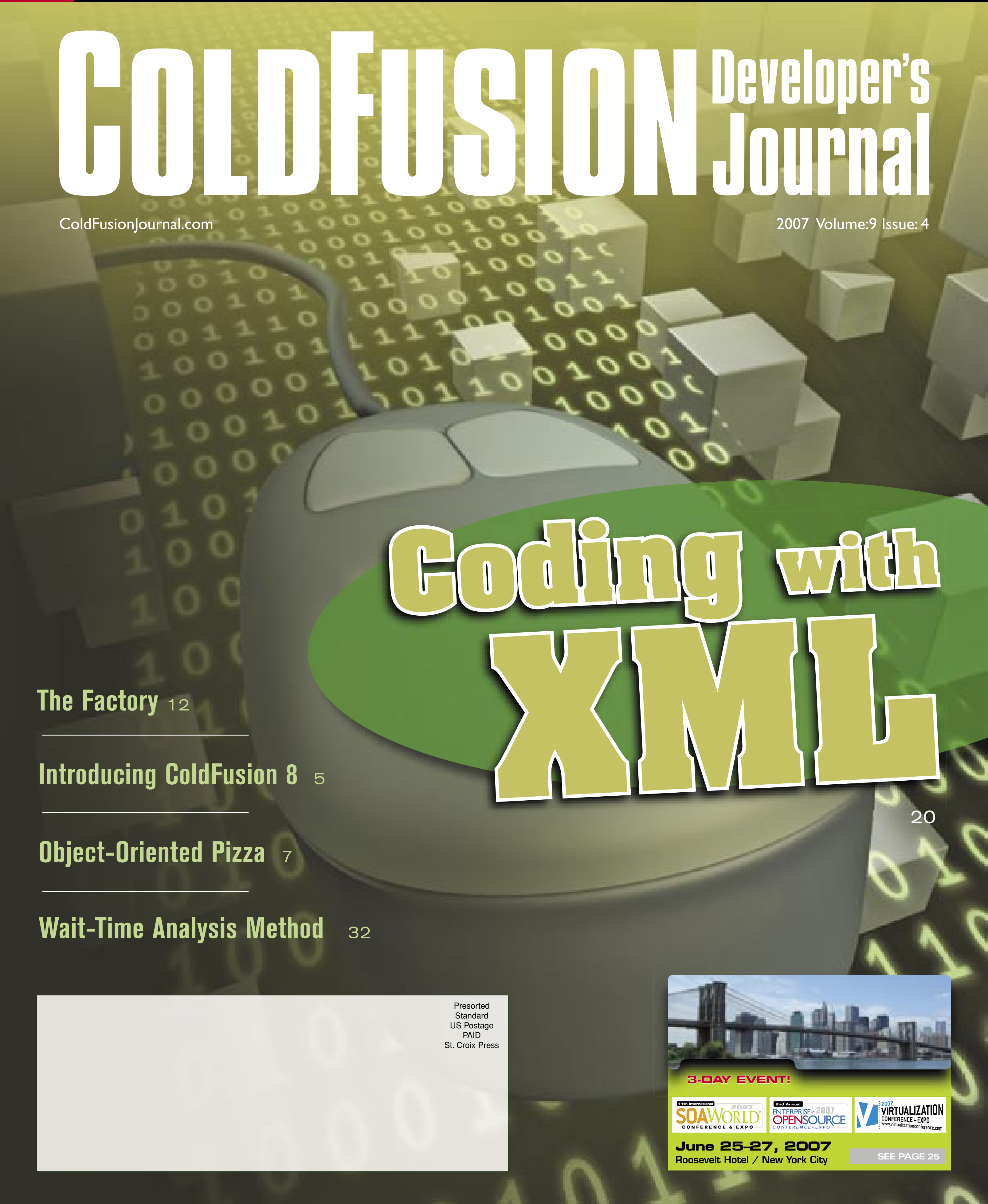
SYS-CON

INTRODUCING COLDFUSION 8: NOW IN PUBLIC BETA PG 5

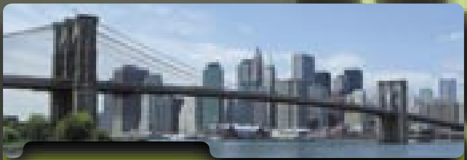
ColdFusion Developer's Journal

ColdFusionJournal.com

2007 Volume:9 Issue: 4



Presorted
Standard
US Postage
PAID
St. Croix Press



3-DAY EVENT!

11th International 2007 SOAWORLD CONFERENCE & EXPO

2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

2007 VIRTUALIZATION CONFERENCE+EXPO www.virtualizationconference.com

June 25-27, 2007
Roosevelt Hotel / New York City

SEE PAGE 25

YUM!
INTERSPEC TACULAR

Adobe® Creative Suite® 2.3

Say hello to the next generation.

It's found in the world's most inspirational places. It's pushing the boundaries of art direction and design. Introducing Adobe® Creative Suite® 2.3. Today's most talented designers and art directors are working smarter and faster because of it. Just ask the masterminds behind INTERspectacular—they rely on the Creative Suite to help bring their ideas into the world. See how they do it at adobe.com/creativemind. It's everything but the idea. Better by Adobe.™

Luis Blanco and Michael Uman,
INTERspectacular

adobe.com/creativemind

Adobe

Relax.

You're hosting with CFDynamics

We at CFDynamics work hard to have the most reliable servers, 99.9% uptime and outstanding customer service! We hope our customers enjoy just a little more "Relax" time because of our efforts.

Try us out today and see why our customers are kicking back a little more often.

- **99.9% Average Uptime!**
- **State-of-the- Art Data Center including HVAC, fire suppression, bandwidth & security**
- **24/7 Network Monitoring**

CFDynamics

The web host that helps you maximize your website

1.866.233.9626 customerservice@cfdynamics.com cfdynamics.com

COLDFUSION Developer's Journal

editorial

editor-in-chief
Simon Horwith simon@sys-con.com

executive editor
Nancy Valentine nancy@sys-con.com

research editor
Bahadir Karuv, PhD bahadir@sys-con.com

production

lead designer
Abraham Addo abraham@sys-con.com

art director
Alex Botero alex@sys-con.com

associate art directors
Louis F. Cuffari louis@sys-con.com
Tami Beatty tami@sys-con.com

editorial offices

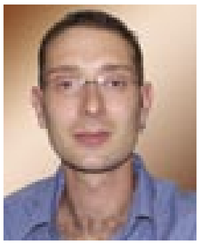
SYS-CON MEDIA
577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677
Telephone: 201 802-3000 Fax: 201 782-9638
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
is published monthly (12 times a year)
by SYS-CON Publications, Inc.

postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
FOR LIST RENTAL INFORMATION:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

For promotional reprints, contact reprint
coordinator Megan Mussa, megan@sys-con.com.
SYS-CON Publications, Inc., reserves the right to
revise, republish and authorize its readers to use
the articles submitted for publication.
All brand and product names used on these pages
are trade names, service marks, or trademarks
of their respective companies.

Introducing ColdFusion 8



By Simon Horwith

On May 30, Adobe released the public beta of ColdFusion 8, which means that we at CFDJ can now begin writing and talking publicly about all the great new features.

I'll add a disclaimer to that statement: CF 8 is currently in public beta and things, though not likely, can change between the public beta and the final release – so the CF 8 specific content in CFDJ will be kept light to null in order to ensure that the content we deliver is accurate for the final release. That said, the features included in CF 8 are unlikely to change between now and the official release, so this month I thought I'd give a high-level overview of some significant new features and why developers and companies should be interested in ColdFusion 8.

The first thing to know about what's new in CF 8 is performance. Not that we didn't have good performance prior to this release, but based on Adobe's test results, developers are likely to see anywhere from a 20–500% performance increase when they run the same code on CF 8 that they're running on CF 7. That, by itself, is a very compelling reason to plan an upgrade/migration to CF 8 when it's released. An additional feature in CF 8 also allows developers to write code that is much more efficient with regards to performance: the ability to programmatically create and control threading (using a new CFTHREAD tag). Oh, but there's so much more.

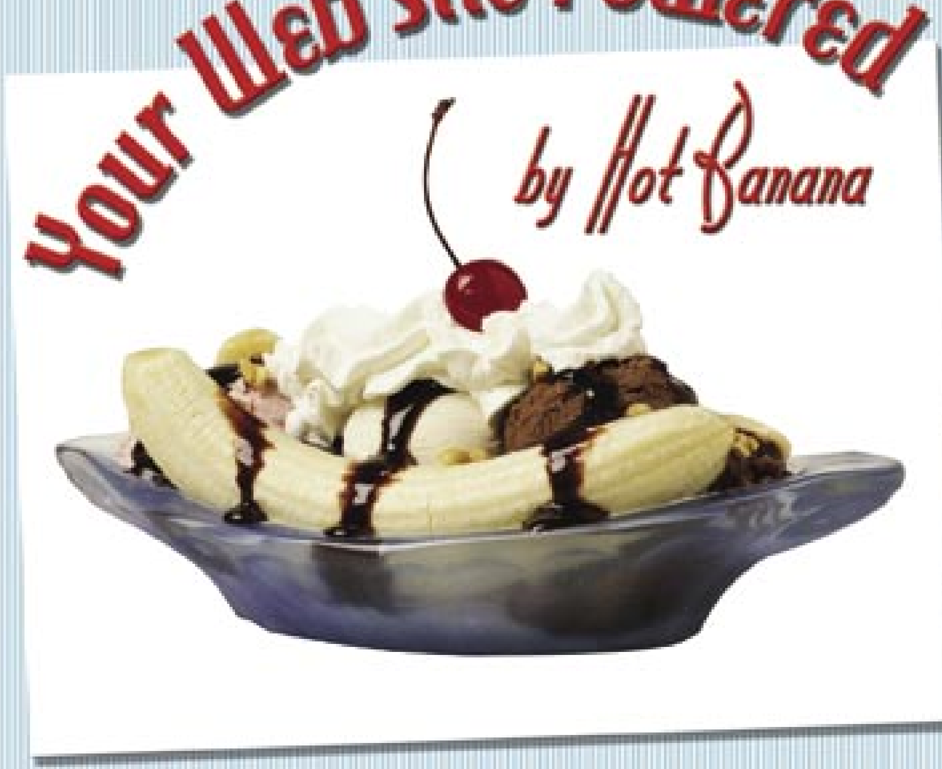
ColdFusion 8 makes developers' lives easier. There are two significant new features that make administration and troubleshooting easier to do, ultimately making developers more productive. The first is a new Eclipse plug-in that allows developers to debug their applications (think "break points," "stepping through code," "watch expressions," etc.). The second feature is a series of CF Administrator enhancements that includes, among other things, a server/multi-server monitor for tracking real-time information, the ability to tune request performance and threading, and the ability to turn the data type-checking on and off in CFCs. An additional nice new CF Admin feature is the ability to define multiple user accounts for the administrator and to restrict CF Admin access by account.
You may be thinking to yourself, "performance and productivity enhancements are great, but what's new?" The actual new features are far too numerous to list here, but I'll mention the ones that I consider extremely significant.

Many new features are all about integration:

- ColdFusion 8 has tighter integration with LiveCycle Data Services (formerly Flex Data Services) and can be installed with it.
- Support for Flex/CF interaction has been improved in several ways, including the ability for CF Assembler CFCs to return queries or structures rather than DTOs and the ability for CF to notify Flex applications of data changes on the server.
- ColdFusion now has support for interaction with Microsoft Exchange Server – you can control connections to the server and programmatically work with Exchange calendars, contacts, and mail services.

– CONTINUED ON PAGE 18

About the Author
Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal *and is the CIO at AboutWeb, LLC, a Washington, DC based company specializing in staff augmentation, consulting, and training. Simon is a Macromedia Certified Master Instructor and is a member of Team Macromedia.*
simon@horwith.com



Hot Banana Web Sites Are Built Fully Loaded!

Why settle for building and managing a *vanilla* Web site when you can savor a full featured Hot Banana Web site?

Hot Banana is a fully-loaded Web content Management System with all the Web site optimization and eMarketing campaign tools your Marketing department craves. And, it's easy-to-use, search engine friendly and fine tuned for peak performance.

So what's the cherry on top? It's ColdFusion of course - you'll love how easy Hot Banana is to implement, integrate and customize.

Schedule your Free Taste Trial Today!

Features...

- 100% browser-based
- Updated RTE
- Powerful DAM
- XHTML & CSS compliant
- Multilingual support
- Granular security
- Flexible workflow
- Press release manager
- Keyword analysis
- Web analytics center
- A/B Testing
- Powerful DAM
- Form builder
- RSS & blogs
- SEO tools
- Content reuse & scheduling
- Lead source tagging & tracking
- Custom metadata
- Event registration
- Email manager

Contact Us Now:
hotbanana.com/cfdj-demo
1.866.296.1803
sales@hotbanana.com

hotbanana®
Web Content Management For Marketing

SYS-CON MEDIA

founder & ceo
Fuat Kircaali fuat@sys-con.com

group publisher
Roger Strukhoff roger@sys-con.com

advertising

senior vp, sales & marketing
Carmen Gonzalez carmen@sys-con.com

advertising sales director
Megan Mussa megan@sys-con.com

associate sales manager
Corinna Melcon corinna@sys-con.com

sys-con events

president, events
Carmen Gonzalez carmen@sys-con.com

events manager
Lauren Orsi lauren@sys-con.com

events associate
Sharmonique Shade sharmonique@sys-con.com

customer relations

circulation service coordinator
Edna Earle Russell edna@sys-con.com
Alicia Nolan alicia@sys-con.com

sys-con.com

vp, information systems
Bruno Decaudin bruno@sys-con.com

information systems consultant
Robert Diamond robert@sys-con.com

web designers
Stephen Kilmurray stephen@sys-con.com
Richard Walter richard@sys-con.com

accounting

financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

subscriptions

Subscribe@sys-con.com
Call 1-888-303-5282

For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $8.99/issue
Domestic: $89.99/yr (12 issues)
Canada/Mexico: $99.99/yr
All other countries $129.99/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. $15 all others

Object-Oriented Pizza

Procedural programming and OO programming buzzwords

By Jeffry Houser

Over the past few years it's gotten a lot harder to write ColdFusion code. That's not to say that CFML has changed significantly. You can still write code today that's much the same as what you wrote five years ago or longer.

However, with the introduction of CFCs in CFMX, ColdFusion started to feel like a real programming language for the first time. CFCs, as you probably know, let us encapsulate data and functionality. With this new tool in the toolbox of CF developers, many slowly started applying advanced programming concepts to their development. This caught on and things have spiraled.

Unfortunately, many CF developers come from a non-programming background. There is all this focus on object orientation, design patterns, inheritance, interfaces, polymorphism, and (insert your own buzzword here) can be quite daunting. Traditional "procedural" development is often considered to be the one true evil. It can be hard to discover where the real value lies.

Applying all this "stuff" to your development often results in longer upfront development time. In theory you'll gain it back in ease of maintenance. Unless you're a better developer than I, the reality is that your first OO attempts will turn into a spaghetti code mess. It may even be worse than many old CF spaghetti code applications, since you're adding a layer of complexity on top of the simple approach. In this article, I'm going to help you try to make sense of the buzz, and give a comparison of procedural approaches and OO approaches along the way.

Create Your First Pizza

What happens when you order a pizza? It's probably something like this:

- Customer comes into a restaurant.
- Customer places order with the waitress.
- Waitress gives order to the cook.
- The cook prepares the pizza and puts it in the oven.
- When the pizza is ready, the cook takes the pizza out of the oven and cuts it.
- The waitress picks up the pizza and delivers it to the customer.

This is a generic enough algorithm. There may be differences and complications depending on a variety of factors, but for the purposes of this article, we can assume that we're building a system to follow that algorithm.

Let's pretend that our pizza restaurant is futuristic, and all the workers are robots. It's our task to write a program that sends out instructions to the waitress robot and cook robot so that they can operate the pizza restaurant. If you were to implement this as a traditional system for processing orders, you might come up with something like this:

- Wait for new customer.
- When customer shows up, waitress gets order from customer.
- Waitress sends order to the cook.

TABLE OF CONTENTS

- The cook prepares the dough, add sauce, cheese, and relevant toppings.
- The cook puts the pizza in the oven to bake.
- Wait until pizza is ready.
- Once ready, the cook takes the pizza out of the oven .
- The cook cuts the pizza cuts the pizza.
- Cook gives pizza to the waitress.
- Waitress gives pizza to the customer.
- Go back to the beginning and wait for a new customer...

This algorithm will definitely get the job done and tell our robots how to deal with the orders. For purposes of this example, we're going to assume that our system is so efficient at processing orders that we don't have to worry about waiting on multiple customers at once. Or if you don't buy that, pretend business is really bad.

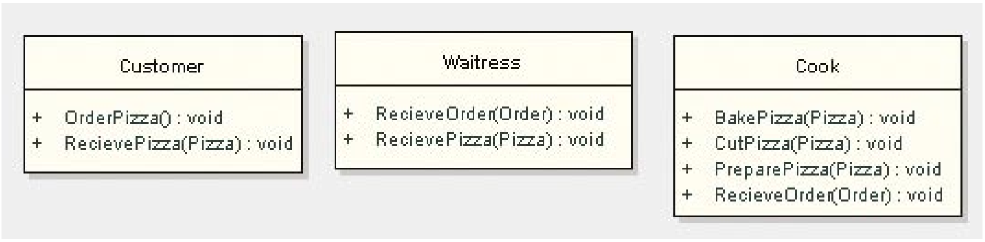

Customer
+ OrderPizza() : void
+ RecievePizza(Pizza) : void
Waitress
+ RecieveOrder(Order) : void
+ RecievePizza(Pizza) : void
Cook
+ BakePizza(Pizza) : void
+ CutPizza(Pizza) : void
+ PreparePizza(Pizza) : void
+ RecieveOrder(Order) : void

Figure 1: The pizza store object model

Elements of the System

Based on the descriptions of the algorithm and the first programming approach, can you identify a few of the elements in this system? Here are a few of the important elements I noticed:

- **Customer:** The customer is the guy who wants the pizza. He doesn't do anything except provide the order to the waitress robot.
- **Waitress:** This is the robot that takes your order.
- **Pizza:** The pizza is an element of the system, with cheese, sauce, and all the various toppings, especially pepperoni.
- **Cook:** This is the robot that prepares and cooks your order.

These elements come into play as we explore different implementation options for this system.

The First Implementation Attempt

Back in the dark ages of CF development, you might have implemented the system like this:

```
<cfif IsDefined("newOrder")>
   <cf_PreparePizza orderdetails=" newOrder" returnVariable="newPizza">
   <cf_BakePizza pizza=" newPizza" returnVariable=" newPizza">
   <cf_CutPizza pizza=" newPizza" returnVariable=" newPizza">
   <cf_DeliverPizza pizza=" newPizza">
 </cfif>
```

Our initial algorithm had a "wait until order" clause, which doesn't apply very well to CF development. So, this code assumes that every order is provided as a page-submit, and we process it on that page. Prior to CFCs, the functionality could only be encapsulated via custom tags or UDFs. This example uses custom tags for preparing the pizza, baking the pizza, cutting the pizza, and delivering the pizza.

With this approach, functionality is encapsulated into custom tags. Data isn't encapsulated. We have two data structures in our main template, newOrder and newPizza. These data structures are passed in and out of the custom tags. This is a procedural approach to the problem. We broke things up into separate chunks, and are just telling the computer the order with which to run our code. I'm going to leave the implementation details of the data structures and custom tags to your imagination. I haven't had a client ask me to control robots from a ColdFusion program yet.

So, Let's Put Some CFCs into the Mix

This whole example is about pizza. Most of the functions we're doing are being done on the pizza. So, we can create a Pizza CFC to handle a lot of this functionality. A few methods to put in the CFC are Bake, Prepare, Cut, and Deliver. Instance variables in the CFC might be price, ingredients, and the order information. Instead of executing multiple custom tags and passing data back and forth, we pass the data into the pizza component, and then merely call the component's methods (or functions) to act on the data. A sample of this approach can be seen in Listing 1. As with the custom tags, I didn't flesh out the code of the CFC.

Is the CFC in this example an object? Technically it meets the definition of an object from the OO paradigm. It also meets the definition of an Abstract Data Type (ADTs) from the procedural paradigm. For all intents and purposes objects and ADTs are the same. They are containers for data that also abstract functionality to perform actions on the data. Today everyone calls them objects because that's a buzzier word.

Add Some Object Orientation

Object-oriented programming is supposed to build a model that mimics the way things really operate in the real world. In procedural programming, your model is built to make it easy for the computer to process. I once heard it said that procedural is requesting the data and doing things to it. Using this as a basis, Listing 2 is very procedural. It's not mimicking the real world; it's breaking things up into small chunks to send to the computer processor for processing.

A pizza doesn't know how to deliver itself to the customer. It doesn't know how to prepare itself. It does not know how to bake itself, or cut itself into slices. In an OO model, this functionality wouldn't be put on a pizza object. An OO is geared towards telling your "things" how to act on the data they already have. Next we'll modify the example to be a bit more OO.

The pizza doesn't actually do anything in the real world. The robots or customers do the real tasks. If we were to create an object for it, it would just be a data container to pass around to objects. Here are a few of the actions that occur in the system:

- **OrderPizza:** This would be something that the customer does.
- **PreparePizza:** This is something that the cook does.
- **BakePizza:** This is also something done by the cook.
- **Cut Pizza:** The cook cuts the pizza.
- **Receive Order:** When the customer orders the pizza, the waitress gets the order. When the waitress gets the order, she needs to pass it to the cook. As such both the waitress and the cook may have a method for receiving the order.
- **Receive Pizza:** When the cook is done cutting the pizza, he needs to give it to the waitress. The waitress has to respond to the receive pizza method. When the waitress gets the pizza, it must be delivered to the customer, and as such the customer must also respond to the Receive Pizza method.

We come out of this description with a customer object, a cook object, and a waitress object. You can see these in Figure 1. Most likely a full-fledged application would also include a pizza object and an order object, used as data containers as arguments to the methods.

With all these objects, our code changes quite a bit. This line will start the order process:

```
Customer.OrderPizza();
```

This assumes that a customer object is already created, of course. The customer OrderPizza function would look like this:

```
<cffunction name="OrderPizza">
 <cfset Waitress.RecieveOrder(MyOrder)>
</cffunction>
```

The Waitresses ReceiveOrder method would look like this:

```
<cffunction name="OrderPizza">
 <cfargument name="Order" type="order">
 <cfset Cook.RecieveOrder(MyOrder)>
</cffunction>
```

The cook's ReceiveOrder method would look like Listing 2. The cook does a lot of work in this process compared to the customer or waitress. The waitress gets the pizza back from the cook:

```
<cffunction name=" RecievePizza">
 <cfargument name="pizza" type="Pizza">
 <cfset customer. RecievePizza (arguments.pizza)>
</cffunction>
```

The customer gets the pizza from the waitress using this method:

```
<cffunction name=" RecievePizza">
 <cfargument name="pizza" type="Pizza">
 <cfset customer. Devour (arguments.pizza)>
</cffunction>
```

Instead of one CFC with lots of data and functionality, we have a lot of CFCs with very little data and functionality. The order is passed around, down the chain from the customer to the waitress to the cook. The cook processes it and sends a pizza back to the customer.

Conclusion

A lot of ColdFusion programmers started programming without using any encapsulation at all. Business logic and display code was mixed in a single template. This was partially due to limitations within the language of ColdFusion. When I talk to them they refer to this old school CF method of programming as procedural, even though it had very little to do with procedural programming. I hope this article enlightened you to a few of the differences between procedural programming and object-oriented programming.

About the Author

Jeffry Houser has been working with computers for over 20 years and in Web development for over 8 years. He owns a consulting company and has authored three separate books on ColdFusion, most recently ColdFusion MX: The Complete Reference (McGraw-Hill Osborne Media).

jeff@instantcoldfusion.com

Listing 1

```
<cfif IsDefined("newOrder")>
 <cfscript>
 newPizza = Createobject('component','pizza');
 newPizza.init(newOrder);
 newPizza.prepare();
 newPizza.Bake();
 newPizza.cut();
 newpizza.deliver();
</cfscript>
```

Listing 2

```
<cffunction name="OrderPizza">
 <cfargument name="Order" type="order">
 <cfset var pizza = CreateObject('component','pizza')>
 <cfset variables.order = arguments.order>
 <cfset PreparePizza(pizza)>
 <cfset BakePizza(pizza)>
 <cfset CutPizza(pizza)>
 <cfset Waitress.RecievePizza(pizza)>
</cffunction>
```

Download the Code...
Go to http://coldfusion.sys-con.com

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.

HostMySite.com

WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL

The Factory

Productivity is back

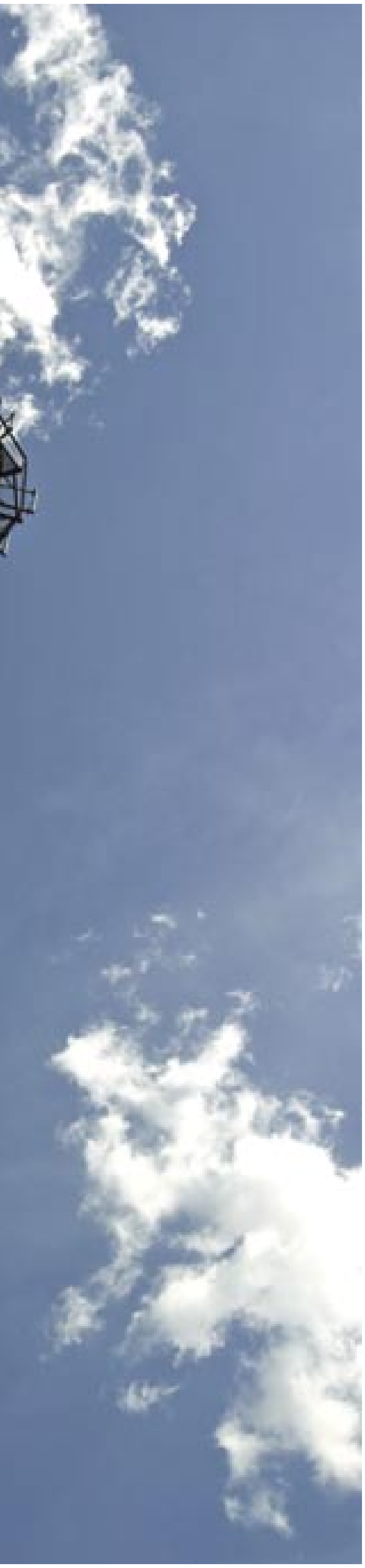
By Michael Givens

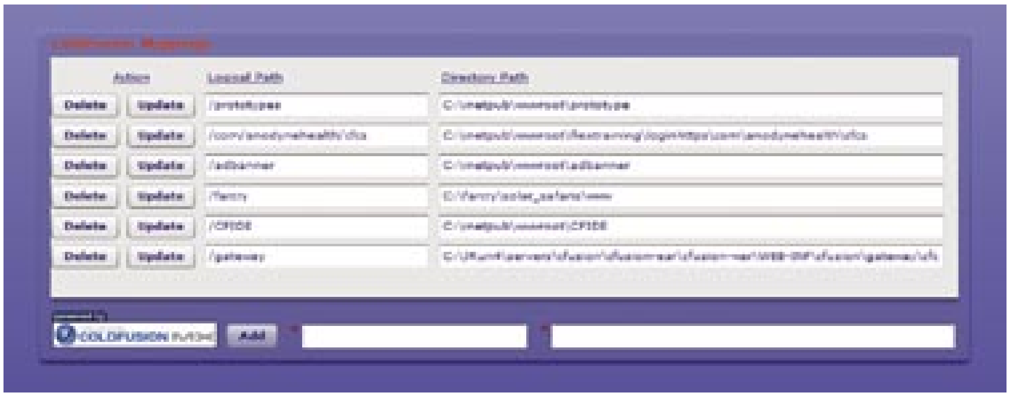
Whether it's in factories or workshops, in mines or forests, in offices or homes, or even in our sacred ColdFusion Administrator, anything that impairs the efficiency of workers is always worthy of improvements in the process or changing the way of doing things.

Levering ColdFusion ServiceFactory to get work done is one example. Figure 1 is a screenshot of an example Flex-ColdFusion hybrid application that allows manipulation of ColdFusion mapping without having (or even needing) access to the ColdFusion Administrator.

Look at the code. The Flex UI code is shown in Listing 1. The initialize attribute in the main MXML file calls the getIP() function during initialization of the Flex application. This function triggers a RemoteObject invocation of the roMapping.getVisitorIP() method. The getVisitorIP() method is the first function shown in Listing 2. (Listing 2 can be downloaded from the online version of this article at http://coldfusion.sys-con.com.) By examining the IP address of the user, we can programmatically allow Flex to decide if the CFC mapping utility is visible. This is a way to limit CFC mapping changes to users on your internal network. For instance, in the Flex code, the getVisitorIP_handler has a conditional that ensures that only IP addresses on an internal network that start with 192.168 can call the RemoteObject's method, roMapping.getMappings(). Of course, you can change that to whatever your internal network's IP addresses start with once you download the code. Speaking of the code, the full source code is available at http://labs.insideflex.com/flextraining/cfmapping/bin/srcview/index.html.

Provided the user is coming from the appropriate IP address, the RemoteObject method, roMapping.getMappings(), instantiates a factory object and calls the ColdFusion servers ServiceFactory getMappings method:

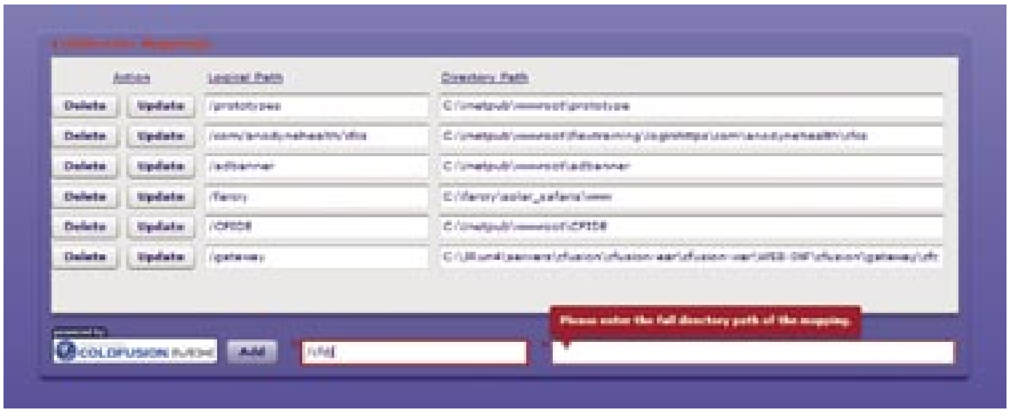




ColdFusion Mappings

Action		Logical Path	Directory Path
Delete	Update	/prototypes	C:\inetpub\wwwroot\prototype
Delete	Update	/com/anodynehealth/cfcs	C:\inetpub\wwwroot\flextraining\loginhttps\com\anodynehealth\cfcs
Delete	Update	/adbanner	C:\inetpub\wwwroot\adbanner
Delete	Update	/fanny	C:\fanny\solar_safaris\www
Delete	Update	/CFIDE	C:\inetpub\wwwroot\CFIDE
Delete	Update	/gateway	C:\JRun4\servers\cfusion\cfusion-ear\cfusion-war\WEB-INF\cfusion\gateway\cfc

Figure 1 ColdFusion Mappings Flex UI



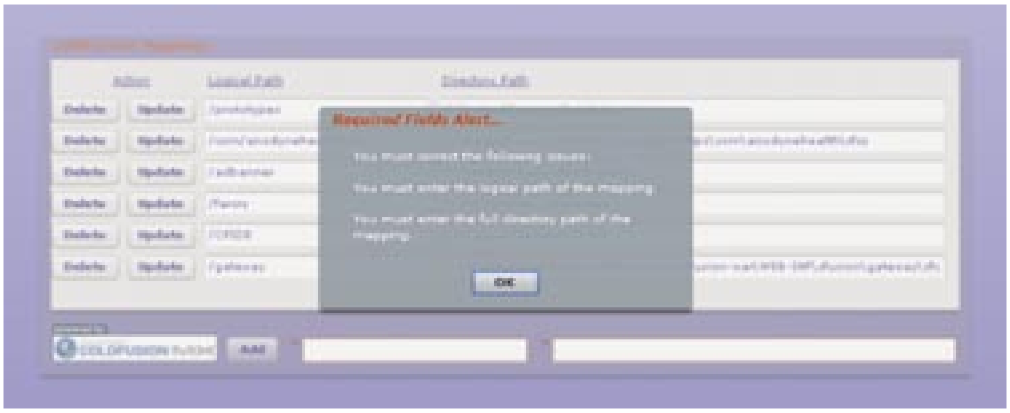
ColdFusion Mappings

Action		Logical Path	Directory Path
Delete	Update	/prototypes	C:\inetpub\wwwroot\prototype
Delete	Update	/com/anodynehealth/cfcs	C:\inetpub\wwwroot\flextraining\loginhttps\com\anodynehealth\cfcs
Delete	Update	/adbanner	C:\inetpub\wwwroot\adbanner
Delete	Update	/fanny	C:\fanny\solar_safaris\www
Delete	Update	/CFIDE	C:\inetpub\wwwroot\CFIDE
Delete	Update	/gateway	C:\JRun4\servers\cfusion\cfusion-ear\cfusion-war\WEB-INF\cfusion\gateway\cfc

Please enter the full directory path of the mapping.

/cfd

Figure 2 Required fields



Required Fields Alert...

You must correct the following issues:

You must enter the logical path of the mapping

You must enter the full directory path of the mapping

Figure 3 Required fields alert

```
<cfset factory=createObject("java","coldfusion.server.ServiceFactory")>
<cfset mappings = factory.runtimeService.getMappings()>
```

A structure of arrays stores the mapping information returned from the ServiceFactory and is returned to the Flex UI as a structure:

```
<cfset stMapping = structNew()>
<cfset aLogicalPath = ArrayNew(1)>
<cfset aDirectoryPath = ArrayNew(1)>
<cfloop collection="#mappings#" item="thismapping">
    <cfscript>
        ArrayAppend(aLogicalPath, thismapping);
        ArrayAppend(aDirectoryPath, mappings[thismapping]);
    </cfscript>
</cfloop>
<cfscript>
    StructInsert(stMapping, "logicalpath", aLogicalPath);
    StructInsert(stMapping, "directorypath", aDirectoryPath);
</cfscript>
<cfreturn stMapping/>
```

Methods for creating, updating, and deleting mappings are also included in the CFC.

Warning

Updating or deleting ColdFusion mappings should be done with caution, especially on shared ColdFusion servers.

Flex Form validation is used and shown in Figures 2 and 3.

For additional information, my Flex Cookbook post, “Get a Client IP Address with a RemoteObject Call,” demonstrates limiting access to Flex content on your internal network or by selected IP addresses. It uses the ColdFusion CFML CGI.Remote_Addr variable passed to the Flex UI via AMF. http://www.adobe.com/cfusion/communityengine/index.cfm?event=showdetails&postId=3462&productId=2.

I hope you found this article useful. If you have any follow-up questions, feel free to contact me.

About the Author

Michael Givens is the CTO of U Saw It Enterprises, a Web consulting firm based in Marietta, GA. As an experienced Web technology specialist, he is willing to shift gears at a moment’s notice to the client’s technology of choice. He is both an Adobe Community Expert and an Adobe Corporate Champion known to share his experience and evangelism of all things Adobe. Certified in both ColdFusion 5 and as an Advanced CFMX Developer, he has been using ColdFusion since the days of Allaire Spectra.

Blogging at http://www.flexination.info/
info@webmxml.com

Listing 1 (cfmapping.mxml)

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    pageTitle="Flex &amp; CFC Mappings" initialize="{getIP()}"
    layout="vertical" backgroundColor="#6778EB"
    viewSourceURL="srcview/index.html">

    <mx:RemoteObject id="roMapping"
        destination="ColdFusion"
        source="flextraining.cfmapping.cfc.mapping"
        showBusyCursor="true">
    <mx:method name="getVisitorIP"
        result="getVisitorIP_handler(event.result)" fault="ro_
fault(event)"/>
    <mx:method name="getMappings"
        result="getMappings_handler(event.result)" fault="ro_
fault(event)"/>
    <mx:method name="createMapping"
        result="createMapping_handler(event.result)" fault="ro_
fault(event)"/>
    <mx:method name="updateMappings"
        result="updateMappings_handler(event.result)" fault="ro_
fault(event)"/>
    <mx:method name="deleteMapping"
        result="deleteMapping_handler(event.result)" fault="ro_
fault(event)"/>
    </mx:RemoteObject>

    <mx:Style>
        Panel {
            borderColor: #666666;
            borderAlpha: 0.4;
            roundedBottomCorners: true;
            headerHeight: 22;
            backgroundAlpha: 1;
            backgroundColor: #E4E4E4;
            titleStyleName: "mypanelTitle";
        }

        .mypanelTitle {
            color: #ff3300;
            textAlign: left;
            fontSize: 12;
            fontWeight: bold;
            fontStyle: italic;
            paddingLeft: 20;
        }
    </mx:Style>

    <mx:Script>
    <![CDATA[
        import mx.rpc.events.ResultEvent;
        import mx.rpc.events.FaultEvent;
        import flash.events.Event;
        import mx.controls.Alert;
        import mx.utils.ObjectUtil;
        import mx.collections.ArrayCollection;
        import mx.events.CloseEvent;
        import mx.events.ValidationResultEvent;

        [Bindable] private var oMappings:Object = new Object();
        [Bindable] private var arcMapLPath:ArrayCollection = new
                               ArrayCollection();
        [Bindable] private var arcMapDPath:ArrayCollection = new
                               ArrayCollection();
        [Bindable] private var aryPath:Array = new Array();
        private var blnSuccess:Boolean = false;
        private var selItm:String = new String();
        private var vResult:ValidationResultEvent;
        private var msg:String = "";
        private var count:Number = 0;
        private var msgprefix:String = "";
        private var sIP:String;

        private function getIP():void {
            roMapping.getVisitorIP();
        }

        private function getVisitorIP_handler(result:Object):void {
            sIP = result.toString();
            if (sIP.substr(0,8)=="192.168.") {
                hbxAddMap.enabled = true;
                roMapping.getMappings()
            } else {
                Alert.show("Sorry, access to this page is
                    restricted...");
            }
        }

        private function getMappings_handler(result:Object):void {
            oMappings = result;
            arcMapLPath.removeAll();
            arcMapDPath.removeAll();
            for (var i:uint=0; i<oMappings.logicalpath.length; i++) {
                arcMapLPath.addItem(oMappings.logicalpath[i]);
            }
            for (i=0; i<oMappings.directorypath.length; i++) {
```

Adobe | Solution PARTNER

INTERGRAL information solutions

<cf_essentials>

Indispensable CFMX tools.

Get an extra 15% discount on the Standard Editions of FusionDebug with the following discount code: "GOT2DEBUG" offer ends soon.

FusionReactor

CFMX / J2EE Server, database and application monitoring

from $249

FusionDebug

Interactive Debugging for ColdFusion MX

from $105

fusion www.fusion-reactor.com

```
        arcMapDPath.addItem(oMappings.directorypath[i]);
    }
}

private function createMapping(selLItm:String, selDItm:String):
         void {
    roMapping.createMapping(selLItm, selDItm);
}

private function createMapping_handler(result:Boolean):void {
    blnSuccess = result;
    if (blnSuccess) {
        //Alert.show(blnSuccess.toString());
       tiLPath.text = “”;
       tiDPath.text = “”;
       roMapping.getMappings();
    }
}

private function updateMappings(selLItm:String):void {
    for (var i:uint=0; i<oMappings.logicalpath.length; i++) {
        if (selLItm==oMappings.logicalpath[i]) {
            var selDItm:String = oMappings.directorypath[i];
        }
    }
    Alert.show(selLItm + “ “ + selDItm);
    roMapping.updateMappings(selLItm, selDItm);
}

private function updateMappings_handler(result:Boolean):void {
    blnSuccess = result;
    if (blnSuccess) {
        //Alert.show(blnSuccess.toString());
        tiLPath.text = “”;
        tiDPath.text = “”;
        roMapping.getMappings();
    }
}

private function validateForm(selLItm:String, selDItm:String):
         void {
    msg = “”;
    count = 0;
    // Validate the logical path.
    vResult = vLPath.validate();
    // If the logical path is invalid.
    if (vResult.type==ValidationResultEvent.INVALID) {
        msg = “You must enter the logical path of the
                mapping.\n\n”;
        count++;
    }
    // Validate the directory path.
    vResult = vDPath.validate();
    // If the directory path is invalid.
    if (vResult.type==ValidationResultEvent.INVALID) {
        msg = msg + “You must enter the full directory path of
                    the mapping.\n\n”;
        count++;
    }
    if (msg!=””) {
        if (count>1) {
            msgprefix = “You must correct the following
                issues:”;
        }
        mx.controls.Alert.show(msgprefix + “\n\n” + msg,
                        “Required Fields Alert...”);
        return;
    } else {
            createMapping(selLItm, selDItm);
    }
}

private function deleteMapping(selLItm:String):void {
            selItm = selLItm;
            Alert.show(“Do you want to delete this mapping?\n\n” +
                        selItm + “\n\n”,
            “Delete Mapping”, 3, this, alertClickHandler);
        }


        private function alertClickHandler(event:CloseEvent):void {
            if (event.detail==Alert.YES) {
                roMapping.deleteMapping(selItm);
            }
        }

        private function deleteMapping_handler(result:Boolean):void {
            blnSuccess = result;
            if (blnSuccess) {
                //Alert.show(blnSuccess.toString());
                roMapping.getMappings();
            }
        }

        private function ro_fault(event:FaultEvent):void {
            // dump error message
            Alert.show(ObjectUtil.toString(event.fault));
        }
    ]]>
    </mx:Script>

    <!-- Define validators. -->
    <mx:StringValidator id=”vLPath” source=”{tiLPath}” property=”text”
        requiredFieldError=”Please enter the logical path of the
                mapping.”/>
    <mx:StringValidator id=”vDPath” source=”{tiDPath}” property=”text”
        requiredFieldError=”Please enter the full directory path of the
                mapping.”/>

    <mx:Panel id=”pnl” title=”ColdFusion Mappings” color=”navy”
                layout=”absolute”
        paddingLeft=”10” paddingRight=”10” width=”98%” height=”98%”>
        <mx:HBox y=”10”>
            <mx:Spacer width=”45”/>
            <mx:Label text=”Action” textDecoration=”underline”
                    width=”50”/>
            <mx:Spacer width=”18”/>
            <mx:Label text=”Logical Path” textDecoration=”underline”
                    width=”200”/>
            <mx:Label text=”Directory Path” textDecoration=”underline”
/>
        </mx:HBox>
        <mx:HBox y=”34”>
            <mx:VBox>
                <mx:Repeater id=”rptDeleteBtn” dataProvider=”
                    {arcMapLPath}”>
                    <mx:Button label=”Delete”
                        click=”deleteMapping(event.currentTarget.
                            getRepeaterItem())”
                        width=”60”/>
                </mx:Repeater>
            </mx:VBox>
            <mx:VBox>
                <mx:Repeater id=”rptUpdateBtn” dataProvider=”
                    {arcMapLPath}”>
                    <mx:Button label=”Update”
                        click=”updateMappings(event.currentTarget.
                                getRepeaterItem())”
                        width=”60”/>
                </mx:Repeater>
            </mx:VBox>
            <mx:VBox>
                <mx:Repeater id=”rptMapLPath” dataProvider=”
                        {arcMapLPath}”>
                    <mx:TextInput text=”{rptMapLPath.currentItem}”
                        width=”200”/>
```

cfunited
Adobe
New Atlanta COMMUNICATIONS
CFDynamics
Universal Mind
SEEFUSION
ColdFusion Developer's Journal
cfunited
June 27- 30, 2007
Washington D.C area
Apollo
Step through your CFML code with FusionDebug
Using Structures with CFCs
ColdFusion Application Security
CSS - Back to Basics
Flex and Ajax: perfect match
Object Oriented Flex
Testing, Monitoring & Tuning CF w/ Open Source tools
Testing for Accessibility
Coding with XML
Flex 2 for ColdFusion Developers
Flex Data Services & CF Scorpio – Faster, Better & Cooler!
AJAX Development with ColdFusion Frameworks
Integrating Spry and ColdFusion
Beyond Basic SQL For CF
Working with RSS
Programing with ColdSpring
Creating and Consuming WebServices
TeraTech
HostMySite.com
AboutWeb
Paper | Thin
ShadoCMS with ZoomFlex
Fig Leaf Software
Your seat

the premier coldfusion technical conference

Don't miss out on the Premier ColdFusion conference, CFUNITED-07 where you can learn from over 50 expert speakers and authors such as Ben Forta, Michael Smith, Simon Horwith, Hal Helms, Ray Camden, Charlie Arehart. Four days of sessions, networking and learning with your programming peers from around the world. Guarantee yourself a seat and register today.

"CFUNITED is a great way to not only network with fellow ColdFusion afficianados, but also to learn new techniques and methods to help your career or business. You also can get inside scooops on developments in ColdFusion and learn best practices and much more."
-Angela T , Attendee

"It is run by developers for developers. The sessions are based on real world experience and real user case studies."
-Rich P , Attendee

"CFUNITED is a great opportunity to launch your career into the next level, allowing developers to learn new tips, tricks and techniques, all while bonding with fellow ColdFusion users many of us only know by name."
-Constanty D , Attendee

"Learning more, networking more, getting NEW ideas...and now, to support CF's continued existence."
-Howard P, Attendee

TeraTech Inc.
405 East Gude Dr, Ste 207
Rockville, MD 20850
301.424.3903

www.cfunited.com

```
            </mx:Repeater>
        </mx:VBox>
        <mx:VBox>
            <mx:Repeater id="rptMapDPath" dataProvider="
                    {arcMapDPath}">
                <mx:TextInput text="{rptMapDPath.currentItem}"
                    width="450"/>
            </mx:Repeater>
        </mx:VBox>
    </mx:HBox>
    <mx:Spacer height="5"/>
    <mx:HRule width="100%"/>
    <mx:Spacer height="5"/>
    <mx:ControlBar>
        <mx:HBox id="hbxAddMap" enabled="false" left="0" top="0"
            bottom="390" right="-458">
            <mx:Image source="assets/images/poweredbycfmx.gif"/>
            <mx:VBox>
                <mx:Spacer height="6"/>
                <mx:HBox>
                    <mx:Button label="Add"
                        click="validateForm(tiLPath.text,
                            tiDPath.text)" width="45"/>
                    <mx:FormItem id="fitLPath" required="true">
                        <mx:TextInput id="tiLPath" width="200"/>
                    </mx:FormItem>
                    <mx:FormItem id="fitDPath" required="true">
                        <mx:TextInput id="tiDPath" width="
                            {pnl.width - 468}"/>
                    </mx:FormItem>
                </mx:HBox>
            </mx:VBox>
        </mx:HBox>
    </mx:ControlBar>
</mx:Panel>
</mx:Application>
```

Download the Code...
Go to http://coldfusion.sys-con.com

Introducing ColdFusion 8

– CONTINUED FROM PAGE 5

- ColdFusion 8 has a new Event Gateway that allows CF to interact with Flash Media Servers and their Shared Objects.
- ColdFusion 8 has native support for .NET – you can now work with .NET assemblies as you would other objects.

In addition to integration with other products, ColdFusion 8 has new and enhanced support for other file formats and technologies:

- ColdFusion 8 has a ton of AJAX support.
  - Developers can programmatically create an AJAX Proxy that allows AJAX client applications to interact with CFC methods.
  - CF now has native support for JSON serialization/deserialization for passing data back and forth with JavaScript applications.
  - There are dozens of enhancements to CFFORM and CFFORM controls that allow developers to easily build AJAX client applications and more rich user interfaces.
- CF 8 introduces native support for publishing and consuming content in RSS 2.0 and Atom RSS feed formats.
- There is native support for creating and manipulating images in CFML with ColdFusion 8.
- ColdFusion 8 now supports the ZIP file format – you can create, open, and manipulate ZIP and JAR files using CFML.
- ColdFusion 8 includes a lot of improved PDF support and developers now have the ability to programmatically manipulate existing PDF documents and PDF Forms

Two very significant enhancements were made to ColdFusion Components. One new feature is support for interfaces – a feature that is extremely useful for architects and one that takes CFCs one more step toward having complete support for all of the features you'd expect in an object-oriented language. The other CFC enhancement, one that for me is ranked in the top three of all the new features in CF 8, is that CFCs are now serializable. This means that ColdFusion Component instances in the session scope will replicate between servers in a cluster – a feature essential when developing true enterprise ColdFusion applications.

There are numerous other language and feature enhancements in ColdFusion 8 including a much enhanced report builder and report functionality, enhancements to the Application.cfc-based application framework, the ability to create online presentations (slide shows), new JDBC drivers, a new Verity version, new encryption functionality, JBoss support, support for JDK 1.6, Apache Derby database support, database driver logging support, the ability to programmatically access information about data sources, and CFML language enhancements that include implicit array and structure creation and JavaScript operator support in expressions.

There are many additional features that I didn't mention and so much more that I could say about each and every one of the features I've described here, but I've tried to touch on all of the significant enhancements and additions that ColdFusion 8 brings to an organization. If you haven't downloaded the ColdFusion 8 beta or looked at the documentation, both are available on the Adobe Labs site at http://labs.adobe.com/technologies/coldfusion8/. In addition, in upcoming months keep your eye on upcoming issues of CFDJ, the Adobe ColdFusion, Labs, and DevNet sites, and developer blogs for more on the benefits of moving to ColdFusion 8..

Have you ever wanted to "see" what's going on inside your ColdFusion servers and applications?

Now you can.

SEEFUSION™

SeeFusion is a unique, powerful tool for real-time monitoring of your ColdFusion applications and servers, providing a wide variety of metrics on request times, database queries, memory utilization, code performance and more!

www.seefusion.com

SeeFusion is a product of Webapper Services, LLC
Development, Consulting, Products, Training
www.webapper.com

Coding with XML

Reading and creating well-formed XML

By Andrew Schwabe

As a ColdFusion developer, hopefully by now you have heard at least a little about XML (eXtensible Markup Language). Wikipedia defines XML as a “general-purpose markup language” designed to “facilitate the sharing of data” and also designed to be “relatively human-legible.”

If you need primer on the basics of what XML is and the rules of how it’s structured, a good place to start is http://en.wikipedia.org/wiki/XML.

Nowadays we see XML in regular daily use. RSS Feeds, podcasts, and Web Services are all XML, or use XML to exchange information. Even HTML is related and basically a type of XML (loosely, but that’s a long discussion for another time). It’s no coincidence that we see XML in use for mass-distributed or syndicated content – XML is designed for a few key purposes: 1) as a format that can easily be interpreted by software, and 2) containing only the data and its structure, with no formatting.

A complete collection of data as shown in Listing 1 is called an XML Document. XML Documents should be “well-formed” – meaning that they conform to all the rules of XML syntax, with no unclosed tags. (Note: The “<street2/>” element, which is empty, uses a trailing slash denoting that it is an empty element. This is essentially the same thing as typing out “<street2></street2>.”)

XML Documents that are meant to comply to a standard format will have an XML Schema definition, or a DTD (Document Type Definition), which defines a tight set of rules as to which content and tags can and can’t be contained in an XML Document of that type. XML Documents are considered “valid” if they conform to their DTD, although for individual sites, many XML documents will not have a DTD and having one is not required.

So how do ColdFusion programmers use and work with

XML? ColdFusion is supposed to empower programmers to build sites and Web applications faster, right? So what tools are available to help us get the job done faster when it comes to XML? That's the primary focus of this article. We'll focus on the building blocks of how ColdFusion works with XML and then show how to use some of the other useful functions.

The built-in functions in CFML that we'll be covering are Xml Parse(), XmlNew(), XmlElemNew(), XmlSearch, and XmlValidate().

Reading XML Documents

Reading and using data from an XML document is the first, and easiest, lesson. The key to coding a project that uses XML is understanding the mechanics of XML. In the past (pre-XML days) if we wanted to pull data from inside text (such as a first name from the data in Listing 1), we'd have needed to search the string and use mid() functions to extract what we want. This is a tedious process. Now with XML we can translate a well-formed XML document into something ColdFusion can understand. We use the XmlParse() function to convert this from structured text into an XML Object, which is basically a collection of structures and arrays.

Take a look at Listing 2 and Listing 3. In this example, we read details of an order in XML format and display the results, formatted with HTML.

On line 1, we read the contents of the XML document into a string named strXmlOrder. In this example we are reading a file from the local file system, however, you could have just as easily read this data from a database or a remote server using CFHTTP or a Web Service call.

On line 5 we parse the XML using XmlParse(). This function transforms the XML from its textual form into a ColdFusion XML Object called XmlOrder. Try doing a <cfdump var="#XmlOrder#"> here and take a look at the results. What you'll see is that the XmlOrder object is an organized collection of structures and arrays.

The XML Object is organized as a tree of XML Elements (an element is an opening and closing tag and any of its contents – including other elements) and attributes (values that are part of the opening tag).

Look again at Listing 2 and let's put things into perspective: <order> is the xmlroot, and also happens to be an XML Element because it has an opening and closing tag. The <order ... > element has one attribute called "id," which has a value of "E10645."

The <customer ...>, <items>, and each of the <item> tags are also XML Elements. "Fname," "lname," and "memberid" are attributes of the <customer ...> element and so on; you get the idea.

Under the xmlroot element, all other "children" elements are organized in an array called XmlChildren. So <customer...> and <items> are both children of <order>, and as such are stored in the XmlChildren array, as element 1 and 2 accordingly.

The Items element has three children elements (the three individual items), and each of those items has three children elements (name, quantity, and price).

Now that you have a feel for how this data is organized, let's go back to Listing 3.

On line 11 we display the order id. Since "id" is an attribute of the <order ...> element, it's neatly put in a structure called XmlAttributes. So we can reference it like so:

```
XmlOrder.order.XmlAttributes.id
```

Since the <order> element also happens to be the xmlroot, we can reference the same attribute like so:

```
XmlOrder.xmlroot.XmlAttributes.id
```

Since XmlAttributes is a standard ColdFusion structure, any functions for struct manipulation will work on it as well, so for example if you wanted to retrieve the list of attributes from the <order...> element, you could use StructKeyList(XmlOrder.order.XmlAttributes).

On lines 15 and 19, attributes of the <customer ...> element are displayed. The customer element is a child of xmlroot, so we refer to it as "XmlOrder.xmlroot.XmlChildren[1]" (the array index is 1 since it appears as the first child of the xmlroot), or we can refer to it by a name like "XmlOrder.xmlroot.customer." To access attributes of the customer element we simply refer to the XmlAttributes struct.

Jump down to line 30 where we are doing a loop over each item in the <items> element. Since each <item> is a child of <items>, it is organized in an array called XmlChildren. Unlike the previous example, we can't refer to each of the items by its name because there are three elements with the same name. Instead, since XmlChildren is an array, we just loop over the array and refer to each <item> with its array index.

On line 31 we are setting a local variable xnItem to each item of the XmlChildren Array. As you may have noticed, XML notation can get long. You can use variables like this to shorten references to your XML Elements. This makes the code a little more human-readable and a lot easier to maintain.

Lines 33-36 use the same techniques as above to reference the information in the XML Document. Note that the Item ID is an attribute, while the other pieces of information are Child elements. These child elements refer to a variable called XmlText. All XML Elements have two property variables: XmlName and XmlText. XmlName always contains the name of the element (or tag), while XmlText is the text between the opening and closing tags that isn't inside any other elements. So for an Xml

Element like "<material>wood</material>," XmlName would be "material" and XmlText would be "wood." Applying this to our example, XmlOrder.xmlroot.XmlName would be "order."

Creating New XML Documents

So far the only XML-specific ColdFusion tag we've used is XmlParse(). I highlight this to emphasize the point that once you parse your XML into an XML Object, it is basically a ColdFusion structure of arrays and structures, and you don't really need any more XML-specific tags or functions to read the data.

Now, let's move on to creating new XML documents. There are a few ways to do this. The first way – which some consider to be "cheating" – is fast and easy.

You can simply hand-code your XML in a CFML document and wrap <cfsavecontent> tags around it (see Listing 4). This is extremely easy because you can drop in your own variables and any other CFML you want right inline with your XML. Note that with this method, you're basically hand-creating a string, which has to be parsed in an XML Object.

You can also use the <cfxml> tag, which works pretty much the same, except that instead of ending up with a string, you get the XML Object without having to call XmlParse() (see Listing 5).

Functionally there's not too much difference between these methods, both are quick and dirty and get the job done.

Now, here's the programmatic way to create a new XML Document:

```
<cfset myXml = XmlNew()>
<cfset myXml.xmlroot = XmlElemNew(myXml,"collection")>
```

The first line creates a new empty XML Object. The second line creates a new XML Element and assigns it to the xmlroot of your XML Object. Note that XmlElemNew() takes two parameters: the first is the XML Object, and the second is the name of the element you're creating. Try doing a <cfdump var="#myXml#"> and look at the results. Note that when the XML Object is created, it's already a ColdFusion object, not a plain text format. If you want to see what the XML looks like in its text format, try doing a <cfdump var="#toString(myXml)#">.

Now lets add an attribute to our new XML Element.

```
<cfset myXml.xmlroot.XmlAttributes.name = "My CDs">
```

Remember that XmlAttributes is a struct, so you can also use any of ColdFusion's struct functions here, such as StructInsert or StructUpdate.

Now let's create a new element for our CD Collection and give it some attributes:

```
<cfset xnCD = XmlElemNew(myXml,"cd")>
```

Here we're creating an XML Element that isn't attached to the XML Document right now. If you did a <cfdump var="#myXml#">, you wouldn't see this new element appear in myXml anywhere.

The technique of creating "unattached" XML elements is an important technique to understand. When we create an XML Element, we can refer to it with its short variable name and manipulate it as much as we want, including adding attributes and child elements to its XmlChildren array.

Let's add a few attributes to our xnCD element:

```
<cfset xnCD.XmlAttributes.title = "The best of Billy Joel">
<cfset xnCD.XmlAttributes.cover = "billjoel_best.jpg">
<cfset xnCD.XmlAttributes.genre = "Easy Listening">
```

Now that we have added some attributes, take a look at your XML Element by doing a <cfdump var="#xnCD#"> and make sure that the values you expected show up properly.

Once you're happy with this element, let's plug it into the XML Document. If you remember, I mentioned earlier that each XML Element has a property called XmlChildren, which is an array of XML Elements. Since it's an array, let's use ColdFusion's ArrayAppend() function to add our new element to the XML Document.

```
<cfset ArrayAppend(myXml.xmlroot.XmlChildren,xnCD)>
```

Now let's take a look at the XML Object by doing a <cfdump var="#myXml#"> and see that our xnCD Element has been added to our XML Document, and now appears in XMLChildren under collection. This technique of assembling XML elements and then "plugging them into" the XML Document is a structured and reliable way for you to programmatically build XML Documents. You can use queries and loop constructs to build XML from database results or from FORM data and ensure that the resulting XML is well formed. As an exercise, try adding some "audio track" XML elements to the above CD by creating new XML elements and appending them to xnCD.XmlChildren before adding it to your XML Document.

Now that we're done creating our XML Document, getting the actual XML Text is as simple as calling ColdFusion's toString() function. So to write the XML to a local file we could do:

```
<cffile action="write"
file="#ExpandPath(".")#/collection.xml"
   output="#tostring(myXml)#">
```

Validating XML Documents

The examples we worked with so far are very basic, and most of us will work with much more complex data. So what happens when we have XML data that may not be valid (see the definition earlier), or even worse, what if it's not well formed? (again reference above).

When you try to parse an XML document, occasionally there will be problems, specifically if the document itself is not well formed. Unfortunately the error you receive is not always very user-friendly such as "An error occurred while parsing an XML document."

If you have XML that is 10 lines long, chances are you can just take a quick look at it and figure out the problem, but if your XML Document has 2,000 rows of data, it's no longer a trivial process. That's where XmlValidate() comes into play.

This valuable and under-used function checks whether an XML Document is well formed and valid, and it also returns a list of what's wrong with the XML Document. That list is exactly what we need to "fix up" or verify that the data we have is usable.

XmlValidate() takes two arguments, the first is the XML you want to check. This can be text you read from an XML file, an XML ColdFusion Object you created, or the path and filename of the XML file on the Web server. The second argument is optional and is for the DTD, and can also be a string, URL, or path and filename.

When you call XmlValidate, it'll check your XML Document and return a struct with several valuable parts: Errors is an array that contains any errors in the document that prevent it from being valid (i.e., complying with the DTD); FatalErrors is an array that contains any errors in the document that prevent it from being well formed (and parsable using XmlParse); Status tells us whether all the tests were passed or not and returns a YES or NO; Warning is an array of any warnings that XmlValidate() found while checking the XML Document.

The FatalErrors array we get from XmlValidate is incredibly valuable. It will tell you which line and at which character it found a problem, and in my experience provides relatively helpful error messages.

Searching XML Documents Using XmlSearch()

We now understand that we can parse an XML Document and get a ColdFusion XML Object that is a bunch of arrays and structures, so we can read pieces as we want.

What if you only want to pull out a few parts of an XML Document, or if you are only interested in one type of XML Element from your document?

XmlSearch is designed to let you search your XML document for specifically named XML Elements. Let's look again at Listing 2. If we wanted to get an array of all the <item> elements, we could refer to the XmlChildren array as we did before. But what if this XML Document contained multiple orders? Then for each order there would be a different set of <item> elements, one set for each order. Getting a list of all the <item> elements is a whole lot more complex in this scenario. Instead of trying to pull apart the information we want, let's use XmlSearch() instead (see Listing 6).

XmlSearch() takes two parameters, the first is your ColdFusion XML Object and the second is an XPath expression. XPath, according to Wikipedia, is an expression language for addressing portions of an XML Document. We are only going to use it for basic purposes, but if you want to read up on XPath, a good place to start is with http://www.w3.org/TR/xpath.

On line 8 we are calling XmlSearch. The first argument is the XML Object we are working with, and the second is the XPath expression. This simple expression simply states "All XML Elements where the XMLName of the Element is Item."

The return type from XmlSearch is an array, so we save the results in an array variable called myResultsArray.

In the results, each item of the array is the actual XML Element that XmlSearch() found. This means that each search result will have the same Structure values as any other XML element, including XMLName, XMLText, XMLAttributes, and XMLChildren.

Once you have your results, you can use any of the techniques above to read and use the data that was found.

What to Study Next

We certainly can't cover every aspect of programming with XML here, but luckily you have the Internet at your fingertips.

Some of the recommended topics for study are listed below:

- **XPath Language** – http://en.wikipedia.org/wiki/XPath
- **CDATA (character data embedded in XML)** – http://en.wikipedia.org/wiki/CDATA
- **XSL** – http://en.wikipedia.org/wiki/Extensible_Stylesheet_Language
- **XmlTransform()** – ColdFusion function – search Google

Summary

I hope this has been a useful overview of coding with XML and the various built-in tools in ColdFusion. These tools should help get started with reading and creating well-formed XML.

About the Author

Andrew Schwabe is chief technology officer at IEXP FusionDox (http://www.fusiondox.org), where he manages the company's FusionDox document management technology for ColdFusion.
aschwabe@gmail.com.

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
Adobe	www.adobe.com/products/coldfusion/flex2		2,3
CFDynamics	www.cfdynamics.com	866-233-9626	4
Cfunited	www.cfunited.com		17
CFDJ	http://coldfusion.sys-con.com/	201-802-3000	31
EdgeWebHosting	edgewebhosting.net	1-866-334-3932	35
HostMySite	www.hostmysite.com	877-215-4678	11, 36
Hotbanana	hotbana.com/cfdj	866-296-1803	6
Intergral	www.fusiondebug.com		15
Paperthin	www.paperthin.com	1-800-940-3087	21
SOAWorld Conf & Expo	www.soaworld2007.com	201-802-3020	25
WebApper	www.seefusion.com		19

Listing 1

```
<?xml version="1.0" encoding="UTF-8"?>
<directory>
   <employee>
       <name last="Smith" first="John" />
       <address>
           <street1>512 Goshen Road</street1>
           <street2/>
           <city>Bordell</city>
           <state>OH</state>
           <zip>35426</zip>
       </address>
   </employee>
</directory>
```

Listing 2: "custorder.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<order id="E10645">
   <customer fname="Robert" lname="Finn" memberid="238"/>
   <items>
      <item id="544">
         <name>Headphones</name>
         <quantity>1</quantity>
         <unitprice>14.95</unitprice>
      </item>
      <item id="871">
         <name>CD Cleaner</name>
         <quantity>1</quantity>
         <unitprice>3.95</unitprice>
      </item>
      <item id="77">
         <name>Greatest Hits Album</name>
         <quantity>1</quantity>
         <unitprice>11.95</unitprice>
      </item>
   </items>
</order>
```

Listing 3: "order.cfm"

```
<cffile action="read"
      file="#ExpandPath(".")#/custorder.xml"
      variable="strXmlOrder">

<cfset XmlOrder = XmlParse(strXmlOrder)>

<cfoutput>
<table>
<tr>
   <td>Order ID:</td>
   <td>#XmlOrder.xmlroot.XmlAttributes["id"]#</td>
</tr>
<tr>
   <td>Member Name:</td>
   <td>#XmlOrder.xmlroot.customer.XmlAttributes["lname"]#, #XmlOrder.
xmlroot.customer.XmlAttributes["fname"]#</td>
</tr>
<tr>
   <td>Member ID:</td>
   <td>#XmlOrder.xmlroot.customer.XmlAttributes["memberid"]#</td>
</tr>
<tr valign="top">
   <td>Items ordered</td>
   <td><table>
   <tr>
      <td>ID</td>
      <td>Item</td>
      <td>Qty</td>
      <td>Price</td>
   </tr>
      <cfloop from="1"
              to="#ArrayLen(XmlOrder.xmlroot.items.XmlChildren)#"
              index="i">
      <cfset xnItem = XmlOrder.xmlroot.items.XmlChildren[i]>
      <tr>
         <td>#xnItem.XmlAttributes["id"]#</td>
         <td>#xnItem.name.XmlText#</td>
         <td>#xnItem.quantity.XmlText#</td>
         <td>#xnItem.unitprice.XmlText#</td>
      </tr>
   </cfloop>
   </table></td>
</tr>
</table>
</cfoutput>
```

Listing 4: create XML using cfsavecontent

```
<cfsavecontent variable="cfsavecontent_example">
<report_using_cfsavecontent>
   <date>
<cfoutput>#DateFormat(now(),"mm/dd/yyyy")#</cfoutput>
</date>
   <topics>
       <topic name="Schedule">
We need to plan the new schedule
</topic>
       <topic name="Budget">
New budgets are available for review
</topic>
       <topic name="Supplies"/>
   </topics>
</report_using_cfsavecontent>
</cfsavecontent>
<cfset testXml = XmlParse(cfsavecontent_example)>
<cfdump var="#testXml#">
```

Listing 5: create XML using cfxml

```
<cfxml variable="cfxml_example">
<report_using_cfxml>
   <date>
<cfoutput>#DateFormat(now(),"mm/dd/yyyy")#</cfoutput>
</date>
   <topics>
       <topic name="Schedule">
We need to plan the new schedule
</topic>
       <topic name="Budget">
New budgets are available for review
</topic>
       <topic name="Supplies"/>
   </topics>
</report_using_cfxml>
</cfxml>
<cfdump var="#cfxml_example#">
```

Listing 6: XML Search

```
<!--- read my XML from a file --->
<cffile action="read" file="#ExpandPath(".")#/custorder.xml"
variable="strXml">

<!--- parse into a CF XML Object --->
<cfset myXml = XmlParse(strXml)>

<!--- search for all "item" elements --->
<cfset myResultsArray = XMLSearch(myXml, "//item")>

<!--- just dump the results so we can see --->
<cfdump var="#myResultsArray#">
```

Download the Code...
Go to http://coldfusion.sys-con.com

“Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises

— Yafim Natis, Gartner Analyst

3-DAY EVENT!

SOAWorld 2007 Plus Enterprise OpenSource Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services	Open Source
› AJAX and SOA	› Open Source Business Models
› Web 2.0	› Open Source ESB
› Universal SOA	› OpenAjax Alliance
› Protecting Web Services	› SaaS and Open Source
› Troubleshooting SOA	› Spring, Hibernate and Eclipse
› Governance	› Seam
› Open-Source SOA	› Open Source Penetration
› XBRL	› Monetizing Open Source
› Service Virtualization	› Open Source Databases
	› AMQP
	› Open Source Middleware

June 25-27, 2007

Roosevelt Hotel / New York City

Register Online! www.SOAWorld2007.com

SOAEOSCONFERENCE.SYS-CON.COM

REGISTER ONLINE TODAY

SAVE $200!

(HURRY FOR EARLY-BIRD DISCOUNT)

11th International 2007 SOAWORLD™ CONFERENCE & EXPO

2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-27, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

BROUGHT TO YOU BY:

› SOA World Magazine focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.

› Enterprise Open Source Magazine EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Exhibit and Sponsorship Info:

Call 201-802-3020 or email events@sys-con.com

ColdFusion

For more information go to...

U.S.

Alabama
Huntsville, AL CFUG
www.nacfug.com

Arizona
Phoenix, AZ CFUG
www.azcfug.com

California
Bay Area CFUG
www.bacfug.net

California
Sacramento, CA CFUG
http://www.saccfug.com/

California
San Diego, CA CFUG
www.sdcfug.org/

Colorado
Denver CFUG
http://www.denvercfug.org/

Connecticut
SW CT CFUG
http://www.cfugitives.com/

Connecticut
Hartford CFUG
http://www.ctmug.com/

Delaware
Wilmington CFUG
http://www.bvcfug.org/

Florida
Jacksonville CFUG
http://www.jaxfusion.org/

Florida
South Florida CFUG
www.cfug-sfl.org

Georgia
Atlanta, GA CFUG
www.acfug.org

Illinois
Chicago CFUG
http://www.cccfug.org

Indiana
Indianapolis, IN CFUG
www.hoosierfusion.com

Louisiana
Lafayette, LA MMUG
http://www.acadianammug.org/

Maryland
California, MD CFUG
http://www.smdcfug.org

Maryland
Maryland CFUG
www.cfug-md.org

Massachusetts
Boston CFUG
http://bostoncfug.org/

Massachusetts
Online CFUG
http://coldfusion.meetup.com/17/

Michigan
Detroit CFUG
http://www.detcfug.org/

Michigan
Mid Michigan CFUG
www.coldfusion.org/pages/index.cfm

Minnesota
Southeastern MN CFUG
http://www.bittercoldfusion.com

Minnesota
Twin Cities CFUG
www.colderfusion.com

Missouri
Kansas City, MO CFUG
www.kcfusion.org

Nebraska
Omaha, NE CFUG
www.necfug.com

New Jersey
Central New Jersey CFUG
http://www.cjcfug.us

New Hampshire
UNH CFUG
http://unhce.unh.edu/blogs/mmug/

New York
Rochester, NY CFUG
http://rcfug.org/

New York
Albany, NY CFUG
www.anycfug.org

New York
New York, NY CFUG
www.nycfug.org

New York
Syracuse, NY CFUG
www.cfugcny.org

North Carolina
Raleigh, NC CFUG
http://tacfug.org/

Ohio
Cleveland CFUG
http://www.clevelandcfug.org

Oregon
Portland, OR CFUG
www.pdxcfug.org

Pennsylvania
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Philadelphia, PA CFUG
http://www.phillycfug.org/

Pennsylvania
State College, PA CFUG
www.mmug-sc.org/

Tennessee
Nashville, TN CFUG
http://www.ncfug.com

Tennessee
Memphis, TN CFUG
http://mmug.mind-over-data.com

Texas
Austin, TX CFUG
http://cftexas.net/

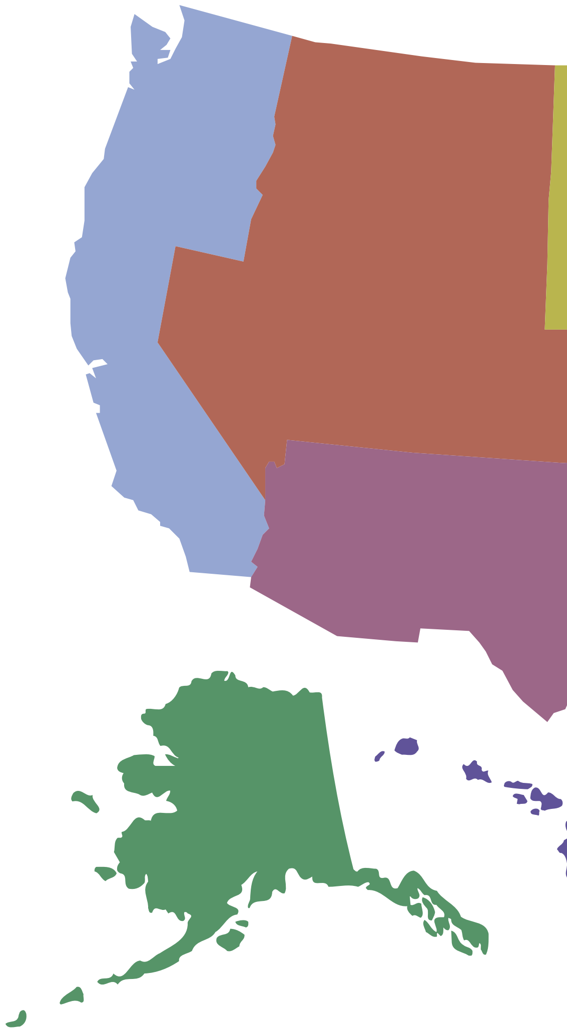
Texas
Dallas, TX CFUG
www.dfwcfug.org/

Texas
Houston Area CFUG
http://www.houcfug.org

Utah
Salt Lake City, UT CFUG
www.slcfug.org

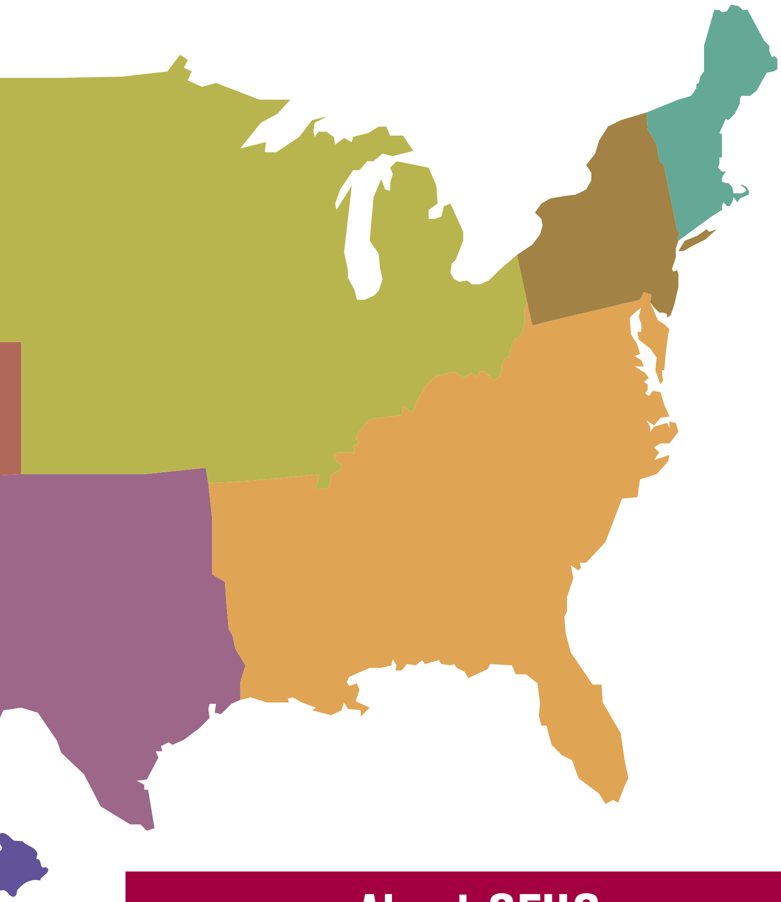
Virginia
Charlottesville CFUG
http://indorgs.virginia.edu/cfug/

Washington
Seattle CFUG
http://www.seattlecfug.com/



Jser Groups

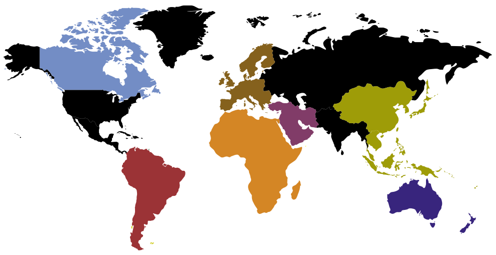
http://www.macromedia.com/cfusion/usergroups



About CFUGs

ColdFusion User Groups
provide a forum of support and technology to Web professionals of all levels and professions. Whether you're a designer, seasoned developer, or just starting out – ColdFusion User Groups strengthen community, increase networking, unveil the latest technology innovations, and reveal the techniques that turn novices into experts, and experts into gurus.

INTERNATIONAL



Australia
ACT CFUG
http://www.actcfug.com

Australia
Queensland CFUG
http://qld.cfug.org.au/

Australia
Victoria CFUG
http://www.cfcentral.com.au

Australia
Western Australia CFUG
http://www.cfugwa.com/

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Germany
Central Europe CFUG
www.cfug.de

Italy
Italy CFUG
http://www.cfmentor.com

New Zealand
Auckland CFUG
http://www.cfug.co.nz/

Poland
Polish CFUG
http://www.cfml.pl

Scotland
Scottish CFUG
www.scottishcfug.com

South Africa
Joe-Burg, South Africa CFUG
www.mmug.co.za

South Africa
Cape Town, South Africa CFUG
www.mmug.co.za

Spain
Spanish CFUG
http://www.cfugspain.org

Sweden
Gothenburg, Sweden CFUG
www.cfug-se.org

Switzerland
Swiss CFUG
http://www.swisscfug.org

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

ColdFusion 8

Tips from the tour

By Ben Forta

The following article is a compilation of Ben Forta's blog entries from his ColdFusion 8 usergroup tour.

ColdFusion 8-Supported Image Formats

Lots of you have been asking about ColdFusion 8's planned <CFIMAGE> tag, wanting to know what image formats will be supported (many of you are asking about GIF and PNG specifically). As long as you keep in mind that ColdFusion 8 is a work-in-progress and that features and specs can change...

ColdFusion 8 has two new functions: GetReadableImageFormats() returns a list of the supported readable image formats, and GetWriteableImageFormats() returns a list of supported writable formats. The following code snippet shows all supported formats:

```
<cfset r=GetReadableImageFormats()>
<cfset w=GetWriteableImageFormats()>

<cfoutput>
Readable (#ListLen(r)#): #r#<br>
Writeable (#ListLen(w)#): #w#<br>
</cfoutput>
```

Running this code (in Beta 2, remember, this can/may still change) displays the following:

```
Readable (15): BMP,GIF,JFIF,JPEG,JPEG 2000,JPEG-LOSSLESS,JPEG-LS,JPEG2000
,JPG,PNG,PNM,RAW,TIF,TIFF,WBMP
Writeable (15): BMP,GIF,JFIF,JPEG,JPEG 2000,JPEG-LOSSLESS,JPEG-LS,JPEG200
0,JPG,PNG,PNM,RAW,TIF,TIFF,WBMP
```

And, yes, GIF and PNG are both supported (both read and write).

Getting Started with the ColdFusion Debugger

After a seven-year hiatus, ColdFusion once again has an interactive debugger, and this time it's built on top of Eclipse (and uses the same debugging interface as Flex Builder and other Eclipse plug-ins). If you are interested in taking the debugger for a spin, here's what you need to know to get started:

First of all, you need ColdFusion 8. If you don't have it yet, get it here: http://labs.adobe.com/technologies/coldfusion8/. You'll also need the ColdFusion 8 Eclipse extensions, and these must be installed (here is some installation help: http://www.forta.com/blog/index.cfm/2007/5/30/Installing-ColdFusion-8-Eclipse-Extensions).

Before you can use the interactive debugger, enable debugging support in ColdFusion. Here's what you need to do:

1. Open the ColdFusion Administrator.
2. Select "Debugger Settings" in the "Debugging & Logging" section.
3. Check "Allow Line Debugging".
4. The debugger needs a port to communicate over. If you can't use the default, change it (just be sure to use an unused port, and you can't use the port that ColdFusion itself is on).
5. By default, ColdFusion allows up to five concurrent debugging sessions; you can raise or lower this value as needed.
6. Click "Submit Changes" and restart ColdFusion as you are instructed to do.

Once you enable line debugging, it will remain enabled even if ColdFusion is restarted. As a rule, don't enable line debugging on production servers, and you may want to always leave it enabled on development boxes.

Now that debugging is enabled, configure Eclipse to tell it which ColdFusion server to debug against. Servers need only be defined once, and once defined you may debug against them as needed. You can define as many servers as you need, local and remote, as long as the server has RDS enabled (and you have an RDS login).

The first thing you need to do in Eclipse is define the RDS settings for your ColdFusion server. These settings are used by the debugger, as well as the wizards, RDS panels, and more. To define your RDS connection, do the following:

1. In Eclipse, select Window->Preferences to display the Preferences dialog.
2. Select ColdFusion in the tree, expand the selection, and select RDS Configuration.
3. Click New to add a new server, or just select any server to edit it.
4. Provide a description, host name, port, and login details, and then save. To use your local ColdFusion server, specify 127.0.0.1 as the Host Name, 8500 as the Port (if using the integrated HTTP server), leave the Context Root blank, and provide the login information.
5. Click "Test Connection" to make sure the RDS connection is working, and then click OK.

Once the RDS connection is defined, you'll be able to browse data sources in the RDS Dataview tab, view available CFCs in the Services Browser tab, use the wizards, and more. And you'll be able to debug applications, but first...

Now that your RDS connection is defined, define the ColdFusion servers you wish to debug against. Here's what you need to do:

1. Locate the Debug button in the toolbar; it's the one with the little green bug on it.
2. Don't click the button. Instead, click the down arrow to the right of it and select "Debug..." to display the Debug dialog, which is used to manage debugging configurations.
3. The Eclipse debugger is used to debug all sorts of applications created in all sorts of languages, and along the left of the dialog you'll see a list of application types that can be debugged. Select "ColdFusion Application".
4. You'll see any defined ColdFusion debugging servers under the "ColdFusion Application" branch. To add a ColdFusion server, click the New button (the leftmost one above the list) while you have "ColdFusion Application" selected.
5. Name this server and then select the RDS server to use from the dropdown list.
6. If you are debugging against a local server (running on the same machine as Eclipse), you can ignore the mappings section. If ColdFusion is on a remote server, you'll need to define mappings here.
7. Click "Apply" to save your changes.

Again, servers need to be defined once, and they'll be saved for future use.

To debug your code you can do the following:

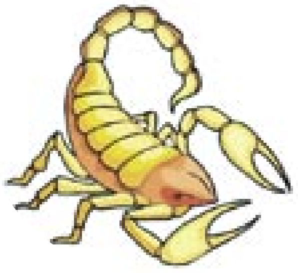
1. Open the ColdFusion file to be debugged in Eclipse.
2. Set breakpoints as needed by double-clicking to the left of the desired line in the vertical bar to the left of the Eclipse editor window. You'll see a little circle appear, indicating that a breakpoint has been set. You can also right-click in the vertical bar and select "Toggle Breakpoint".
3. Now you need to switch to the Eclipse Debug view (called a "Perspective" in Eclipse). If you see a Debug button listed with the Perspectives on the top right of Eclipse, click it. Or, just click the Open Perspective button (or select Windows->Open Perspective) and select "Debug".
4. Once you are in the Debug perspective, start the debugging session. To do this, go back to the dialog where you defined the ColdFusion server (under the Debug button), select the ColdFusion server to debug against, and click the Debug button at the bottom of the dialog.
5. Eclipse will pause while it opens a debug session, connecting to the specified ColdFusion server. You'll see the connection show up in the Debug window.
6. Now just open any browser and run your application. When you request a page with a breakpoint, execution will pause and the debugger will pop up. You'll be able to step through the code, view variables and expressions (in the top right panel), see generated server output, and more.
7. When you're done debugging, you can terminate the debug session by clicking the red square Terminate button.

That'll do it. I know this seems complex and a lot of work, but the bulk of it is the initial setup. Once that setup is complete, you'll be able to quickly and easily fire up the debugger on demand and when needed.

ColdFusion 8 File I/O Enhancements

At one of the usergroup sessions someone asked if there was a way to get file information (size, date time, etc.) easily using a function. I said they should use <CFDIRECTORY>, but afterwards remembered that we did indeed add a new function to ColdFusion 8 called GetFileInfo() that returns a structure containing:

- canread
- canwrite
- ishidden
- lastmodified
- name
- parent
- path
- size
- type



Which makes this a good opportunity to review some of the file i/o changes coming in ColdFusion 8.

For starters, if you ever had to work with large text files in ColdFusion (maybe parsing a large CSV file), you'll know that doing so is very inefficient. You probably use code like this:

```
<!--- Read entire file --->
<cffile action="read"
   file="#fileName#"
   variable="myFile">
<!--- Loop through file variable one line at a time --->
<cfloop list="#myFile#"
   index="line"
   delimiters="#chr(10)##chr(13)#">
   <!--- Do stuff with line here --->
   ...
</cfloop>
```

This is slow for two reasons. Not only does ColdFusion read the entire file into memory in a variable all at once, but also looping through the file requires treating it as a list, which involves lots of parsing that can also be resource intensive.
Well, inefficient no more. In ColdFusion ColdFusion 8 you'll be able to replace the above code block with this:

```
<!--- Loop through file one line at a time --->
<cfloop file="#fileName#"
   index="line">
   <!--- Do stuff with line here --->
   ...
</cfloop>
```

This code block opens the file, reads one line at a time, and closes it when done. I actually used this myself recently in a ColdFusion code snippet that had to parse a massive tab-delimited file, turning each line into a query row. Replacing the old <CFFILE> <CFLOOP> with a new <CFFILE FILE=> cut down the processing time from several minutes to under 10 seconds. Although reading files line by line is the more common use case, you can also read by *n* characters at a time, like this:

```
<!--- Loop through file 100 characters at a time --->
<cfloop file="#fileName#"
   index="chars"
   characters="100">
   <!--- Do stuff with line here --->
   ...
</cfloop>
```

In addition to the <CFLOOP> enhancements, we've also added lots of new file i/o functions that you can use to access and manipulate files directly. The new functions include:

- FileClose()
- FileCopy()
- FileDelete()
- FileIsEOF()
- FileMove()
- FileOpen()
- FileRead()
- FileReadBinary()
- FileReadLine()
- FileSetAccessMode()
- FileSetAttribute()
- FileSetLastModified()
- FileWrite()
- GetFileInfo()
- IsImageFile()
- IsPDFFile()

ColdFusion 8 Makes Obtaining Database and Table Info Easy

This feature was first demonstrated at a usergroup presentation in Connecticut: a new tag in ColdFusion 8 named <CFDBINFO> that returns information about databases (and data sources, and tables, and columns, and stored procedures, and more). This first code snippet shows how to obtain a list of tables in a specified data source (using the default database):

```
<cfdbinfo type="tables" datasource="myDSN" name="tables">
```

To get column details you can do the following:

```
<cfdbinfo type="columns" datasource="myDSN" table="myTables"
name="columns">
```

This would return a query containing column names, type, size, default values, whether it allows NULL values, key associations, and more.

As you can see, <CFDBINFO> accepts a TYPE attribute that tells it what information you want, and the following types are supported:

- **columns:** Returns column details for a specific table.
- **dbnames:** Returns the databases in a specified data source.
- **foreignkeys:** Returns foreign key name information, including the associated primary key, and delete and update rules.
- **index:** Returns index specifics, including column details, page usage, and whether or not the index is unique.
- **procedures:** Returns available stored procedures.
- **tables:** Returns the names of tables within a specific database.
- **version:** Returns database drive version details.

Multi-Threaded Application Development in ColdFusion 8

ColdFusion MX7 introduced the ability to asynchronously spawn ColdFusion requests using an event gateway. While many take advantage of this capability, it has some significant limitations, the biggest of which is that threads can only be spawned; there is no way to monitor spawned threads or wait for them to finish. (The other limitation is that the functionality is only available in ColdFusion Enterprise.)

ColdFusion 8 provides far more sophisticated multi-threading capabilities via the new <CFTHREAD> tag. This tag is used to perform several actions:

- JOIN causes the current thread to wait until one or more specified threads have finished processing.
- RUN creates a new thread that starts processing immediately.
- SLEEP makes another thread pause for a specified number of milliseconds.
- TERMINATE kills another thread.

There are lots of use cases for this new functionality, but at a minimum there are two primary usage scenarios:

- Many requests process user submissions, for example, a user uploaded file. The way most ColdFusion applications work today is that the file is processed on the server (parsing it, converting it, saving it, etc.) while the user waits. But in truth, there is no reason users should have to wait for your application to finish its processing. A better user experience would be to receive the file in the action page, spawn a new thread to actually process it, and return control back to the user instantly. This creates a far more responsive user experience.
- Applications often have to perform multiple operations (perhaps updating database rows, writing log entries, generating an e-mail, firing server-side HTTP requests, and more). Most ColdFusion applications perform these tasks sequentially, one after the other, and then return to the user when complete. But if the various operations are not actually dependent on each other, you could spawn a thread for each, having them execute concurrently and, if necessary, wait until they are complete to continue processing. The result is a faster application, as multiple operations are being performed concurrently.

The code to spawn a thread is very simple:

```
<!--- Use a separate thread to perform file processing --->
<cfthread action="run" name="threadFile" file="#myFile#">
   <cffile file="#ATTRIBUTES.file#" ...>
</cfthread>
```

Here a thread named "threadFile" is spawned. An argument (the file to be processed) is passed to <CFTHREAD>, and so that attribute is available within the thread in the ATTRIBUTES scope.

Within threads there are several important scopes. Any locally defined variables are implicitly thread local. THREAD is a special scope (a sub-scope of VARIABLES) that is available to all threads spawned by a single parent. ATTRIBUTES is used to access any variables passed as attributes to <CFTHREAD>.

The previous example spawns a thread that could continue processing long after the parent page terminates. If you needed to wait for a thread to complete, you could use the following code:

```
<cfthread action="join" name="threadFile">
```

JOIN is used to wait for one or more threads to complete, and multiple thread names may be specified (as may a timeout value).

Once defined, the thread name can be accessed as a structure that exposes the following members:

- ELPASEDTIME is the amount of time since the thread was spawned.
- ERROR contains any error messages generated by the code in the spawned thread.
- NAME is the thread name.
- OUTPUT contains any generated output. This output will not be sent to the client, but parent page code can access the output that can then be used as needed.
- PRIORITY is the thread priority level (HIGH, LOW, NORMAL).
- STARTIME is the time the thread started.
- STATUS is the thread status (NOT_STARTED, RUNNING, TERMINATED, COMPLETED, WAITING).

To check that threads executed properly without errors, you could JOIN the threads, and then check STATUS to see if they completed. A status of TERMINATED means an error occurred (or that threads were explicitly terminated), in which case ERROR would provide details as to what happened.

ZIP and JAR File Access in ColdFusion 8

During the usergroup presentation at the Mid-Michigan CFUG in East Lansing, I mentioned another new ColdFusion 8 tag, one we haven't yet discussed publicly; this tag will now appear last in alphabetical CFML tag listings. <CFZIP> provides access to ZIP and JAR files, supporting the following actions:

- DELETE deletes one or more files from an archive file.
- LIST lists the contents of an archive file.
- READ reads the contents of an archived file into a variable.
- READBINARY reads the contents of a binary archived file into a variable.
- UNZIP extracts files from an archive file.
- ZIP compressed files into an archive file.

A child <CFZIPPARAM> tag is also supported as an optional way to provide file names and other attributes.

Roundup

Everyone is excited about ColdFusion 8; the blogs have been buzzing, feedback has been superb, lots of attendees expressed relief and gratitude at seeing such a compelling ColdFusion built in this new Adobe era – no complaints, no negative feedback, just real enthusiasm and excitement. The only frequently heard complaint was from users who want it now!

About the Author

Ben Forta is Adobe's evangelist for the ColdFusion product line. He is the author of several books.

ben@forta.com

Efficient Web Content Management

CommonSpot™ Content Server is the ColdFusion developer's leading choice for efficient Web content management.

Our rich Web publishing framework empowers developers with out-of-the-box features such as template-driven pages, custom content objects, granular security, and powerful ColdFusion integration hooks (just to name a few), allowing you to rapidly build and efficiently deploy dynamic, high performance Web sites.

CommonSpot's open architecture and extensive APIs enable easy customization and integration of external applications. With CommonSpot, you can design and build exactly what you need. Best of all, CommonSpot puts content management in the hands of content owners. With non-technical users responsible for creating and managing Web content, developers are freed to focus on strategic application development.

Evaluate CommonSpot today.

To see your site *running* under CommonSpot, call us at 1.800.940.3087.

features.

- 100% browser-based
- Content object architecture
- Template driven pages
- 50+ standard elements
- Content reuse
- Content scheduling
- Personalization
- Flexible workflow
- Granular security
- Mac-based authoring
- 508 compliance
- Full CSS support
- Custom metadata
- Taxonomy module
- Extensible via ColdFusion
- Web Services content API
- Custom authentication
- Replication
- Static site generation
- Multilanguage support

1.800.940.3087
www.paperthin.com

Paper Thin

Wait-Time Analysis Method

A new best practice for application and database performance management

By Don Bergal

Until recently, tuning IT application performance has been largely a guessing game. This is both surprising and unacceptable considering the relentless focus IT organizations put on cost-efficiency and productivity.

The traditional approaches to database and application tuning that involve collecting large volumes of statistics and making trial-and-error changes are still in widespread use. Today, most server management and monitoring tools deliver "server-oriented" statistics that don't translate to concrete end-user benefits.

The landscape is changing, however. The current thinking of leading consultants, DBAs, and training organizations is focusing on performance tuning practices that are tied directly to end-user service levels and improvements in operating efficiency.

Wait-Time analysis is a new approach to application and database performance improvement that allows users to make tuning decisions based on the optimal service impact. Using the principles of Wait-Time analysis described here, DBAs, developers, and application owners can align their efforts with the service levels desired by their IT customers. Wait-Time analysis lets IT find the root cause of the most important problem impacting customers and identify which critical resource will resolve it.

What Is Wait-Time Analysis?

Measure Time

If you were trying to shorten your commute to work, what would you measure? Would you count the number of tire rotations? Would you measure the car's temperature? Would these statistics have any meaning in the context of your goal? All that really matters is what impacts the time for your trip. All the other statistics are distractions that don't help your mission. Wait-Time analysis gets to the root of the problem to achieve the end business result. Although this seems obvious, common IT practices suggest that other practices hold the answer. Rather than immediately focusing on the time to complete requested services, IT tools barrage the user with detailed statistics that count the number of many different operations. So while the

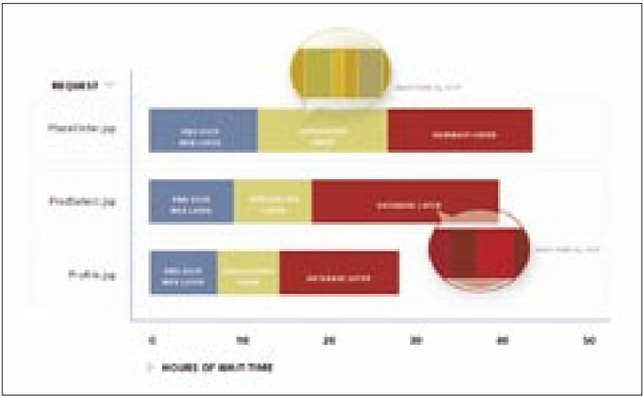


Figure 1: analysis – exactly where is every request spending time? Identifying every step inside each layer for every request exposes the true sources of end-user Wait-Time

DBA should really be looking at how long it took for the database to return the results of a query, typical tools display the number of input/output (I/O) operations and locks encountered.

Get the Details

Under the trial-and-error approach, what level of detail do you need to actively improve your commute time? If the only statistic you have is that the trip took 40 minutes, you can compare one day to the next, but there's not enough data to help improve the situation. What you need is detailed insight into how long you spent at each stoplight, which stretches of road have the most stop-and-go traffic and how long you waited there. This detail is essential to making the exercise useful.

The same concept applies to IT performance systems. When Wait-Time is typically measured, a "black box" approach is taken, where the user sees how long a server took to respond to a request. However, no indication is given as to which of the thousands of steps performed by the server were actually responsible for the delay. As will be shown here, it's important not just to measure Wait-Time but to break it down into sufficient detail so that you can take action.

Wait-Time analysis for IT applications is the singular focus of measuring and improving the service time to the IT customers. By identifying exactly what contributes to longer service time, IT professionals can focus not on the thousands of available statistics, but on the most important bottlenecks that have direct and quantifiable impact on the IT customer.

Wait-Time Analysis for Service Level Management

Because Wait-Time analysis measures the collective time delays causing end users to wait for an information request, it's the measurement technique most closely matched to end-user service levels. For organizations focused on Service Level Management (SLM) techniques, or those bound by Service Level Agreements (SLAs), Wait-Time analysis techniques allow the IT department to measure the performance that is most relevant to achieving the stated service level goals. Service level management typically identifies technical metrics that define whether performance is adequate, and Wait-Time data is the basis for evaluating those metrics.

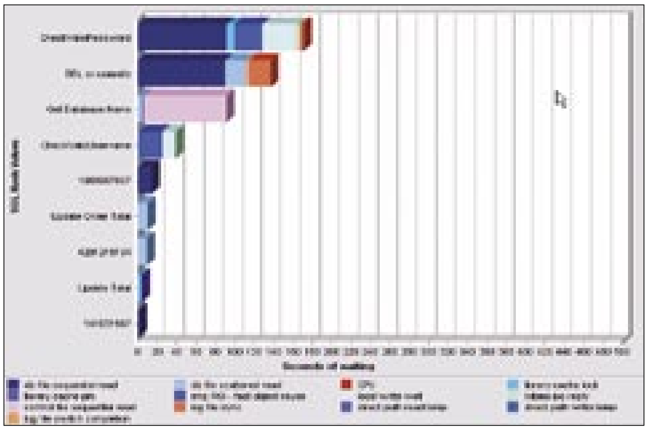


Figure 2: Wait-Time display immediately identifies the problem. Wait-Time analysis, like the example shown here, tells the DBA or system manager: "This is where you need to focus, here is the exact problem"

The Problem with Conventional Statistics

There are so many management tools gathering thousands of statistics from IT systems. Don't these provide the same answer as Wait-Time methods? Why are they not effective?

Traditional approaches to database tuning and performance analysis introduce the same errors identified in the driving example above.

1. Event Counters versus Wait-Time Methods

Typical tools count the number of events, but don't measure time. These statistics are numerous and easy to capture, so they tend to flood management dashboards. But, are they useful?

Broad management dashboards have sophisticated displays of monitored data, but counting events or calculating ratios doesn't indicate or predict better performance for database customers. In fact, this approach can have the effect of covering up, rather than exposing, the real service level bottlenecks.

The example is an excerpt from a long summary of counted statistics. Clearly there's much detail and technical accuracy. But where would you go to begin your diagnosis? Do these raw numbers reveal a performance problem? Is the value for "physical writes direct" in the table too high or too low? There's no indication of impact on the end-user service level to make that judgment.

On the other hand, Figure 2 ranks individual SQL requests by Wait-Time. The statement with the highest Wait-Time is at the top of the list. Its relative impact on overall user service is reflected in the length of the bar – measuring how much time users experience waiting on this request. Without counting how many times an operation occurred, this is a much more meaningful measure of end-user service.

2. System-Wide Averages

Typically statistics are gathered across an entire system, rather than on a basis that applies to an individual user request. When averaging performance across all requests, it becomes impossible to tell which requests are the most critical resource drains and which resources are impacting service levels.

Vendor-supplied database tools, for example, typically display data across the entire database without breaking it down into specific user requests. As a result, there's no indication which end-user functions were impacted.

3. Silos versus End-to-End Analysis

Another key problem with typical IT monitoring tools is the creation of individual information "silos" that localize statistics for a single type of system, but don't expose an end-user's view of performance.

Because of the differing technical skill sets, separate groups manage databases, application servers, and Web infrastructure. Each group has a primary focus – to optimize the performance of their box. And typically they use the most common and convenient statistics to measure and improve performance. For an application server, this often means watching memory utilization, thread counts, and CPU utilization. For a database, this is a count of the number of sessions, number of reads, or number of processes.

The problem is that there's no view of the end objective – minimizing service time for the customer – and no collaboration across these groups focusing them beyond their individual server operations. In reality, the database bottlenecks are a direct result of the application procedure calls while the application responds to Web requests. All of these combine to have a direct impact on end-user service. Without the ability to track the flow of transactions across the multiple systems, each IT group can only try to optimize its own statistics, not of the response time to the customer.

4. Finger Pointing

The real trouble starts when a cross-functional group assembles to try to respond to a customer-reported problem. With each department watching their own server-oriented statistics, the result is a "finger-pointing" session where blame is deflected from one group to the next.

Without a performance measurement system that identifies in exact detail the root cause of the performance bottleneck, finger pointing becomes inevitable. By contrast, by measuring end-user Wait-Time with the recommended detailed granularity, management can identify exactly where in the IT value chain the bottleneck lays and who is really responsible. Using Wait-Time techniques to pinpoint the source of the problem helps eliminate the finger pointing.

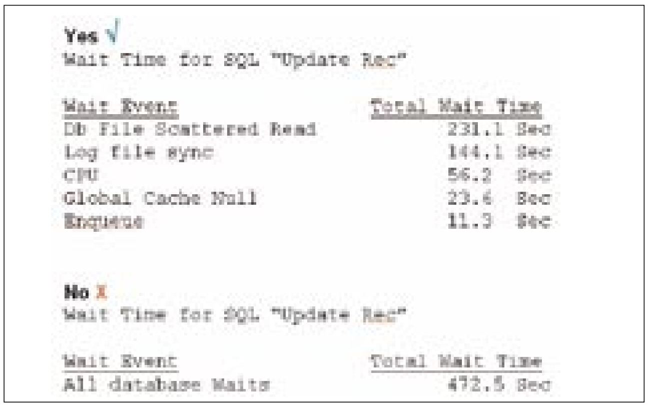
Key Requirements for Wait-Time Analysis

Wait-Time analysis is an approach to application performance management that captures and delivers data in a way that enables business decisions that have optimal service impact.

The foundations of Wait-Time analysis are three requirements – measuring User Requests, measuring Every Step, and measuring accumulated Time.

Requirement One: Every User Request – Individually

This requirement states that all IT performance statistics must correspond to specific user requests, not averages across the entire system. Individual SQL statements or Web user screens must be tracked individually as they pass through the respective servers. In a database or application server, mixing data across all requests has the effect of averaging all responses and hiding any unique information about the request of interest. To effectively identify the problem, each SQL or Java application screen must be monitored and optimized separately.



```
Yes √
Wait Time for SQL "Update Rec"

Wait Event                       Total Wait Time
Db File Scattered Read                231.1 Sec
Log file sync                         144.1 Sec
CPU                                   56.2  Sec
Global Cache Null                     23.6  Sec
Enqueue                               11.3  Sec


No X
Wait Time for SQL "Update Rec"

Wait Event                       Total Wait Time
All database Waits                    472.5 Sec
```

Figure 3: Seeing every step individually shows which ones are bottlenecks. An average of all the steps doesn't identify the problem

Requirement Two: Every Step

To be actionable, every user request must be measured with sufficient granularity to identify each step taken along the path from the end user through the database. This requires more detail than simply designating the database layer as the source of delay. It requires measuring each of the individual processes along the execution path. For an Oracle or SQL Server database, these steps correspond to hundreds of individual Wait-Events. For a Web application, the steps can be Java methods that are executed on an application server.

You can't take action if all you know is that your request waits on Java or Oracle. But if you know that your request is hung in a specific "getCreditCard.do" method or "Enqueue" database lock then you have sufficient detail to productively work the problem.

Requirement Three: Measure Time

The most important requirement is measuring the time spent on a request, not counting how often a computing resource was used. The principle follows logically from the business purpose of the information system, which is to process requests and deliver output as quickly as possible. Counting events provides no indicator of how long a database user must wait for a response, or how long a request must wait for the execution of a Java method.

In the Wait-Time service-oriented performance approach, time is the most important resource to measure.

Conclusion: Practical Considerations for Wait-Time Analysis

The Wait-Time approach to performance monitoring described here is only practical if it can be implemented efficiently in a performance-sensitive production environment. While basic tools to extract Wait-Time from databases on an individual session basis were the first step in this type of analysis, more efficient approaches have now been developed that meet ease of use, low impact, and continuous monitoring requirements. Beyond the database, it's now possible to employ Wait-Time analysis efficiently in end-to-end production application environments.

With increased focus on service levels as the most important measure of IT productivity, Wait-Time analysis has come to the forefront as the monitoring technique that ties IT practice to overall IT goals. This movement is aided by the combination of superior results experienced by leading consultants and trainers as well as the availability of excellent packaged tools. Wait-Time analysis tells the IT organization exactly where the problem lies, who should fix it and how it impacts the customer. Unlike traditional methods that barely deliver clues, Wait-Time, implemented in sufficient detail, delivers answers.

About the Author

Don Bergal is the chief operating officer of Confio Software and is responsible for overseeing all of the company's sales, marketing, product management, and business development initiatives. Don has over 15 years experience in the software, services, and data communications industries. He earned a BS in engineering from the University of Michigan and an MBA from the Harvard Business School.

DonBergal@confio.com

For answers call us at 1-866-EDGEWEB

3 3 4 3 9 3 2

When calling your web host for support you want answers, not an annoying song stuck in your head from spending all day on hold. At Edgewebhosting.net, we'll answer your call in two rings or less. There's no annoying on-hold music, no recorded messages or confusing menu merry-go-rounds. And when you call, one of our qualified experts will have the answers you're looking for. Not that you'll need to call us often since our self-healing servers virtually eliminate the potential for problems and automatically resolve most CF, ASP, .NET, SQL, IIS and Linux problems in 60 seconds or less with no human interaction.

Sleep soundly, take a vacation, and be confident knowing your server will be housed in one of the most redundant self-owned datacenters in the world alongside some of the largest sites on the Internet today and kept online and operational by one of the most advanced teams of skilled Gurus, DBAs, Network and Systems Engineers.

By the Numbers:

- 2 Rings or less, live support
- 100 % Guarantee
- 99.999% Uptime
- 2.6 GBPS Redundant Internet Fiber Connectivity
- 1st Tier Carrier Neutral Facility
- 24 x 7 Emergency support
- 24 Hour Backup
- Type IV Redundant Datacenter

EDGE WEB HOSTING

For a new kind of easy listening, talk to EdgeWebHosting.net

http://edgewebhosting.net

CFDJ Readers' Choice Award · MXDJ Readers' Choice Award

2003 - 2006

∘ Shared Hosting ∘ Managed Dedicated Servers ∘ Managed Colocation ∘ Semi-Private Servers
∘ ColdFusion ∘ BlueDragon ∘ ASP ∘ .NET ∘ .Linux ∘ .Java ∘ SQL Server ∘ .MySQL ∘ Self-Healing Servers

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.

HostMySite.com

WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL