo in the account settings dialog at the mail server's address), this template is loaded, trying to present the error message. The file could be extended to serve as a general error-handling template for the <CFERROR> tag with minimal effort, resulting in application-wide error handling.

Summary

This article is intended to be an introduction to WAP/WML development with ColdFusion based on the example of an e-mail service application. My goal was to remove all the obstacles and present you with a nice but simple example app. Advanced WML aspects – for example, the WMLScript client-side scripting language (like JavaScript for the traditional Web) – is out of this article's scope. To explore these advanced techniques (after having mastered the basics), I recommend you take a look at some of the books available on WAP/WML development.

About the Author

Christian Schneider, a certified ColdFusion and Web site developer, has more than four years of intensive experience developing CF-based intranet applications for banks and logistics corporations.

@ MAIL@CHRISTIAN-SCHNEIDER.DE

WebServices JOURNAL .NET J2EE XML

XML JOURNAL

THE FIRST & ONLY WEB SERVICES RESOURCE CD!

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN 400 EXCLUSIVE WEB SERVICES & XML ARTICLES

EDITED BY SEAN RHODY

$119 CD VALUE FROM WEB SERVICES JOURNAL

web services EDGE conference & expo

$100 coupon Inside

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

CD is edited by well-known WSJ Editor-in-Chief Sean Rhody and organized into more than 40 chapters containing more than 400 exclusive WSJ & XML-J articles.

Easy-to-navigate HTML format!

Bonus:

Full .PDF versions of every WSJ & XML-J published since the first issue

- 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 Script
- XML Industry Insider
- <e-BizML>
- XML & Business
- XML Demystified
- XML & E-Commerce
- XML Middleware
- XML Modeling
- CORBA & XML
- Data Transition
- XML @ Work
- XML & Databases
- Electronic Data Interchange
- Ubiquitous Computing
- Information Management
- Objects & XML
- XML Pros & Cons
- Java Servlets
- XML Filter
- UML Integration
- WSDL
- Beginning Web Services
- Web Services Tips & Techniques
- Frameworks
- Management
- Security
- UDDI
- .NET

3 YEARS
25 ISSUES
400 ARTICLES
ONE CD

Now Shipping

$79

ONLINE
ORDER AT
JDJSTORE.COM
SAVE
$40

Using JSP Custom Tags in CFMX

What they are, how and why you might use them, and where to find them

BY CHARLES AREHART

CFMX, the new release of ColdFusion, will allow us to use JSP custom tags within our ColdFusion programs. Some may say, "So what?"

In this article I hope to show you why it's definitely worth looking into this new possibility.

At the time of this writing CFMX, previously known as *Neo*, is still in beta and the nondisclosure agreement prevents people writing about its features, but one of the few things that have been public knowledge since last year's DevCon is the ability to leverage JSP custom tags.

I've been given the OK by Macromedia (indeed the encouragement) to show this stuff to you. But be aware that what I'm showing is how things work as of beta 3. It's possible that things may change prior to the production release, but again, this functionality has been shown in its current form for nearly a year.

JSP Custom Tags Are Coming

In CFMX it may soon not be unusual to see something like the following in some ColdFusion code:

```
<cal:calendar month="3"
year="2002" />
```

This is an example of calling a JSP custom tag. This one (which really does exist) will generate an HTML calendar for March 2002, as shown in Figure 1. Pretty nifty. We don't have such a tag in ColdFusion, and it's not that trivial to build calendars, so to be able to generate one so easily with a single tag is powerful.

Some will quickly point out that it's certainly possible that someone could write (and indeed has surely written) a similar sort of ColdFusion custom tag and make it available at the Macromedia Developer's Exchange (at http://devex.macromedia.com/developer/gallery/). Custom tags have been around in ColdFusion for a long time, and they're certainly an underused resource for many.

But this is a JSP custom tag. Don't let that throw you. Does that example above look really all that different to you as a CF developer? Sure, the syntax is a little different from a normal CF custom tag call. Where perhaps we may have used <cf_calendar>, if a tag of that name existed, this one uses <cal:calendar>. That's curious and will be explained in a moment.

But other than that, you don't need to know how the tag works (or how it's built) – all you need to do is call it. And this particular one has many more features than the simple example above demonstrates (you can easily make hyperlinks on given dates, add next/previous month links, and more).

This is a perfect example of a custom tag: something that didn't exist in the language, that someone else created, that we can easily reuse to add functionality to our programs, and that has lots of flexibility.

Mar 2002

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

FIGURE 1: Output of calendar JSP custom tag

A New Source for Custom Tags

What's more important about this example is that this Calendar tag came (for free) from a site devoted to JSP custom tags, at http://coldjava.hypermart.net/servlets/. Don't be fooled by the name "coldjava" in the URL. This site really has nothing to do with ColdFusion. It's a total coincidence that the name might suggest a relationship to CF.

It's simply one of many such sites that serve as JSP tag library repositories for JSP developers. I've listed several more later in this article. They're not writing these tags for us, but for the world of JSP developers, and that's a pretty big world, which in many cases is itself only just becoming aware of the possibilities of custom tags.

So what we're talking about here is opening the door to an even larger network of possible sources where we can find reusable custom tags. The fact that they're JSP custom tags is nearly transparent to us.

A Closer Look

Let's look a little closer at the example above. I left out a few pretty important points, such as how to get the custom tag, where to put it, and some other important details.

First, this particular tag is available at http://coldjava.hypermart.

net/servlets/caltag.htm. If you can, take a look at that page. As I said, the tag will generate an HTML calendar for you, and the site shows the various syntax alternatives as well as the results of running many of its available variations.

Indeed, in the simplest JSP examples shown there, the use of the custom tag is nearly identical to the way we'd use it in CF. In fact, here's how it might show the example we used below above:

```
<%@ taglib uri="taglib.tld"
prefix="cal" %>

<cal:calendar month="3"
year="2002" />
```

These two lines are indeed an example of a fully functioning JSP page, albeit a very simple one. The only truly foreign thing is that first line, which is a JSP statement. We can't use that specific line of code but will change it to a corresponding CF equivalent. See the sidebar below for more on this.

So, the first line in the preceding sample tells the JSP page where to find the custom tag, meaning where on the server the page should locate the custom tag. We'll change that to a corresponding CF tag in a moment.

You may be thinking, "Aha! In CF we don't need to indicate where custom tags are stored. CF figures that out itself." That's true, and it's indeed a nice feature of CF. But, in fact, the CFMX designers have chosen to follow this same logic when using JSP custom tags. To actually run the simple example I showed earlier, I really need to use lines that might look like this:

```
<CFIMPORT TAGLIB="/WEB-INF/lib/
caltag.jar" PREFIX="cal">

<cal:calendar month="3"
year="2002" header="true" />
```

That first line looks an awful lot like the JSP code above. It's also a brand-new tag in CFMX. CFIMPORT tells your CF page where on your server to find the JSP custom tag library that you're trying to use. It also names a "prefix" that will describe the means by which you'll refer to tags in this library throughout this page.

Not only does this make our use of tags look more like JSP's use of them, it actually adds some benefit, which in fact will be available to ColdFusion custom tags as well, but that's beyond the scope of this article and would cross the NDA lines to say any more!

Note, too, that I said this CFIMPORT points to the tag *library*, not the tag itself. That's another difference from CF custom tags. JSP custom tags are generally distributed as libraries of several tags (often called "taglibs", hence the TAGLIB attribute name). It's a more effective way of organizing a set of related custom tags. (Boy, we could sure use a similar capability in organizing collections of related CF custom tags. Wink-wink.)

Just to clarify, the "calendar" after "cal:" is the actual name of the custom tag, or to be more technically accurate, it's the name of the custom tag handler in the taglib. Anyway, for our purposes just know that the documentation for the tag will tell you what to put on the right side of the colon. Don't worry about case sensitivity: CF won't care if the actual tag name is upper- or lowercase. Naturally, you're free to choose your own PREFIX and use that on the left side of the colon; the cases of the PREFIX and the tag name don't have to match..

I said the CFIMPORT tells CF where to find the custom tag library on your server. How do you get it on your server? And where do you place it?

JSP CUSTOM TAGS VS JSP TAGS

It's important to clarify that you can *not* mix JSP and ColdFusion syntax in a single CFMX page. We're talking here about using JSP custom tags, not JSP tags and directives. The use of the JSP custom tag (cal:calendar) is identical to that shown in the simplest JSP code example.

But you may see some examples that show the attribute values for a custom tag being obtained using JSP tags, just as we might use CF variables in the attribute values for a tag. You'd need to change those references before running the code in your CF template. The JSP syntax is often pretty easy to interpret, though not always. Again, we're just talking about translating JSP custom tag documentation into CF.

Getting a JSP Custom Tag

If you're interested in leveraging a JSP custom tag library from one of the public sites I refer to here, you simply need to find where on the site it offers a means to download the given custom tag. In the case of the site for the calendar tag offered above, there's a link to the tag's support file at the bottom of the page.

Actually, it offers links to both a ".jar" file and a ".tld" file. The .tld file is another remnant of the JSP specification for custom tags, called the "tag library descriptor" file. It's an XML file. We don't really have to have that or even understand it to leverage the custom tag. We should be able to use TLD's, and the CFMX doc shows it working, but I've had difficulty trying to use it, so I'll chalk it up to beta incompleteness.

All we really need is the .jar file, which is the actual "Java Archive" (get it? "JAR"?) that holds the Java code for the custom tag. Did I scare you just then, talking about Java? Don't worry. You don't need to know Java to use JSP custom tags. Remember, in CF it's just <cal:calendar>, which is really not that foreign.

So you want to click on the link for the .jar file, which should cause your browser to download the file (if not, right-click on the link and choose Save As). If it asks whether to open or save the file, save it. When it offers a location to save it, you'll generally want to put it in a location that's related to where your CFMX application's CF files are stored.

If your CF files are in C:\CFusion MX\wwwroot\ (which is where they might be if you choose to use the built-in CFMX Web server), then you'd also have an underlying C:\CFusionMX\wwwroot\WEB-INF\lib\ directory, and indeed that's where you want to put this .jar file. (Because CFMX is built atop a J2EE platform, this WEB-INF directory structure is a remnant of that, but you'll typically just put your CF files in the wwwroot referred to above or any subdirectory of it.)

Once the file is placed there, at least in my experience with the beta, you need to restart the CFMX server. Perhaps that will change by the time it goes to production.

> "Don't worry. You don't need to know Java to use JSP custom tags"

Pointing to the TagLib

Now, with your .jar file in place, you're ready to use the CFIMPORT. Note, again, that the JSP example shows:

```
<%@ taglib uri="taglib.tld"
prefix="cal" %>
```

But we're going to change that to:

```
<CFIMPORT TAGLIB="/WEB-
INF/lib/caltag.jar"
PREFIX="cal">
```

There are a couple of differences. Not only do we use a TAGLIB attribute rather than a URI attribute, but I'm also going to recommend (for now) that you change the latter's reference from the .tld to the .jar file you downloaded. (If I learn I've missed something about using TLD files, I'll share it in the comment area of the online version of this article at the **SYS-CON** Web site.)

Note also that the slashes must be "/" and not "\"; otherwise you'll get a "cannot find file" error. And if you do a copy and paste of a JSP taglib directive, don't forget to remove the "%" at the end of the tag before the ">" or you'll get an "invalid token" error.

One last thing: you may think, "But what if I want to use this JSP custom tag in multiple CF templates? I'll just put that CFIMPORT in my application.cfm". Sorry, you can't do that. At least not as of beta 3. Similarly, you can't even put it in a file and CFINCLUDE it. It's a bummer, but for now you must place that tag wherever you intend to use a custom tag. And there's no default central repository for JSP custom tags to be shared by all apps on a CF server.

Otherwise, that's all there is to it. The steps are:

- Find a suitable custom tag that may seem helpful.
- Download it.
- Place the .jar in the WEB-INF/lib directory.
- Create a CFIMPORT naming that directory and .jar file name and a prefix.
- Create references to the prefix and actual custom tag name in your template.

Now go find some custom tags that interest you!

Finding More Custom Tag Libraries

I mentioned previously that there are several repositories of custom tags, each with dozens or more custom tags. I've found ones for such generally useful things as creating state and country SELECT lists, performing stock quote lookups, and doing credit card number validity checking. Of course, there are CF custom tags and UDFs to do most of these sorts of things as well. But I've also found some for caching page content, SOAP processing, WAP coding, and other things that perhaps CFers may not have thought of yet.

Perhaps the best places to start are some sites set up as "portals" to keep track of available custom tags and taglibs of all sorts (see sidebar below). They're also good places to visit occasionally to see what may be new in the JSP custom tag world.

Among the sites the URLs will point to are a few specialized libraries that may or may not be generically useful for CF applications. Some of the JSP custom tags will indeed be useful only on real JSP pages, but I've found that many of them can be used perfectly well within CFMX

TAG LIBRARIES

- www.jspin.com/home/tags
- http://jsptags.com/tags/
- http://java.sun.com/products/jsp/taglibraries.html
- www.aewnet.com/root/webdev/jsp/jsptaglibs/
- www.opensymphony.com/transformtags/exampleapp/tagroadmap.jsp

ABOUT THE AUTHOR

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

A few taglibs are part of projects to bring standard use of JSP custom tags (and in some cases entire frameworks) to JSP developers. These include things like the Sun JSP Standard Tag Library (JSTL), the Jakarta Taglib project, and the Jakarta Struts project. These offer new frameworks for JSP developers in the form of taglibs, but they may be beyond the interest and reach of CF developers.

Actually, some of the tag libraries offer JSP developers the kind of easy, tag-based control over page flow and database query processing that we've always enjoyed natively in CF. That's kind of interesting to observe, making it seem like CF is the ultimate tag library.

Be careful with that phrase, though. CFMX is not "CF as a JSP tag library." It's really much more than that. Still, some JSPers may indeed see CFMX as a new, easier alternative to JSP itself, even with the many taglibs out there trying to offer a more complete framework.

Finally, a few tag libraries are intended for use with specific J2EE application servers, such as those for JRun, iPlanet, Orion, Gefion, and Oracle Java Web servers. It's likely that few of the tags in these libraries may be generally useful, but they're worth a look. You can't predict when someone in that community may think of something that may be broadly useful.

Admittedly, some of the tags out there in publicly available JSP tag libraries are no more clever than those we already have available in ColdFusion custom tags, such as those at the Developer's Exchange. But you never know. It's like finding a new directory in the Developer's Exchange, but it's a much bigger world of developers out there building JSP custom tags.

At each of these sites you'll generally find help on how to use the tags. Sometimes the documentation for using a JSP custom tag may show it using JSP syntax, but it will often be readable enough for CF programmers. More to the point, perhaps more CF people will start sharing their discoveries of useful JSP custom tag libraries, including documentation of how to use them within CF. Indeed, what we really need is a repository for people to share their observations of useful JSP custom tags. I'll start one at www.systemanage.com/taglibs/.

> **"CFMX is not 'CF as a JSP tag library.' It's really much more than that"**

Learning More

As far as learning more about using JSP custom tags within CFMX, you should look to the CFMX documentation (and release notes, new features guides, etc.) that will be made public when the product goes into production.

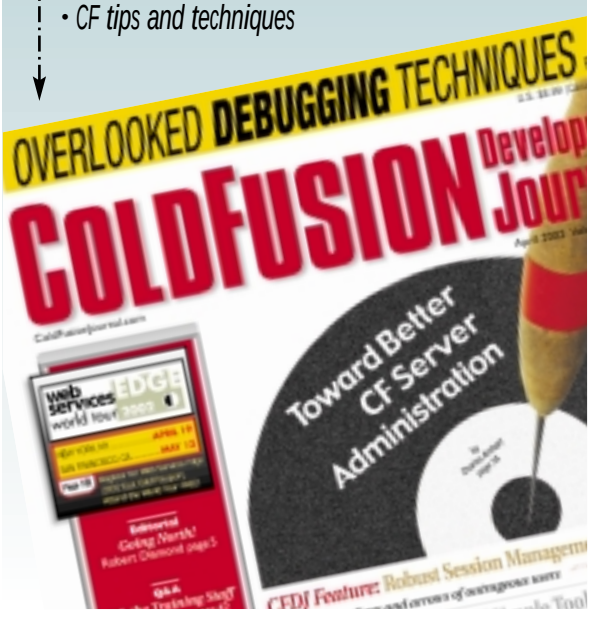
As for learning more about the general subject of JSP custom tags, several books cover it from a Java or JSP perspective. They can be helpful, but mostly for those creating JSP custom tags or using them within JSPs. I'll point you to a couple. There are certainly others. Perhaps the most complete book on the subject is *Mastering JSP Custom Tags and Tag Libraries*, by James Goodwill. It's just come out.

There are chapters in other books, as well as several articles and tutorials of the same sort online. Perhaps the best place to point you is a repository listing many of them at www.jspinsider.com/tutorials/jsp/taglibraries.view.

When All's Said and Done...

Now you have it: what JSP custom tags are, how and why you might use them within CF, and where to find them and more information about them. Should you be interested in such possibilities? I think so. And, again, you definitely don't need any Java or JSP experience to benefit from them. They can simply provide some useful, reusable functionality that you can leverage in your CF (or JSP) pages. There's quite a bit more that we could discuss about them, but this should be enough to get you started.

@ CAREHART@SYSTEMANAGE.COM



By Philip Chalmers

Robust CF Session Management

This is the second part of a two-article series about how to use ColdFusion to solve some common session management problems in Web applications. Part 1 contained:

- A summary of my recommendations
- Why you need cookies for session management
- Detecting and handling timeouts

This part discusses:

- Back, Forward, Refresh, cloned windows, and multiple Submits
- Proxy servers
- Users who enter your site via an inner page
- Multiple sessions with different browsers
- Closing a session

Back, Forward, Refresh, Cloned Windows, Multiple Submits

How do you prevent users from getting confused or entering duplicate transactions as a result of using these browser capabilities?

Chromeless Windows Are a Poor Solution

I've seen sites that try to avoid this problem by running the app in a chromeless window opened by a link on the home page. I think this is a poor solution because:

- It makes the whole system dependent on JavaScript and so excludes about 12% of users. Can you afford to turn them away?
- A user who knows the browser's function keys can use them even in chromeless windows. For example, you can't even detect the use of F5 (Refresh) in Internet Explorer.
- There are many situations where use of the "Back" and "Forward" buttons is desirable – for example, to return from the product details page to the search results or shopping cart page.
- The home page becomes a "splash" page, which many visitors will find annoying.
- Chromeless windows and other pop-ups are disreputable because of the way they've been used to push advertising at users, and many users kill them before they even see the title bar.

CFLOCK Reads and Writes of Session Variables

Suppose a user opens two windows containing a page that initiates an update of some session variables, then clicks the "Update" buttons/links on each in quick succession. I'll leave the rest to your imagination.

The "Back" and "Forward" Buttons

I wouldn't want to disable these in any way:

- Disabling Back is generally regarded as the ultimate user-hostile act on the Web.
- The user legitimately may wish to review something from a previous page.
- So you also have to allow Forward.

But there's a problem, for example, if a user completes an order, then uses the Back button or browser menu to redisplay a page used in building that order. The system may be showing parts of an order that has already been completed, instead of building a new order.

So the effect of Back and Forward doesn't matter for some pages but must be carefully controlled in other cases, mainly pages that implement a "transaction" (e.g., the checkout phase of a shopping cart) or that prepare or support a transaction (e.g., update shopping cart and view shopping cart).

Many advisors suggest using the HTML META tag or CFHEAD tag to disable the browser's cache. But then the browser will often display a screen saying something like:

Warning – this page is based on form data that has expired – do you wish to resubmit the form?

and if the user clicks "Yes," you have a potential duplicate Submit as well, which could cause duplication of an order with unpleasant consequences for the user's credit card and your reputation.

For pages that update data I suggest you create a session-level update counter and increment it whenever you get a request for a display page that can initiate an update to any significant data, whether that data is in a database or in memory (for example, a shopping cart). Include this counter value as a HIDDEN INPUT in every form that can request an update.

Avoid initiating updates via links, as then you'd have to return the counter to the server as a URL parameter and the user could edit it.

When the action page receives a request, it should compare the stored and submitted values of the counter. If the submitted value is lower, display a warning about using out-of-date versions of pages, with a link to some suitable point from which the user can resume the dialog.

Surviving the slings and arrows of outrageous users

Part 2 of 2

If the submitted value is higher, the most likely cause is that there's been a session timeout at some stage. If the stored session variable contains the counter's initial value, the timeout most likely occurred between the time the pages were displayed and when the users submitted the data and you should treat it as a timeout (apologize and link to the startup page). Otherwise the user has probably gone back to a version of the display page created before the timeout and you should display the warning about using out-of-date versions of pages.

Don't pass the counter in your session management cookie because the browser will always return the latest value even if an update request is sent via an out-of-date page.

This approach will remove the danger of out-of-date/duplicate updates when the user receives a browser message such as the one mentioned previously.

The "Refresh" Button

Refresh on its own is usually harmless. The main danger is when it's used as a response to browser warnings of the same type mentioned above, and we've already seen how to deal with that.

Cloned Windows

In 2000 there were reports of an "exploit" that relied on cloned windows that hit several shopping cart systems. A shopping cart was filled, and credit card details were authorized for, for example, $100. The perpetrator then used a previously cloned window to change the details of the goods (which the system thought were authorized by the credit card company) to a higher value. (I got this information from Macromedia's CF forum at http://webforums.macromedia.com/coldfusion/messageview.cfm?catid=3&threadid=197569).

Using an update counter avoids the risk of duplicate or inconsistent updates (or worse, additions) from cloned windows provided that you CFLOCK session variables before updating the page counter.

Multiple Submits

Using an update counter avoids the risk of duplicate updates/additions if the user clicks Submit more than once, and without this protection does not rely on JavaScript (about 12% of users disable JavaScript or use non-JavaScript browsers).

Problems with Proxy Servers

Proxy servers serve as much as possible from their own caches, so different users get the same versions of pages.

I've seen reports in Macromedia's support forums (for example, http://webforums.macromedia.com/coldfusion/arcmessageview.cfm?catid=2&threadid=19588) of proxies sharing the same CF session management cookies between users so they're sharing a session! This despite the fact that Netscape's specification for cookies (http://home.netscape.com/newsref/std/cookie_spec.html) says proxies should keep different users' cookies separate and always transmit them in both directions.

Fortunately, there's a simple solution: append a random number as a URL parameter in all links and in the ACTIONs of all forms, with the same name in all cases. This makes each user's request for the same page look different to the proxy server. CF can generate random integers from 1 to 100,000,000, so the risk that the same page will be requested with the same number by users on the same proxy server within the same week is extremely small.

Note: In forms the random number must be part of the ACTION, not a HIDDEN INPUT, because this technique requires the random number to be part of the URL as shown in the address/location bar – that's what makes the proxy server think it's a different page.

It doesn't matter if the user edits the address/location bar. If two users edit it to the same value, the session update counter described above will prevent them from doing anything that compromises your app. Your app will simply ignore the random number parameter.

Users Who Enter Via an Inner Page

Of course, users who enter via an inner page are welcome. They're more likely than most visitors to be interested in what your site offers because they've probably got the URL from a bookmark they created after a previous visit, a search engine query, or an e-mail from a friend who knows what they're interested in.

For these users, required session variables may not exist, and the app's session management cookie doesn't exist unless this is a revisit. The checks on cookies and updating sequence numbers will handle both situations.

Your "please enable cookies" page must allow for visitors to inner pages. Welcome them, explain why this page has appeared and what could have caused it, ask them nicely to enable per-session cookies, and direct them to your startup page. And links to "about us" and other information-only pages should probably be offered for visitors who've found your site via a search engine or an e-mail – they need reassurance that they've come to the right place.

Multiple Sessions with Different Browsers

CF creates a separate session for each different browser on the same client PC (e.g., Internet Explorer and Netscape) because each browser has its own set of cookies and therefore different CFID and CFTOKEN values.

If you need to prevent or control multiple simultaneous sessions with the same user, you can:

- Require users to log in when the dialog reaches a point where control is needed. You then have the option to reject duplicate log-ins.
- Maintain an application-level structure keyed on userid and containing a time stamp updated on every page request from the user.
- Schedule a process to remove entries older than the app's session timeout interval.

Closing a Session

You can't force the user to close a session. He or she can simply use the browser to go somewhere else (unless you run your app in a chromeless window) or close the browser window.

But you may suggest that the user should log out for his or her own security. You could even include JavaScript to create a "please logout" pop-up window if the user leaves your site without logging out. But don't rely on it. Remember, 12% of users run without JavaScript and the rest can close the pop-up.

If the user does log out:

- Don't use StructClear(Session) in CF 4.5 or later to kill the session; the user may legitimately be using another window to talk to another app on the same site. Instead, clear the critical session variables individually. You can use the app's session management cookie to identify them. Any other app the visitor may also be using should have a separate session management cookie.

- Delete all name-value pairs from the session management cookie except for "cookiesOn=yes". If you delete this, the app will show the "please enable cookies" page if the user returns. Don't delete the CFID and CFTOKEN cookies because the user may legitimately be using another window to talk to another app on the same site.

References and Additional Information

- *Macromedia's support forums:* Information relevant to the scope of this article, in particular, postings about the hack using multiple browser windows (search this page at http://webforums.macromedia.com/coldfusion/messageview.cfm?catid=3&threadid=197569 for the word *exploit*); the history of CF session and client management (Steve Nelson's February 1, 2000, posting in this thread at http://forums.allaire.com/coldfusion/arcmessageview.cfm?catid=2&threadid=160630 neatly summarizes the pros and cons of various techniques); several items about the risks of passing CFID and CFTOKEN as URL parameters, including http://webforums.macromedia.com/coldfusion/messageview.cfm?catid=12&threadid=273249 (robbart's posting at 4:30 p.m. on February 25, 2002, points out that search engines can pick up CFID and CFTOKEN from URL parameters and serve them to all their users!); the problems with proxy servers (http://webforums.macromedia.com/coldfusion/arcmessageview.cfm?catid=2&threadid=19588).
- *Netscape's specification for cookies:* http://home.netscape.com/newsref/std/cookie_spec.html.
- *Web browser usage and capability stats:* www.upsdell.com/browsernews/res_design.htm and www.w3schools.com/browsers/browsers_stats.asp.
- Funaro, M. (2000). "So You Want to Manage a Session on Load-Balanced Servers?" *CFDJ*, June (Vol. 2, issue 6).
- *Another method for checking for cookies:* www.thenetprofits.co.uk/coldfusion/faq/ (I don't like it because it causes flicker and messes up the Back button if JavaScript is disabled).
- *Webmonkey's tutorial on cookies and CF:* http://hotwired.lycos.com/webmonkey/00/29/index3a.html; This explains the basics of using cookies, including some points where the ColdFusion documentation is incomplete or misleading, and gives links to other useful information about cookies.
- *Independent source of information about cookies:* www.cookiecentral.com/faq/; section 2 addresses the user's concerns in nontechnical language. I link to it on "please enable cookies" pages to reassure users.
- *Shopping cart:* www.freecfm.com/w/Webdeli/Phils_Deli/index.cfm (which I wrote) uses all of these techniques except logging out and controlling multiple sessions by the same user.

About the Author

Philip Chalmers has been working in information technology since the early 1970s. He's relatively new to ColdFusion, but has specified, designed, and developed systems on platforms ranging from mainframes to PCs in about a dozen languages. His first technical publication appeared in 1979, when he wrote several sections of the Adabas Design Guide.

@ PHILIPCHALMERS@BLUEYONDER.CO.UK

Ask the **Training Staff**

A source for your CF-related questions

BY
BRUCE VAN HORN

This month we have three questions to consider. I like them all because they're very practical and applicable to many kinds of applications.

I particularly like the third question about encryption methods because it gives me an opportunity to cover two undocumented ColdFusion functions. I hope you find my answers helpful. Keep those questions coming!

Q: *My question has to do with forcing session variables to time out after a specified period of time – not a period of inactivity. I am writing an online testing application that should allow a user only 10 minutes to take a test. How can I force their sessions to expire after 10 minutes?*

A: This is easily accomplished by setting a session variable, let's say "Session.StartTime", equal to the current time using the Now() function (see Listing 1). Your exam pages need to test for the existence of this variable. If it does exist, check to see if the time set in that variable is more than 10 minutes old. Use the DateDiff() function to check the number of minutes between Session.StartTime and the current value of Now().If the value of DateDiff() is greater than 10 minutes, delete the session variable and redirect the user to another page. If the user tries to go back to the page without Session.StartTime having been created, redirect them to an error page.

Q: *I am writing a domain name registration application and need an easy way for users to input multiple domain names to check against a whois search. I know I can build a separate form field for each domain name, but that gets pretty messy if they need to check more than five domains. I'd like them to just enter all the domains they want checked into a single form field, but I'm not sure how to parse out the individual domain names once the form is submitted. How do I do this?*

A: This is a great question and somewhat piggybacks on one of the questions in last month's column. What you do is create a TextArea form field that allows users to input as many domain names as they want, each domain on a separate line in the text box. This concept works for any kind of data that needs to be submitted in bulk (e.g., e-mail addresses, product codes, keywords). The key to success is having each entry on a separate line. This is easy for users because they can copy data from a spreadsheet or text document and just paste the data into the form field. Once the form is submitted, you simply treat the data as a list delimited by a carriage return and a line feed. Your CF code needs to loop over the list using chr(13) and chr(10) – the ASCII codes for a carriage return and line feed, respectively – and then perform whatever code you need to execute for each iteration of the loop (in your case, check each domain name against a whois search). See Listing 2 for a simple example.

Q: *I need to store sensitive information (social security numbers, passwords, etc.) in a database and don't know how to protect the data from being seen by the other developers working on the application. Any suggestions?*

A: Wow, an entire issue of ***CFDJ*** could (and maybe should) be devoted to this one question! I won't go into any Web site or database security issues here, but I will address how to encrypt the data in the database using a few different methods. I can't believe how many applications I encounter that simply store this kind of data (even credit card numbers!) without using any kind of data encryption. These developers are taking huge security risks that aren't necessary, given how easy it is to employ basic encryption and decryption methods.

First, if the data, once set into the database, never needs to be displayed again to a user but simply used for comparison or validation (like a password or social security number), I'd use the Hash() function. This CF function provides a fairly strong encryption algorithm, but it's an encrypt-only function. There is no way to decrypt data stored using Hash(). This is useful for doing validation since you can compare two strings that have been encrypted to see if they match. Listing 3 gives an example of using Hash().

At a bare minimum you should use the Encrypt() and Decrypt() functions, although the encryption algorithm isn't very strong. It will at least disguise the actual value from the casual database viewer. If you do need to retrieve and display (decrypt) the data back to the user or for application use (like charging a recurring fee to a credit card), I prefer to use the undocumented Cfusion_Encrypt() and Cfusion_Decrypt() functions, as they provide stronger encryption. The syntax is the same for either pair of functions. You need to supply an arbitrary key/seed value for the encrypt/decrypt processes. Be sure to use the same key/seed you used to encrypt when decrypting. Listing 4 gives an example.

• • •

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

@ BRUCE@NETSITEDYNAMICS.COM

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

Listing 1

```
<!--- When the user starts the exam, set a session vari-
able equal to the current time --->
<cfset Session.StartTime = Now()>

<!--- In your exam pages, use this code to check for the
Session.StartTime variable. If it exists, check to see if
the time stamp is more than 10 minutes old. If it is,
delete it and redirect users to another page. If the vari-
able doesn't exist, redirect them to an error page --->
<cfif isDefined("Session.StartTime")>
 <cfif DateDiff("n",Session.StartTime,now()) gt 10>
  <cfset tmp = StructDelete(Session,"StartTime")>
  <cflocation url="ExamExpired.cfm">
 </cfif>
<cfelse>
 <cflocation url="YouMustStartExam.cfm">
</cfif>
```

Listing 2

```
<!--- Create a form with a textarea field for data entry.
Tell your users to put each entry on a separate line! --->
<cfform action="#cgi.script_name#" method="POST">
 <textarea cols="35"
           rows="10"
           name="DomainNames"
           wrap="off"></textarea>
 <br>
 <input type="Submit" value="Check Domains">
</cfform>

<!--- In your form action code, set a variable equal to
the carriage return and line feed ASCII codes, then use
that as the delimiter for your list --->
<cfif isDefined("Form.DomainNames")>
 <cfset crlf = chr(13) & chr(10)>
 <cfloop index="DomainName"
         list="#Form.DomainNames#"
         delimiters="#crlf#">
   Execute code for each #DomainName#
 </cfloop>
</cfif>
```

Listing 3

```
<!--- When inserting a password, use the Hash() function --
->
<cfquery datasource="MyDataSource">
 INSERT INTO Users (UserName, Password)
 VALUES ('#Form.UserName#','#Hash(Form.Password)#')
</cfquery>

<!--- Use Hash() when changing passwords --->
<cfquery datasource="MyDataSource">
 UPDATE Users
  SET Password = '#Hash(Form.Password)#'
 WHERE UserID = #Val(Session.UserID)#
</cfquery>

<!--- Use Hash() when validating a user --->
<cfquery datasource="MyDataSource" name="qCheckLogin">
 SELECT UserID, Name
 FROM Users
 WHERE UserName = '#Form.UserName#'
   AND Password = '#Hash(Form.Password)#'
</cfquery>
```

Listing 4

```
<!--- Use CFusion_Encrypt() to encrypt the data --->
<cfset EncryptedData = CFusion_Encrypt("Bruce Van
Horn","MyKey")>

<!--- Use CFusion_Decrypt() to decrypt the data --->
<cfoutput>
#Variables.EncryptedData#<br>
#CFusion_Decrypt(Variables.EncryptedData,"MyKey")#
</cfoutput>
```

CODE LISTING

The code listing for this article is also located at

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

Click Portal Server/eWorkspace Server from Intrafinity

Manage your projects with eWorkspace Server

REVIEWED BY
TOM MUCK

A portal server can be a great addition to the operation of a company and a big plus for commercial Web sites.

For one, it can allow employees to access everything through one main page, from the local weather to their Outlook boxes and calendars. In addition, administrators can have control over the content and can offer a standard configuration for everybody to use within a department or the entire company. For commercial Web sites, portals allow a visitor the opportunity to create a personalized home page.

Intrafinity Inc. offers a ColdFusion-based portal server, Click Portal Server, that is customizable, allowing a business to add its own modules when relevant or as needed. It also enables the organization and centralization of all the information and applications necessary to run an enterprise day to day. The Click Portal Server is also useful for bringing clients, suppliers, and customers into the overall framework of an organization by giving them access to specific portions of the portal (see Figure 1).

eWorkspace Server

On top of Click Portal Server the company has added another product, the standout of the two. The eWorkspace Server is a customizable workspace and content server that allows a company to store information, documents, schedules, and communications on a project-by-project basis. This combination is versatile enough for a small company but powerful enough for a large corporation (see Figure 2).

What makes the product powerful is the sophisticated permissions framework that allows the creators or administrators of each document or area to administer the permission levels of the objects. Users can be set up in different groups that have different access levels. A shared workspace can be accessible by team members, salespeople, administrators, and even clients. This allows users to create a central location where everything related to a specific project can be stored and accessed.

Both applications are ColdFusion based, built with a SQL Server 2000 back end, with an Oracle-compatible version slated for release in late May; the Click Portal Server is available in SQL Server and Oracle-based ASP and Java versions as well. The server will run on ColdFusion versions 4 and up.

Ease of Use

It has always been my contention that a product should (1) be easy to use and (2) have a familiar interface so that you can begin work immediately without having to open a manual to learn where everything is. The eWorkspace Server uses the standard Windows Explorer–type interface, which is immediately familiar. This is a big plus. The interface is intuitive and easy to navigate. Because it's a ColdFusion application, response time is quick. The interface is clean and uncluttered.

Installation is a snap as well. The installer automatically added the complex SQL Server database to the

FIGURE 1: Click Portal Server

FIGURE 2: eWorkspace Server

local SQL Server 2000 instance, and also automatically configured IIS with a variety of virtual directories. One minor hitch was solved with a quick call to tech support – cookies were off in my browser, which prevented me from accessing the administrator pages. The staff at Intrafinity was responsive and knowledgeable.

Organization

The key to a successful project is organization. If your project is well organized it will more likely be completed on time and within budget. The eWorkspace Server provides the tools you'll need to organize your projects. As a Web developer, I saw the eWorkspace Server as a way to organize and maintain Web projects and files. It's capable of doing this, but it actually can be used for any type of project. Another feature of the product allows projects to be set up as templates. As you develop projects, you often find yourself following the same procedures from project to project and, over time, have developed a set of best practices to ensure their success. By implementing a template, you can use the same basic structure for each workspace and save time in the process.

Another nice feature is the version control for documents. This allows team members to make changes in documents while maintaining the originals. New versions can be set as the "active" version, while others remain in place with version numbers and creation dates. The server handles all this automatically. All you do is upload the documents. This can be useful for team collaboration on a single document.

A built-in calendar allows schedules to be set up and maintained. Like other aspects of the server, calendar events can be commented on, voted on, and also monitored for activity.

Content is also searchable, with full-text search capabilities on all documents.

Collaboration and Communication

Collaboration is key for a successful project. Collaboration is based on communication, thus both functions can be well served by a workspace server. E-mail is a good means of communication, but e-mails can be lost, disorganized, or unavailable to the people who might need them.

VITALS

Click Portal Server / eWorkspace Server Suite
Intrafinity Inc.

Address: 24 Cecil St., 2nd Floor
Toronto, Ontario
M5T 1N2, Canada

Phone: 1 866 802-4447

E-Mail: cfdj@intrafinity.com

Web: www.intrafinity.com

Test Environment: Windows 2000 Server, ColdFusion 5 Enterprise, SQL Server 2000. 750 MHz Athlon, 512 MG RAM

Pricing: Flexible to meet customer needs: server license (varies) plus $32-$50 per user

The eWorkspace Server provides several mechanisms for easy communication among team members. Threaded discussion forums offer the best level of intraproject communication. Topics can be started and administered by anyone in the group and votes on different topics can be organized as well.

Another feature of the program I particularly like is the Activity tab. By clicking on this tab, you can view all activity for a specific item. This is handy for team leaders or administrators to check the progress of different team members. You can easily determine from this tab whether a specific team member has been able to view the information yet, or if a client has previewed a specific document.

E-mail notification of events or changes in a selected part of the workspace can be sent to team members daily, or when the event occurs. This is especially handy because it's all done automatically: you don't have to create e-mail messages manually to alert your team members of a change. Each aspect of the server has checkboxes to allow instant or daily notification of changes or activity in a particular object. For example, if a team member has accessed a particular document that you created and uploaded, a notification can be sent to you immediately.

Messaging is also possible with the built-in chat mechanism. While it isn't the most sophisticated chat messaging around, the fact that it's tied into the package makes it very useful. In addition, third-party chat servers can be incorporated into the server.

One of the goals of a company is to better serve its customers, and one way to do this is to decrease the separation between the company and its customers, suppliers, and partners. By using eWorkspace Server's object-level permissions, you can set up extranets that bring these outside forces into the eWorkspace Server's collaborative environment.

ABOUT THE AUTHOR

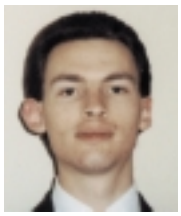
As senior applications developer for Integram in northern Virginia, Tom Muck develops back-end applications for expedited, electronic communications. A Team Macromedia member, Tom has coauthored three UltraDev books including Dreamweaver UltraDev 4: The Complete Reference.

@TOMMUCK@BASIC-DRUMBEAT.COM

Using MS-SQL Stored Procedures with ColdFusion

Improving site performance

BY IAN RUTHERFORD



My three-part article on stored procedures (***CFDJ***, Vol. 3, issues 10, 12, Vol. 4, issue 2) has mushroomed into four, but since I'm sure most of you write perfect code the first time…

…this extra part will be of interest only to the small group of programmers who occasionally make mistakes when coding.

Stored procedure programming with ColdFusion has many benefits including speed, features, and data validation. However, coding errors in stored procedures can cause some of the most cryptic and sometimes misleading error messages you'll ever see. There are several issues and at least one bug to be aware of when working with stored procedures.

Decyphering Error Messages

First let's look at some of the common error messages. The most useful messages you'll receive occur when you have a coding error that prevents the stored procedure from compiling. For example, if you write a procedure with the following code:

```
CREATE PROCEDURE dbo.pr_test
AS
SELECT
  *
FROM
  testing
```

and you don't actually have a table named "testing", you'll receive the following error:

```
ODBC Error Code = S0002 (Base
table not found)
[Microsoft][ODBC SQL Server
Driver][SQL Server]Invalid
object name 'testing'.
SQL = "pr_test"
```

The most common error message you'll probably receive is:

```
ODBC Error Code = 22005 (Error
in assignment)
[Microsoft][ODBC SQL Server
Driver]Invalid character value
for cast specification
SQL = "pr_test"
```

Unfortunately, the message doesn't tell you which line is causing the problem or whether the problem lies in your ColdFusion code or in the T-SQL code. The first thing to do is to make sure that the data types in your <cfprocparam> tags match the values in your stored procedure and are in the same order. The next thing to check is that the values you're inputting into the <cfprocparam> tags match the data types you've chosen. Finally, check to make sure that any <cfprocparam> tags that are of type "out" actually return the correct variable type. (Later I'll show you how to use the Query Analyzer Debugger for this purpose.)

Another common error, especially when you're coding a procedure with lots of parameters, is:

```
ODBC Error Code = 37000 (Syntax
error or access violation)
[Microsoft][ODBC SQL Server
Driver][SQL Server]Procedure
'pr_test' expects parameter
'@BillToEmail', which was not
supplied.
```

This error is caused by leaving out a <cfprocparam> tag, which results in the stored procedure's sending fewer arguments than the stored procedure expects.

Finally, you can receive the following error, which is caused by sending too many <cfprocparam> values to the stored procedure:

```
ODBC Error Code = 37000 (Syntax
error or access violation)
[Microsoft][ODBC SQL Server
Driver][SQL Server]Procedure or
function pr_test has too many
arguments specified.
```

You may receive other messages, but these are the most common.

A Bug

Now that you know some of the errors that can result from coding mistakes, let's look at a bug that can cause strange results in your code. First, look at the procedure in Listing 1. All listings can be accessed at www.sys-con.com/coldfusion/sourcec.cfm. Notice that there's a variable called @VendorCreditLine listed as input but that the stored procedure doesn't do anything with it. Second, assume that you forgot to set a default value for the column in the Vendor table for VendorCreditLine. When you run the procedure you'll receive the following error:

```
ODBC Error Code = 23000
(Integrity constraint violation)
[Microsoft][ODBC SQL Server
Driver][SQL Server]Violation of
UNIQUE KEY constraint
'IX_users_userID_Constraint'.
Cannot insert duplicate key in
object 'Users'.
```

	Column Name	Data Type	Length	Allow Nulls
▶🔑	CartDetail_Pk	int	4	
	Item_Fk	int	4	

FIGURE 1: Design View in the SQL Enterprise Manager

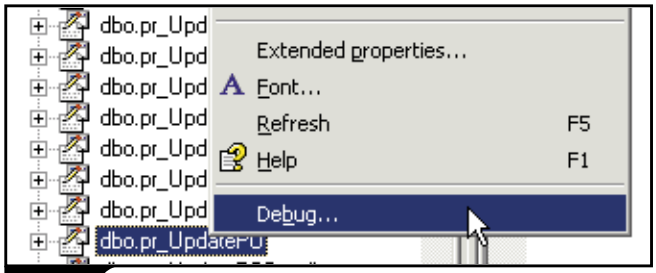


FIGURE 2: Starting debugging in the Query Analyzer

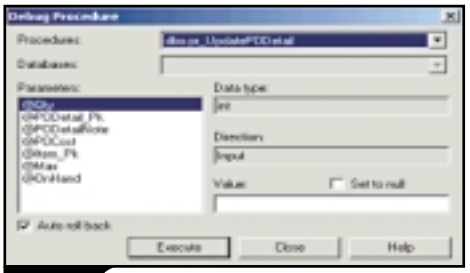


FIGURE 3: The configuration window for the debugger

This would lead you to assume that somehow you had tried to insert a userid that was already in the database. However, if you search the database for the ID you'll never find it. The bug, caused by the database driver and not ColdFusion (according to Macromedia), causes the procedure to execute twice instead of erroring when it encounters an error. Since the procedure executes for a second time before rolling back the Insert code in the User table, the database thinks it has already input the information. To catch the error you need to run the procedure directly through Query Analyzer, which properly throws the correct error (a failed attempt to insert nothing into the VendorCreditLine column, which requires an integer).

A second problem, which may at first seem like a bug but is not, occurs when you use <cfprocparam> tags of type "Out" and cfsqltype "char". If you try to run a Len() function in ColdFusion without running Trim() first on a char value, you'll get a "true" result, even if the result is blank, because the char data type pads the field to its full length. If you need to see whether the char really has any content, run Len(Trim()) instead of just Len().

Problems with Numbers and Null

When you choose data types, make sure the type you choose is really the one you need. Decimal and integer data types can create hard-to-find math errors if you choose the wrong one. The integer data type can only take whole numbers (0, 1, 2, etc.). If you run the following code:

```
DECLARE
  @MyInteger [int]
SET @MyInteger = 3
SET @MyInteger = @MyInteger/2
```

@MyInteger won't equal 1.5; it'll equal 2 because all fractional math is rounded in the Integer data type.

The decimal data type can contain whole numbers but also decimals. (This should be obvious.) However, decimal also has two attributes that can affect the results you get from math operations. The first is Scale; the second, Precision. Scale tells how many digits appear to the right of the decimal. Precision tells how many digits the number can contain. When you create a decimal field in a table, it defaults to a scale of 0 and a precision of 18. Make sure you change this or your decimal values will basically be treated like integers. To set a decimal variable you'll want to use the following syntax:

```
DECLARE
  @MyDecimal [decimal](precision,scale)
```

Make sure you set the precision value large enough to avoid having your numbers truncated.

NULL is a strange beast that can cause many problems in your database. I strongly suggest never using it. When you're performing math operations, NULL trumps everything, resulting in NULL results. To avoid NULL values being inserted by ColdFusion into your stored procedures, always set the NULL attribute to "no":

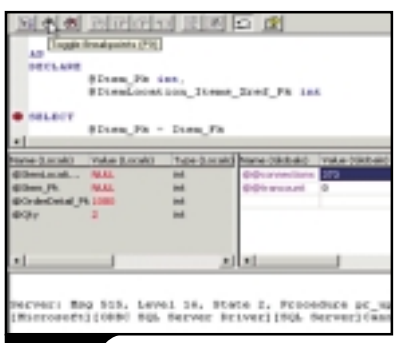


FIGURE 4: Main debugger window

```
<CFPROCPARAM TYPE="In"
  CFSQLTYPE="CF_SQL_CHAR"
  VALUE="#Client.UserID#"
  NULL="No">
```

In your database, always uncheck the "Allow Nulls" box in the design view () (see Figure 1).

Debugging with the Query Analyzer Debugger

The easiest and most reliable way to track down problems with a stored procedure is to use the Query Analyzer Debugger that comes with MS-SQL Server. This tool allows you to input variable values, set stop points, and view the values of all variables throughout the procedure. To use the debugger right-click a procedure in Query Analyzer and select "Debug" (see Figure 2). You'll then see the set window for the debugger (see Figure 3).

Put in a value for each of the "Input" variables and set the "Input/Output" variables to null. (The variables you set as output variables in your stored procedure will appear as "Input/Output" in the debugger.) If you just leave the output variables blank instead of setting them to null, you'll receive error messages when you run the debugger. Also, make sure the "Auto Rollback" box is checked or you'll actually make database changes when you run the debugger. When you click "Execute" you'll see the screen in Figure 4.

The top pane contains your procedure and shows breakpoints (the red dots in the left gutter). The middle pane is broken up into three sections. The first shows the values of all local variables at each step in the procedure. The second shows global variables. The third (not shown) lists all procedures being called by the debugger. If you nest procedures, this is where you'd see them. There is also a bottom pane that contains any return codes and record sets that your procedure creates.

To debug your procedure, set several breakpoints in critical areas (places where variables get set) so you can confirm that the values are what you were expecting. Using the Query Analyzer Debugger will help you code your stored procedure properly and provide some assurance that it will function in your application. But how should you handle errors that crop up in ColdFusion?

When you run the debugger, you should see "@Return_Value = 0" in the bottom pane (if the procedure ran correctly). This value can be accessed in ColdFusion using the cfstoredproc.statuscode variable. To get the value back from the database, set "return Code='yes'" in the <cfstoredproc> tag.

In your CF code you can use <cfif> to handle errors. For example:

```
<cfif cfstoredproc.returncode>
  Put error code here.
<cfelse>
  Put normal processing here.
</cfif>
```

> "Coding errors can cause cryptic and misleading error messages"

Unfortunately, even if the stored procedure ran incorrectly, it's still possible that it changed things in the database. To prevent this from happening on the CF side, use transaction tags. For example:

```
<cftransaction>
  (your stored procedure)
< cfif cfstoredproc.returncode >
  <cfset Client.Message="Your
stored procedure broke.">
  <cftransaction action="roll-
back"/>
<cfelse>
  <cfset Client.Message="The
procedure ran correctly.">
  <cftransaction action="com-
mit"/>
</cfif>
</cftransaction>
```

It's also possible to code error handling within the stored procedure (see Listing 2). Notice that I check the value of "@@Error" after each statement and that I use transaction statements around each query.

Another possible way to enhance your error handling is to include an output variable that returns custom error messages from the procedure to ColdFusion (see Listing 3). This allows you to create user-friendly messages that can tell people exactly what went wrong in your application.

• • •

I hope this final article on using stored procedures with ColdFusion will help you become a more efficient coder as well as produce better applications. If you have any questions, please e-mail me.

ABOUT THE AUTHOR

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

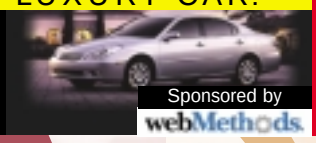
@ IRUTHERF@CATHOLICSTORE.COM

LARGEST GATHERING OF DEVELOPERS, PROGRAMMERS, AND *i*-TECHNOLOGY PROFESSIONALS IN THE WORLD!!

JUNE 24–27, 2002 • JACOB K. JAVITS CONVENTION CENTER • NEW YORK, NY

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!



Sponsored by webMethods

REGISTER BY MAY 31 TO SAVE $200!

GO TO WWW.SYS-CON.COM NOW!

JAVA

Focus on Java

Java, now mainstream, is the dominant back-end technology upon which next-generation technologies are evolving.

Hear from the leading minds in Java how this essential technology offers robust solutions to *i*-technology professionals and senior IT/IS strategy decision-makers.

JAVA IN JUNE ESPECIALLY IN NEW YORK

JDJEDGE conference&expo

Java Track: June 25–27

The Java Track, with something for every developer from beginner to advanced, will also look into the role that Java is playing in building up Web Services.

- JV1 Java Security Advanced Concepts
- JV2 Optimizing Database Performance in J2EE Applications
- JV3 Detecting, Diagnosing, and Overcoming the Five Most Common J2EE Application Performance Obstacles
- JV4 Building Asynchronous Applications Using Java Messaging
- JV5 Building "Smart Client" Applications Using J2SE and J2ME
- JV6 .NET vs J2EE
- JV7 Building Scalable Web Applications and Web Services
- JV8 Hot Breaking Session
- JV9 Java Tools for Extreme Programming
- JV10 Building Truly Portable J2EE Applications
- JV11 Security for J2EE Application Servers

XML

Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

XML NEXT G OF ENTERPRISE DEPLOYMENT

XMLEDGE conference&expo

XML Track: June 25–27

The XML Track will focus on the various facets of XML technologies as they apply to solving business computing problems. This track targets audiences ranging from beginners to system architects and advanced developers.

- XM1 Data – a Key Part of Web Services
- XM2 OASIS Standards Update
- XM3 A Universal Business Language
- XM4 Achieving Standards-Based Mobile eBusiness Success with XML & Web Services
- XM5 Using XML for Rapid Application Development and Deployment with Web Services
- XM6 Open Up Your RDBMS with Open Standard Technologies
- XM7 XML Web Services: Standards Update
- XM8 Bringing XML to PKI
- XM9 Building a Corporate Information Model in XML
- XM10 XML in the Enterprise and Inter-Enterprise World
- XM11 XML and RDB Relationships

WEB SERVICES

Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at Web Services Edge East 2002.

Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting-edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Keynote panel that will debate and discuss the realities and promise of Web Services.

WEB SERVICES SKILLS, STRATEGY, AND VISION

web services EDGE conference&expo

Web Services Track: June 25–27

The Web Services Track will concentrate on the latest emerging standards. It will include discussions of .NET, Sun ONE, J2EE and App Servers, the role of UDDI, progress of the standards-making bodies, SOAP, and Business Process Management. It is intended for developers, architects, and IT management.

- WS1 Starting Out in Web Services: Fundamentals in Web Services
- WS2 State of the Web Services Industry
- WS3 Web Scripting Languages: Options for Dynamic Web Development
- WS4 Building a Web Services Application Infrastructure
- WS5 Deploying a Corporate Portal
- WS6 The Enterprise Service Bus (ESB): Leveraging Web Services
- WS7 Developments in Web Services Standards
- WS8 Unlocking the Value of Enterprise Web Services
- WS9 Guarding the Castle: Security and Web Services
- WS10 Practical Experiences with Web Services and J2EE
- WS11 Designing Web Services Using UML

.NET TRACK: June 25–27

Combining technology, platform, and architecture, the .NET Track provides insight on the latest developments for the Windows platform.

- NT1 Configuring .NET for Security, Performance, and Reliability
- NT2 Changing Your Environment to .NET
- NT3 Going Mobile with .NET
- NT4 .NET on Other Platforms (FreeBSD, MONO, Portable .NET)
- NT5 Inside the CLR
- NT6 Accessing Data from .NET
- NT7 Advanced .NET Web Services
- NT8 Migrating Legacy Code to .NET
- NT9 Advanced User Interfaces with GDI+

IT Strategy Track: June 25–27

The IT Strategy Track will focus on the managerial and design aspects of the various development disciplines. Topics will include Standards, Architecture, Design Patterns and Best Practices, Project Planning and Management, and Extreme Programming.

- IT1 Developing, Deploying and Managing Web Services
- IT2 Key Trends and Technologies for Building an Enterprise Web Services Architecture
- IT3 Selecting a Framework: Toolkit, Platform, or Roll Your Own?
- IT4 Getting Started with Web Services
- IT6 Overcoming the Web Services Barriers
- IT7 The Real Issue: Improving Your Enterprise with Enterprise Web Services
- IT8 Minimizing Risks and Maximizing Investments in J2EE Development Through the Use of Reusable Application Architecture and Frameworks
- IT9 Application Integration – Building a Flexible Web Services Architecture
- IT10 The Economics of Web Services
- IT11 Hooking It All Together - Application Integration Case Study

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- *i*-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT /C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Sun Microsystems at JDJEdge 2002: Java Fast Paths & Java University℠ Program at the Jacob K. Javits Convention Center, NYC

Java Fast Paths

Attend fast-paced, code-level, full-day Java technology developer training designed to provide you with the skills to confidently approach the industry's core Java technology certification exams. Don't just say you know it… prove it! Most developers recognize the expanding importance of gaining Java technology certification. If you're like those who've already taken the exams, the real hurdle to certification is finding time to prepare for the tests.

Monday, June 24, 2002:

- Java™ 2 Platform: Developer Certification Fast Path
- Java™ 2 Platform: Architect Certification Fast Path
- Web Component Developer: Certification Fast Path

Java University℠

The Java University℠ Program complements this year's JDJEdge Conference by offering three aggressive, full-day, code-level training classes for experienced software developers, architects and engineers.

Attend code-level training in sessions designed by industry luminaries, and recognized experts. Sessions cover XML and Web services technology. Whether you're a beginning or a veteran developer, architect, or software engineer, you'll benefit from these value-packed full-day courses. Register now. Seating is limited.

Tuesday, June 25, 2002
Developing Solutions Using Java™ Technology and XML

Wednesday, June 26, 2002
Web Services Programming Using Java™ Technology and XML

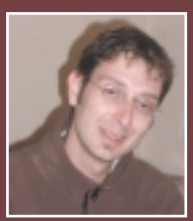
Thursday, June 27, 2002
Java™ APIs for Enterprise Web Services

FOR EXHIBIT INFORMATION CONTACT: MICHAEL PESICK, SYS-CON EVENTS, 135 CHESTNUT RIDGE RD. • MONTVALE, NJ 07645 • 201 802-3057 MICHAEL@SYS-CON.COM

OWNED AND PRODUCED BY SYS-CON EVENTS SYS-CON MEDIA

Tales from the **List**

Efficiently inserting bulk data



BY **SIMON HORWITH**

ABOUT THE AUTHOR

Simon Horwith, a senior developer and Macromedia-certified ColdFusion instructor at Fig Leaf Software in Washington, DC, has been using ColdFusion since version 1.5. He is a contributing author to Professional ColdFusion 5.0 *(WROX) as well as technical editor of* The ColdFusion 5.0 Certification Study Guide *(Syngress). This spring Simon plans to launch www.how2cf.com, a ColdFusion "how-to" site hosted by cfdynamics.*

The CFDJList discussion thread we'll examine this month began with a post by list subscriber Helen Warren. Helen's e-mail described one of her daily on-the-job tasks of parsing a pipe-delimited text file containing approximately 30,000 records (about companies) and either inserting or updating data in a SQL Server 2000 database based on the text file's contents.

In order to accomplish this, Helen first uses SQL Server's BULK INSERT command to import the data from the text file to a temporary "holding" table. She then loops over the records in the holding table using a CURSOR, and performs either an insert or update on a company information table in her database, depending on whether or not the company already has an entry in the table. She then adds a record to a second table in the event that this is either a new company or that certain column values have been updated in the company table record. Helen's question was whether there are better-performing alternative methods to accomplish this same task.

One possible suggestion was to convert the pipe-delimited file to XML and use SQL Server 2000's XML parsing functionality to pass the XML-formatted data directly to a stored procedure. Of course, SQL Server's XML parsing in this scenario would be efficient only for the bulk insert portion of her task, not for any necessary updates. To make this work, Helen would have to predetermine which entries in the text file are required for insert (that is, which belong to companies that don't exist yet in her database), format those records as XML, and pass that XML to a stored procedure that performs the insert (using the openXML command). The remaining data would then need to be inserted into a holding table, looped over with a cursor, and passed to a stored procedure that performs the necessary updates one row at a time. The advantages to this approach are:

- The data (and data manipulation routines) used to insert or to update the database would be separated, which results in fewer records having to be parsed by a cursor as well as more modular data processing logic.
- It's possible that the data, once in XML format, could be more efficiently worked with either on the SQL Server or in CFML.

What are the drawbacks to this idea?

- A routine that transforms the text file into XML would have to be written and then executed on the data. It could be costly to predetermine which records need to be inserted and which updated.
- New stored procedures would have to be written and tested, including a procedure to parse XML and map its elements to table columns.
- Typically, XML representations of data are more verbose, though more descriptive, than pipe-delimited data. Thirty thousand records would make for a rather large XML packet.
- The use of cursors is still required, though not as many records would have to be cursored over.

Another suggestion was to use DTS to import the data and manipulate it. Though this approach is one that would allow the import and all data manipulation (inserts/updates) to be performed by a single database procedure, Helen quickly pointed out that in cases where the text file to be imported resides on the local or network file system, BULK INSERT can be up to twice as fast as DTS. This is because BULK INSERT doesn't have to go through SQL Server's NetLib Network communication layer in order to open and retrieve data from flat files.

The conclusion of the thread was that for the time being, Helen is going to leave things the way they are: using BULK INSERT and cursors. She'll continue to look into some of the functionality available with SQL Server 2000's XML parsing constructs, because though performance isn't too bad now, she can foresee a time when the use of cursors will result in poor performance due to the larger amount of data or to changes to the business logic requirements.

As a follow-up to the discussion, Joel Firestone asked how (syntactically) to perform a bulk insert on a SQL Server database table. Helen responded with this syntax example:

```
BULK INSERT t_loadtable FROM 'c:\My
Documents\Torchbox\abc1000\abc.txt' with
(fieldterminator = '|', rowterminator = '|
')
```

She then noted that the line break just before the last ') is a line break (it represents a line break in the text file). Using '{CR}{LF}' or just {'LF'} did not work for Helen. She also recommended using a fully qualified table name (dbname.owner.tablename) in place of "t_loadtable".

Thanks to the discussion begun by Helen, the list subscribers were given a glimpse into how a large amount of data stored in a text file can be imported into SQL Server and used to modify table data being accessed by ColdFusion applications. Helen's problem isn't an uncommon one, and the discussion gave an excellent real-world example of a solution developers commonly implement, as well as several other possible solutions.

@ SHORWITH@FIGLEAF.COM

<dot.com>
- buyer's guide
- coldfusion forums
- mailing list
- coldfusion jobs
- coldfusion store

<magazine>
- advertise
- authors
- customer service
- editorial board
- subscribe

<content>
- archives
- digital edition
- editorial
- features
- interviews
- product reviews

<conferences>
web services edge conference expo june 24–27 2002
web services edge conference expo oct 1–3 2002

<search cfdj>
Search

<cfdj specials>
Collect All Four!
JDJStore.com $229.99
JDJ, XML, CFDJ, CFAdvisor - ALL Four!!-The Complete Works

<bestsellers>
1. IBM WebSphere Application Server 3.5 Standard Edition $651.99
2. CFDJ and CF Advisor - The Complete Works $529.99
3. AltoWeb Application Platform Release 2.5 $1,540.00
4. ColdFusion 5.0 Server Enterprise $4,599.99
5. ColdFusion 5.0 Server Professional $1,219.99

May **2002**

What's Online

www.sys-con.com/coldfusion

CFDJ Online

Now you can stay ahead of the competition and be the first to know what's happening in the ColdFusion industry! Visit www.sys-con.com/coldfusion every day for the latest news, events, and developments and become the most informed CF professional!

2001 Readers' Choice Awards

Did you miss the March edition of ***CFDJ***, with the results of the Readers' Choice Awards? No reason to fret – our awards are available online. Now you can read all about the awards that have come to be known as the "Oscars of the software industry"! The ***CFDJ*** Readers' Choice Awards are the only awards of their kind. Visit www.sys-con.com/coldfusion to learn which products our readers feel live up to the ***CFDJ*** quality of excellence.

Hot Topics in the CF Discussion Groups

What issues are the most important CF professionals talking about? You'll never know unless you visit the CF Discussion Groups, where you can read about the latest topics, find past subjects, get answers to your technical questions, and post your own thoughts online.

Free Newsletters

Subscribe to FREE newsletters from **SYS-CON**! These informative and interesting newsletters are delivered directly to your e-mail and can be accessed anywhere you can use your computer. The newsletters are available weekly from ***Java Developer's Journal***, twice a month from ***XML-Journal***, ***Web Services Journal***, and ***Wireless Business & Technology***, and once a month from ***CFDJ***, ***WebSphere Developer's Journal***, and ***WebLogic Developer's Journal***. ***Linux Weekly*** is also available. Pick one or choose them all!

ColdFusion Store CD Special

Get the complete works from ***CFDJ*** and ***CF Advisor*** at $30 off the regular price with this special offer from the ***JDJ*** Store.This is the most complete library of exclusive ***CFA*** and ***CFDJ*** articles – over 200 you can keep on hand for research and review. Order today and save!

The Latest at CFBuyersGuide.com

Visit the most-read CF resource on the Internet and take advantage of this FREE listing from CFBuyersGuide.com at www.sys-con.com/coldfusion/wbg/index.cfm. This is an excellent resource for those who want to learn about the newest books, software, tools, and services related to ColdFusion. Features on this site include Web hosting, testing tools, books, e-business software, custom tags, Web development tools, consulting services, education and training, and much more!

CFDJnews

Electric Rain Offers Swift 3D Importer for Flash MX

(Boulder, CO) – Electric Rain, Inc., has announced an improved solution for incorporating Swift 3D vector animations into Macromedia Flash MX projects. The importer, which will be accessed through the Macromedia Flash MX menu system, will recognize a new proprietary Swift 3D file format, resulting in smoother workflow, enhanced functionality, and smaller files.
www.erain.com

EBStor.com Approved as Associate Macromedia Partner

(Manassas, VA) – Pacel Corp. has announced that EBStor.com has received approval as a Macromedia Associate Solution Provider Partner.

Two senior software engineers of EBStor.com recently attained Advanced ColdFusion 5.0 Developer Certified Professional Certificates in order to facilitate elevation to this status.

Associate partners provide an extensive array of custom Web development and design services using Macromedia's business platform technologies. Internet clients rely on these services to build and deploy scalable, secure Web applications.

Benefits of partnering with Macromedia include access to sales engineers, software discounts, and a host of marketing, communications, and e-learning benefits.
www.ebstor.com

In Beta: BlueDragon, New Atlanta App Server Using CFML for Web Publishing

(San Francisco) – A beta version of a Java-based Web application server that implements the ColdFusion Markup Language standard for dynamic Web publishing is available from New Atlanta Communications, LLC. BlueDragon enables the integration of CFML with Java Servlets, JavaServer Pages, and other J2EE technologies.

BlueDragon will be available in a standalone server version and as an add-on to existing J2EE application servers such as IBM WebSphere and BEA WebLogic. It also contains JDBC drivers for all popular database servers, including JTurbo, New Atlanta's J2EE-certified Type 4 JDBC driver for connecting to Microsoft SQL Server.
www.newatlanta.com

Events

CFUN-02
June 15–16, Rockville, MD
www.cfconf.com/cfun-02/

DevCon2002
October 27–30, Lake Buena Vista, FL
www.cfconf.org/

CF_Cruise
The Caribbean, 2003
Departs from Tampa, FL
www.cfconf.org/cf_cruise/

CFDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
ACTIVE PDF	WWW.ACTIVEPDF.COM	949.582.9002	4
CFDYNAMICS	WWW.CFDYNAMICS.COM	800.422.7957	35
COLDFUSION DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	31
DYNAMIC BUYER	WWW.IBM.COM/SMALLBUSINESS/DYNAMICBUYER	800.426.7235	43
EMPIRIX	WWW.EMPIRIX.COM/DOUBLE/CFM	866.228.3781	2
E-ZONE MEDIA	WWW.FUSETALK.COM	866.477.7542	13
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	37
INTERLAND	WWW.INTERLAND.COM	866.253.0827	3
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	52
JAVA DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	23
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	26, 27, 49
MACROMEDIA	WWW.MACROMEDIA.COM/GO/USERGROUPS		9
MACROMEDIA	WWW.MACROMEDIA.COM/GO/MASTERING		47
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CERTIFICATION	877.460.8679	51
NEW ATLANTA	WWW.NEWATLANTA.COM		15
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	39
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	41
RACKSHACK	WWW.RACKSHACK.NET	800.504.SURF	11
SYS-CON MEDIA	WWW.SYS-CON.COM	800.513.7111	44
SYS-CON EVENTS	WWW.SYS-CON.COM	201.802.3057	45
WEB SERVICES EDGE	WWW.SYS-CON.COM	201.802.3069	18, 19
WEB SERVICES JOURNAL	WWW.SYS-CON.COM	800.513.7111	25
WEBLOGIC DEVELOPERS JOURNAL	WWW.WEBLOGICDEVELOPERSJOURNAL.COM		49
XML-JOURNAL	WWW.SYS-CON.COM	800.513.7111	29

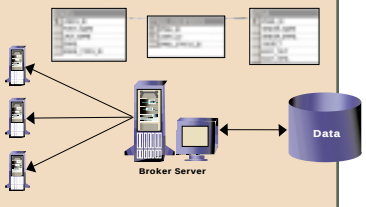
Next Month...

Don't miss the June issue!

ColdFusion MX and XML: Reading XML
By David Gassner
The first in a series describing new functionality in Macromedia ColdFusion MX relating to reading, creating, and processing XML files

Flash Up Your Forms with Components
By Dennis Baldwin
Build a sign-up form with FlashMX components and integrate it with CF5

Beyond CFMAIL – Sending Large Volumes of E-mail
By Tom Peer
A simple way to write your e-mails directly to your mail server's outbound queue folder



Fused, on Level Zero
By Kevin Towes
How to apply ColdFusion programming principles to collectively develop powerful Flash applications

ColdFusion Developer's Journal

Intermedia
www.imtermedia.net