

COLD FUSION Developer's Journal

ColdFusionJournal.com

April 2003 Volume: 5 Issue: 4

Announcing...

2003 CFDJ
READERS'
CHOICE
AWARDS



VOTE NOW!

GO TO WWW.SYS-CON.COM

Editorial

Winds of Change

Robert Diamond page 5

CF Community

An Interview with Christian Cantrell and Sarge of Macromedia

Simon Horwith page 7

CFUGs

Complete Listing of ColdFusion User Groups

page 48

ColdFusion News

NTT DoCoMo to Embed Macromedia

Native Deployment of CFML on .NET Offered

page 50

RETAILERS PLEASE DISPLAY
UNTIL JUNE 30, 2003

\$8.99US \$9.99CAN



**SYS-CON
MEDIA**

Foundations:



Mistakes Fuseboxers Make

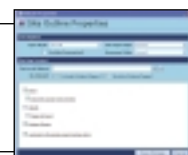
Hal Helms
page 32

Design Patterns in ColdFusion: Iterator Pattern PART 2 A simple yet powerful solution



Brendan O'Hara
10

Blueprints: It's Not ColdFusion – It's J2EE! Setting the record straight



Vince Bonfanti
18

<BF on CF>: Undocumented ColdFusion MX – 2 Being able to tweak XML configuration files can be a lifesaver

Ben Forta
22

CFCs: Reviving Super Calling overridden parent component functions from a child component



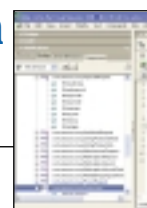
Samuel Neff
24

Custom Tags: Extending ColdFusion

Here's a tag that can help speed up the performance of your Web site 28

Raymond Camden

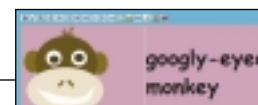
Journeyman CF: Exploring Amazon Web Services with ColdFusion MX They're there to serve us all



Charlie Arehart
36

Product Review: Site Executive 3.0 Content Management System

Web content management with fine style control



Jennifer Larkin

44

FUSETALK

www.fusetalk.com/new

IVIS TECHNOLOGIES

www.ivis.com

ACTIVEPDF
www.activepdf.com

editorial advisory board

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

editorial

editor-in-chief

Robert Diamond robert@sys-con.com

technical editors

Charlie Arehart charlie@sys-con.com
Raymond Camden raymond@sys-con.com

editorial director

Jeremy Geelan jeremy@sys-con.com

executive editor

Jamie Matusow jamie@sys-con.com

editor

Nancy Valentine nancy@sys-con.com

associate editors

Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com

assistant editor

Jennifer Stille jennifer@sys-con.com

production

production consultant

Jim Morgan jim@sys-con.com

art director

Alex Botero alex@sys-con.com

associate art directors

Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com

contributors to this issue

Charlie Arehart, Vince Bonfanti, Raymond Camden,
Ben Forta, Hal Helms, Simon Horwith,
Jennifer Larkin, Samuel Neff, Brendan O'Hara

editorial offices

SYS-CON MEDIA

135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645

TELEPHONE: 201 802-3000 FAX: 201 782-9600

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

is published monthly (12 times a year)

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

postmaster: send address changes to:

COLDFUSION DEVELOPER'S JOURNAL

SYS-CON MEDIA

135 Chestnut Ridge Rd., Montvale, NJ 07645

©copyright

Copyright © 2003 by SYS-CON Publications, Inc.

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

For promotional reprints, contact reprint coordinator.

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

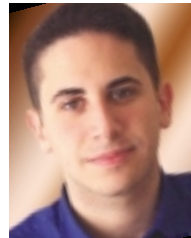
All brand and product names used on these pages are trade names, service marks, or trademarks of their respective companies.



Winds of Change

If you're at all like me (in which case, let me say "good luck to you!"), you spend a decent amount of your developer week traipsing around Macromedia's Web site. They've got an enormous wealth of information on there for those of us using CF, Flash, Dreamweaver, and the rest of the Macromedia product line as well.

They aren't of course providing us with all of these wonderful resources *just* because they're lovely people with very kind hearts. It's part of a focus on customer retention (they want us forever...FOREVER) and on cross-selling Macromedia's other products and services. ("That product works with that one? Who knew?!") However, all joking aside, with the



By Robert Diamond

unmatched volume of information combined with the blogs, forums, custom tag directories, and everything else I've neglected to list, it should be at the top of most CFers' bookmarks.

Macromedia's site is highly trafficked of course – more so than a lot of people realize. They report site traffic – per day! – as being 1 million customers, 250,000 software downloads, 4 million downloads of the Flash player, and just to make things more complicated for their Web team, they publish it in 12 different languages hitting all the major points of Macromedia's global presence.

They've changed a lot of things on the site, as you'll probably notice right off the bat.

Initial feedback on the various discussion groups seem to be mixed. Most of the changes are for the better, but nearly all of them will require some getting used to – and definitely some updates to a lot of the deeper link bookmarks that you might have. I know that we're definitely having some fun here at **CFDJ** with editors checking through all of our links on the **CFDJ** Web site to Macromedia content. (That's 550+ articles for those of you keeping track.)


Speaking of the **CFDJ** site, we're working on a serious upgrade to it here as well, adding in lots of new features and usability enhancements, and would welcome as much feedback as possible throughout the process. So, if there are features you'd like to see, that would make your life easier – suggest away! My e-mail address is here as always. . .

Also, by the time this issue is in your hands, voting will have begun in the 2003 **CFDJ**

Readers' Choice Awards, so make sure to hop on over to our Web site to cast your vote for the best products and services of the year. We've got some new public and behind-the-scenes procedures in place as part of our constant goal to reduce voter fraud (and you nasty voters out there know who you are!). As always we're striving to make this year's

awards even better than those that have come before. I think you'll be pleased with the results.

On a conference note, I'll see those of you attending CFUN 03, June 21 and 22 in Rockville, MD, as we present another **CFDJ** keynote panel discussing the latest issues of the day. For topics you'd like to see covered there, and subsequently in the magazine as well – shoot me on over an e-mail. For more information on the conference, visit www.cfconf.org/cfun-03.

Change and progress are usually for the better, as I think an ex-girlfriend told me (on her way out the door of course), and to stay competitive in today's world of Web sites, we developers must constantly strive to improve. That's a big personal theme of mine, and it's one that I hope you can see reflected in the content of the magazine and Web site. **CFDJ** is here to help. 

About the Author

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

robert@sys-con.com

NEW ATLANTA COMMUNICATIONS

www.newatlanta.com

president & ceo

Fuat A. Kircaali fuat@sys-con.com

business development

vp, business development

Grisha Davida grisha@sys-con.com

advertising

senior vp, sales & marketing

Carmen Gonzalez carmen@sys-con.com

vp, sales & marketing

Miles Silverman miles@sys-con.com

advertising director

Robyn Forma robyn@sys-con.com

advertising account manager

Megan Ring-Mussa megan@sys-con.com

associate sales managers

Carrie Gebert carrie@sys-con.com

Kristin Kuhnle kristin@sys-con.com

Alisa Catalano alisa@sys-con.com

sys-con events

president, events

Grisha Davida grisha@sys-con.com

conference manager

Michael Lynch mike@sys-con.com

customer relations

senior customer care/circulation specialist

Niki Panagopoulos niki@sys-con.com

manager, jdj store

Rachel McGouran rachel@sys-con.com

sys-con.com

vp, information systems

Robert Diamond robert@sys-con.com

web designers

Stephen Kilmurray stephen@sys-con.com

Christopher Croce chris@sys-con.com

online editor

Lin Goetz lin@sys-con.com

accounting

financial analyst

Joan LaRose joan@sys-con.com

accounts receivable

Kerri Von Achen kerri@sys-con.com

accounts payable

Betty White betty@sys-con.com

subscriptions

Subscribe@sys-con.com

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

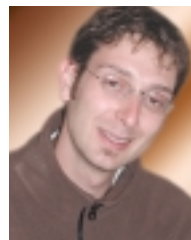
(U.S. Banks or Money Orders)

Back issues: \$12 U.S. \$15 all others

An Interview with Christian Cantrell and Sarge

This month I've chosen to deviate from the normal **Tales from the List** format and interview Christian Cantrell and Sarge – two Macromedia employees who are largely responsible for offering support to the ColdFusion developer community. Christian is Macromedia's ColdFusion community manager and Sarge is a high-tier technical support officer for Macromedia. I thought that speaking with the two of them and sharing their thoughts would help us developers better understand what Macromedia is doing to support our transition to the MX suite as well as give us some insight as to the direction the company is heading.

The interview left me feeling that Macromedia is indeed committed, now more than ever, to providing top-notch support to developers and institutions using their products, and that as they move forward, they are becoming more proactive in giving developers the tools they need to get the job done quickly and efficiently (in the form of product extensions). The questions are addressed to both Sarge and Christian unless otherwise stated.



By Simon Horwith

CFDJ: What is your official title at Macromedia, how long have you been there, and what were you doing before Macromedia?

Christian: My official title at Macromedia is server community manager. I've been in the position for about two months now. Before coming to Macromedia, I worked as a Java and ColdFusion application developer for a small startup in the ASP (application service provider) space.

Sarge: Senior product support engineer in Macromedia Worldwide MX Professional Services (MXPS). MXPS is a group within Macromedia Worldwide Server Product Support. Before joining Macromedia (as an Allaire consultant) I was an information engineer at Logicon, Inc., working as the manager of the GCSS Web/Portal Development Team. GCSS Web/Portal were the first ColdFusion applications to integrate with the DOD PKI, Java, and WDDX. They have been fielded in CONUS, Hawaii, Germany, and Korea.

CFDJ: What does your position entail?

Christian: My two primary responsibilities are making sure information flows back and forth between Macromedia and the server community, and making sure the DRK contains a full complement for server-related tools and utilities. That is something that has really changed since I started with

Macromedia; the DRK is becoming just as much a resource for server developers as it once was for Flash developers.

Sarge: The delivery of the MXPS support offerings, including installation, configuration, and migration assistance; application

—continued on next page

About the Author

Simon Horwith, senior consultant at Fig Leaf Software in Washington, DC, has been using ColdFusion since version 1.5. He is a Macromedia-certified Advanced ColdFusion and Flash developer and is a Macromedia-certified instructor. In addition to administering the CFDJList List Serve and presenting at DC-area CFUGs, Simon is a contributing author to Professional ColdFusion 5.0 (WROX) and to ColdFusion MX – The Complete Reference (McGraw-Hill), as well as technical editor of The ColdFusion 5.0 Certification Study Guide (Syngress).

shorwith@figleaf.com

mentoring; architecture and code reviews; performance analysis and tuning; and advanced CF system administration training.

CFDJ: *So far, what has been the most challenging aspect of what you do at Macromedia?*

Christian: I would say that the most challenging aspect of my job is keeping up with the entire ColdFusion community day to day. The ColdFusion community is immense in size, and they are both prolific and very creative. Keeping up with their ideas and their work is quite a challenge. Reading every post on every list and forum is not actually a requirement of my job, but I honestly love to see what people are creating and debating out there.

Sarge: Educating clients on CFML coding best practices/standards and the best methods of developing secure, highly available applications, and managing their expectations of what they are trying to achieve with ColdFusion. What I see more frequently today are network and system administrators with no ColdFusion experience tasked with ColdFusion administration. So the challenge is to provide them with the skills they need to effectively manage their ColdFusion servers. Meeting these challenges is very rewarding for me.

CFDJ: *Christian, the ColdFusion development community is close knit – is there anything that developers could be doing, but aren't, that would help Macromedia support the community?*

Christian: One thing I would encourage the community to do is use the wish-list form (www.macromedia.com/support/email/wishform). I can't tell you how many great ideas I have seen discussed on the lists that nobody has entered into the wish-list form. Everything people send to us is reviewed and discussed. If you have ideas out there, make sure you let us know what they are. Don't assume that just because you posted your idea to **CFDJ**List or cf-talk that Macromedia has a record of it.

CFDJ: *Christian, what, if anything, are Macromedia's plans for offering more support to the community through CFUGs, the Macromedia Web site, DRK releases, publications, online mail lists like the CFDJ List and CF-Talk, conferences, and/or Team Macromedia?*

Christian: The DRK is a big priority of mine. Starting with volume 3, I think ColdFusion developers will find the DRK to be an extremely important and useful resource. The other thing I would like to mention is DevNet. DevNet is a service that's really designed to give Macromedia developers more and better access to the resources they need to be competitive (you can read all about DevNet on Macromedia's Web site). Finally, keep an eye on our Weblogs. Although I don't have any specifics to offer right now, we are starting to think it's really time to take them to the next level.

CFDJ: *Sarge, what, if anything, is Macromedia planning to do/doing/done to improve support service?*

Sarge: We've increased our communications with the customers in order to ascertain how we can better serve them. We've redesigned our internal communication processes to better facilitate escalations and knowledge transfer. Management has upgraded our resources to enhance our ability to research, replicate, and resolve issues. They've redesigned our support plans and service offerings to provide more cost-effective solutions. They've also restructured the team to better align talent along the product lines.

CFDJ: *Sarge, how has the release of ColdFusion MX and the rest of the MX suite of products (Macromedia Studio MX, including Flash MX, Dreamweaver MX, Fireworks MX, FreeHand MX, and a developer edition of ColdFusion MX) affected your job?*

Sarge: It has created excitement and provided more challenges. The new CFMX platform definitely has the team buzzing about the new feature sets, particularly Macromedia Flash Remoting, Web services, and J2EE integration. My team's backgrounds are in development, so Macromedia Studio MX kind of got us back to our roots, as our first assignments were to go and build applications using Macromedia Studio MX. The challenge comes from the requirement to support CFMX and the underlying application servers: JRun, WebSphere, IPlanet, BEA WebLogic, etc.

CFDJ: *Sarge, what is the most common misinformation among developers with regards to Macromedia technical support?*

Sarge: I have heard that developers have asked for faster response time from product support. However, what developers need to understand is that we work closely with clients (remotely and on site) to replicate their issues in-house for detailed analysis and resolution. Issues will sometimes remain open until we can reliably duplicate them on our systems. This is the only way we can perform analysis without being on site. Once we have concrete results, we communicate them back to clients, in the forums, and via TechNotes. So it's not that we – or the company as a whole – do not care about

"One thing that has really impressed me about Macromedia is that although we are a relatively large company, we really tend to operate much more like a small entity"

—Christian Cantrell
Macromedia's ColdFusion community manager



—continued on page 43

NETQUEST

www.nqcontent.com

Design Patterns in ColdFusion: Iterator Pattern PART 2

A simple yet powerful solution

Last month I introduced design patterns, including the Template Method pattern and how it encourages polymorphism and helps remove the common switch-case constructs we normally utilize in custom tags for purposes of code reuse. This month our topic is the Iterator pattern, a simple yet powerful design pattern you can use to generically traverse through a custom collection CFC. You'll need to know what a collection is and exactly where the Iterator pattern comes into the picture. I'll go over Java Iterator syntax and demonstrate a ColdFusion implementation using CFCs. First, it's important to understand what a collection is and what it is useful for.

CFCs as Custom Collections

In most object-oriented languages a collection is simply an object that groups multiple other objects within a single container. In the Web world, collections are often useful for retrieving data from a database table or XML file, storing data somewhere in memory, allowing the data to be manipulated, and then updating the original table or file. Collections encourage software reuse by providing a standard interface for grouping "objects" and can provide algorithms to manipulate them. Java has a large number of collection classes, including Vector, HashMap, and ArrayList among others. Programmers can inherit from these classes or use them as data members when creating custom collection



By Brendan O'Hara

classes. ColdFusion has only a few collection data types (array, query, list, and struct) and until recently did not have a way to write custom collection classes. With the introduction of CFCs that's now possible.

I may want to write a generic CFC that encapsulates database access for the Employee table at my company. I will call it EmployeeCollection.cfc (see Listing 1). Initially, I give it a single method, Init(), that runs the basic query and loads the collection. I can add additional methods like Add() and Remove() to manipulate the data in the collection and therefore the Employee table. My collection will also have a single accessible data member, which is a query result set containing rows from the Employee table.

What Exactly Is an Iterator, Anyway?

An iterator, which is sometimes known as a cursor or enumeration, is an extremely powerful interface for accessing items from a collection one item at a time. The word "iterate" means "repeat or do many times." An "iterator" is therefore "something that repeats or does the same thing many times." Basically, it is a looping construct where you don't need to know the data structure in order to loop over it. The collection may store its internal data as a list, array, or query. It also takes the responsibility for traversal through collections of data away from the collections themselves and puts it into a standard iterator interface. Because it is separate from the collection itself, it also allows more than one iterator to exist at the same time. Additional iterators can be implemented that filter collections against certain criteria. An example of this may be an iterator implementation that only displays news articles based on defined preferences or one that filters corporate documents based on security level.

There are two general types of iterators: internal and external. In an internal iterator the iterator itself determines how the iteration takes place before applying some additional processing. The more common external iterator gives control of iteration to the programmer through a few standard methods.

Iterators and Enumerations Explained

Unlike some design patterns, the Iterator pattern has numerous implemen-

tations and variations in different languages. Those of you familiar with Java may know two similar but different types of interfaces (iterator and enumeration) used for navigating and traversing collections. These will give us a general idea of how we may implement iterators in ColdFusion.

In Java, the Enumeration interface was originally the preferred traversal method with just two methods:

- **HasMoreElements():** Tests if this collection contains more elements
- **NextElement():** Increments the Collections index and returns the next element

The Java enumeration interface has now been succeeded as the preferred method of collection traversal by the iterator interface. The Java iterator interface now contains the following methods, two of which are nearly identical to HasMoreElements() and NextElement():

- **HasNext():** Tests if this iterator contains more elements
- **Next():** Increments the iterator and returns the next element
- **Remove():** Removes the current element from the collection (optional)

Here we've only added the optional ability to remove items from a collection during traversal and marginally shortened the syntax. Let's take a look at a possible ColdFusion implementation using a base class named AbstractIterator.cfc and three derived classes – QueryIterator.cfc, ArrayIterator.cfc, and ListIterator.cfc. These will allow polymorphic iteration through lists, arrays, and query result sets.

A ColdFusion Iterator

AbstractIterator.cfc (see Listing 2) is a simple abstract CFC whose derived CFCs iterate various ColdFusion collections. The AbstractIterator.cfc contains the following methods:

- **HasNext():** Tests if this iterator's collection contains more elements
- **Next():** Increments the Iterators index and returns the next element

Just as in Java, this CFC's HasNext() method lets you test if there is at least one more element in the collection. This allows you to use a rarely used "conditional loop" with the following syntax, in

which "It" is the iterator CFC:

```
<cfloop condition="#It.hasNext()#">
    #It.next()#
</cfloop>
```

This syntax would be virtually identical whether you were iterating through an array, list, or query. Most ColdFusion collection data types contain objects that are all of the same data type (usually strings), but some can contain a multiplicity of data types. Let's look at some ordered ColdFusion data types, which are really collections. We'll see how we would normally traverse their internal data and how we would traverse their data using an iterator.

List

A list is a collection of string values (which can be numeric as well) separated by one or more delimiters.

```
<cfloop list=" Cat,Dog,Bull,Moose index="animal">
    <cfoutput># animal #</cfoutput>
</cfloop>
```

If you have a collection that stores information in a list it should return a ListIterator.cfc (see Listing 3). In the following example the variable "It" is a ListIterator for a list containing "Cat", "Dog", "Bull", and "Moose". Here is how you would access the list's data:

```
<cfloop condition="#It.hasNext()#">
    <cfoutput>#It.next()#</cfoutput>
</cfloop>
```

Array

A ColdFusion array is a numerically indexed collection of objects, which can also be of multiple data types. If you wanted to output the words "Cat", "Dog", "Bull", and "Moose", which are stored in myArray, you would do this:

```
<cfloop from="1" to="#ArrayLen(myArray)#"
    index="X">
    <cfoutput>#myArray[X]# </cfoutput>
</cfloop>
```

If you have a collection that stores information in an array it should return an ArrayIterator.cfc (see Listing 4). In the following example the variable "It" is an ArrayIterator for an array containing "Cat", "Dog", "Bull", and "Moose".

Here is how you would access the array's data:

```
<cfloop condition="#It.hasNext()#">
    <cfoutput>#It.next()#</cfoutput>
</cfloop>
```

Notice that this is exactly the same syntax we used for the List iterator. This is because using a polymorphic interface like iterator allows us to standardize syntax. This allows you to write custom tags, functions, and CFCs that take an iterator CFC as a parameter and do some processing on every item in the collection, or perhaps only items that fit some specific criteria. But what about the most common ColdFusion collection data type, the query result set?

Iterating Through a Result Set

In your earliest days with ColdFusion you learned how to call a query with a simple select statement, and navigate through and output the results of the query using CFOUTPUT or CFLOOP. It was simple, and although everything was coded inline, it worked – we've been using and manipulating query result sets ever since. If you have a query result set named "qEmployees", which contains the fields "firstname" and "lastname" for every employee in your company, you would do this:

```
<cfoutput query="qEmployees">
    #FirstName# #LastName#<br>
</cfoutput>
```

We may have graduated at some point to using <cfloop> so as not to nest one set of <cfoutput> tags within another set. If you have a collection that returns a QueryIterator.cfc (see Listing 5) named "It", which contains the fields "firstname" and "lastname" for every employee in your company, you would do this:

```
<cfloop condition="#It.hasNext()#">
    <cfset employee = It.next()>
    # employee.firstname# #
    employee.lastname#<br>
</cfloop>
```

The only difference between this and the ListIterator.cfc and ArrayIterator.cfc examples is that the QueryIterator.cfc returns a structure, which must be set into a variable and then used explicitly.

However, the `ArrayIterator.cfc` could just as easily return a structure since arrays can contain any data type.

Now that we have an idea of what an iterator interface is going to look like and what it should do, we need to look at `EmployeeCollection.cfc` and add the ability to access its employee data via an iterator.

Adding an Iterator() to a Collection CFC

When we last looked at `EmployeeCollection.cfc` it had only a single method `Init()`, which simply acted as a constructor to query the `Employee` table and build the `EmployeeCollection` CFC. Now we have to add an `Iterator()` method that returns a `QueryIterator.cfc` for accessing the collection's internal data. To traverse any collection using this syntax a CFC, which extends `AbstractIterator`, is returned by the `Iterator()` method.

The way I implement this interface across projects is that I require CFC collection objects, which store data such as arrays, result sets, and lists, to implement a single public method `Iterator()`, which returns an iterator for that collection. As I said above, filtered iterators can be optionally implemented allowing iteration over a subset of the collection or using a specific algorithm. You can "reset" an iterator after it's exhausted its

collection, or at any time, if you add an optional `Reset()` method. You can also create a new and distinct iterator instance for each usage by simply calling the `Collections.Iterator()` method again and assigning it to a different variable. Please note that although query result sets are passed by reference in ColdFusion MX, `QueryIterator.cfc` duplicates the result set so each iterator object is a wholly separate entity. Let's take a look at the `Iterator()` method we will add to `EmployeeCollection` CFC (see Listing 6).


```
<cffunction name="Iterator"access="public"

    returntype="com.myCompany.AbstractIterator"
    <cfset var myIterator = createObject
    ("component","com.myCompany.
    QueryIterator")>
    <cfinvoke component="#myIterator#"
        method="init"
        collection="#my.resultset#">
    <cfreturn myIterator>
</cffunction>
```

First, we instantiate a `QueryIterator.cfc` using the `createObject()` method. Then we call the `QueryIterator.cfc`'s own `Init()` method by passing in the local CFC variable `my.resultset` (which contains the `EmployeeCollection.cfc` result set). We then return the `QueryIterator.cfc` reference. Iteration happens exactly as our

previous `QueryIterator.cfc` example.

Create Your Own Iterator Implementations

The Iterator pattern and associated methods are very flexible and can be molded to match your performance and programmatic needs. In this article I've discussed what an iterator is and why they are useful. Try wrapping your database access in CFC collections and objects and create an iterator that fits your individual needs. Dissect the `AbstractIterator.cfc`, `QueryIterator.cfc`, `ArrayIterator.cfc`, and `ListIterator.cfc` to see how all this is being accomplished programmatically – you'll find it quite simple. The examples from these files and `IteratorExamples.cfm` (see Listing 7) will get you started. 

About the Author

Brendan O'Hara is one of the coauthors of Advanced Macromedia ColdFusion MX Application Development, published by Macromedia Press. Brendan has a Macromedia ColdFusion MX Developer Certification along with Java and Linux certifications from Penn State University. He works as a ColdFusion architect and Java developer for E-tech Solutions and Amkor Technology in the Philadelphia suburbs.

bohar@amkor.com

Listing 1: EmployeeCollection.cfc

```
<cfcomponent displayname="employeeCollection">

<cfproperty name="My"
type="struct"
    hint="Protected and Inheritable Instance Structure"
    required="yes" default="StructNew()">
<cfproperty name="My.DSN" type="any"
    hint="The DSN being queried to fill the collection"
    required="no" default="Employees">
<cfproperty name="My.resultSet" type="any"
    hint="The resultSet being iterated through"
    required="yes" default="">

<!--- PSEUDO CONSTRUCTOR CODE --->
<cfset My = structNew()>
<cfset My.DSN = "Employees">
<cfset My.resultSet = "">

<!--- BEGIN // INITIALIZATION FUNCTIONS --->

<cffunction name="init"
    access="public"
    output="false"
    Hint="Initializes Query object // always
    returns this for method chaining"
    returntype="struct">
    <cfargument name="department" required="false" type="string"
    default="">
    <cfargument name="sortOrder" required="false" type="string" default="">
    <!--- This is the actual implementation!!!
    <cfquery name="qInit" datasource="#My.DSN#"
        Select *
        FROM Employees
        <cfif arguments.department neq "">
            WHERE Department = '#arguments.department#'
        </cfif>
        <cfif arguments.sortOrder neq "">
            ORDER BY #arguments.sortOrder#
        </cfif>
    </cfquery>
```

COOLFUSION

www.coolfusion.com

```
// This is a demo implemntation!!!! --->
<cfset var qInit = querynew("location,firstName,lastName")>
<cfset queryaddrow(qInit, 3)>
<cfset QuerySetCell(qInit, "location", "Philadelphia", 1)>
<cfset QuerySetCell(qInit, "firstName", "Dave", 1)>
<cfset QuerySetCell(qInit, "lastName", "Duncan", 1)>
<cfset QuerySetCell(qInit, "location", "Louisville", 2)>
<cfset QuerySetCell(qInit, "firstName", "Elmo", 2) >
<cfset QuerySetCell(qInit, "lastName", "Smiggins", 2) >
<cfset QuerySetCell(qInit, "location", "San Diego", 3) >
<cfset QuerySetCell(qInit, "firstName", "Rita", 3) >
<cfset QuerySetCell(qInit, "lastName", "Morebeno", 3) >
<cfset My.resultSet = qInit>
<cfreturn this>
</cffunction>
```

```
<cffunction name="Iterator"
    access="public"
    returnType="struct">
    <cfset var myIterator =
createObject("component","com.myCompany.QueryIterator")>
    <cfinvoke component="#myIterator#"
        method="init"
        collection="#my.resultset#" />
    <cfreturn myIterator />
</cffunction>

</cfcomponent>
```

Listing 2: AbstractIterator.cfc

```
<cfcomponent displayName="AbstractIterator">

<cfproperty name="My"
type="struct"
    hint="Protected and Inheritable Instance Structure"
    required="yes" default="StructNew()">
<cfproperty name="My.collection" type="any"
    hint="The collection being iterated through"
    required="yes" default="">
<cfproperty name="My.index" type="numeric"
    hint="The current index of the my.collection"
    required="no" default="0">
<cfproperty name="My.length" type="numeric"
    hint="The length of my.collection"
    required="no" default="0">
<cfproperty name="My.currentItem" type="any"
    hint="The item at the current index of my.collection"
    required="no" default="">

<cfset my = structNew()>
<cfset my.collection = "">
<cfset my.index = 0>
<cfset my.length = 0>
<cfset my.currentItem = "">

<cffunction name="init"
    access="public"
    output="false"
    Hint="Initializes Iterator object //
always returns this">
    <cfabort showerror="Error: This Method is Abstract and must be over-
```

```
ridden">
</cffunction>

<cffunction name="hasNext"
    access="public"
    returnType="boolean">
    <cfif my.length GTE my.index + 1>
        <cfreturn true>
    <cfelse>
        <cfreturn false>
    </cfif>
</cffunction>

<cffunction name="next"
    access="public"
    returnType="any">
    <cfabort showerror="Error: This Method is Abstract and must be over-
ridden">
</cffunction>

</cfcomponent>
```

Listing 3: ListIterator.cfc

```
<cfcomponent displayName="ListIterator" extends="AbstractIterator">

<cfset my.delimiters = "">

<cffunction name="init"
    access="public"
    output="false"
    Hint="Initializes ListIterator object //
always returns this for method chaining"
    returnType="struct">
    <cfargument name="collection" type="string" required="true">
    <cfargument name="delimiters" type="string" required="false"
default=",">
    <cfset my.collection = arguments.collection>
    <cfset my.delimiters = arguments.delimiters>
    <cfset my.length = listlen(my.collection, my.delimiters)>
    <cfreturn this>
</cffunction>

<cffunction name="next"
    access="public" output="true">
    <cfif this.hasNext()>
        <cfset my.index = my.index + 1>
        <cfset my.currentItem = ListGetAt(my.collection, my.index ,
my.delimiters)>
    </cfif>
    <cfreturn my.currentItem>
</cffunction>

</cfcomponent>
```

Listing 4: ArrayIterator.cfc

```
<cfcomponent displayName="ArrayIterator" extends="AbstractIterator">

<cffunction name="init"
    access="public"
    output="false"
    Hint="Initializes ArrayIterator object //
always returns this for method chaining"
```


EDGE WEB HOSTING

www.edgewebhosting.net

```

                returntype="struct">
<cfargument name="collection" type="array" required="true">
<cfset my.collection = arguments.collection>
<cfset my.length = arraylen(my.collection)>
<cfreturn this>
</cffunction>

```

```

<cffunction name="next"
            access="public">
    <cfif this.hasNext()>
        <cfset my.index = my.index + 1>
        <cfset my.currentItem = my.collection[my.index]>
    </cfif>
    <cfreturn my.currentItem>
</cffunction>

</cfcomponent>

```

Listing 5: QueryIterator.cfc

```

<cfcomponent displayName="QueryIterator" extends="AbstractIterator">

<cffunction            name="init"
                    access="public"
                    output="false"
                    Hint="Initializes QueryIterator object //
always returns this for method chaining"
                    returntype="struct">
    <cfargument name="collection" type="query" required="true">
    <cfset my.collection = duplicate(arguments.collection)>
    <cfset my.length = my.collection.recordcount>
    <cfset my.currentItem = structNew()>
    <cfreturn this>
</cffunction>

<cffunction name="next"
            access="public">
    <cfif this.hasNext()>
        <cfset my.index = my.index + 1>
        <cfset Columns = listToArray(my.collection.columnList)>
        <cfloop from="1" to="#arraylen(Columns)#" index="field">
            <cfset my.currentItem[Columns[field]] =
my.collection[Columns[field]][my.index]>
        </cfloop>
    </cfif>
    <cfreturn my.currentItem>
</cffunction>

</cfcomponent>

```

Listing 6: EmployeeCollection_InitOnly.cfc

```

<cfcomponent displayName="employeeCollection">

<cfproperty name="My"
            type="struct"
            hint="Protected and Inheritable Instance Structure"
            required="yes" default="StructNew()">

<cfproperty name="My.DSN" type="any"
            hint="The DSN being queried to fill the collection"
            required="no" default="Employees">

<cfproperty name="My.resultSet" type="any"

```

```

            hint="The resultSet being iterated through"
            required="yes" default="">

```

```

<!--- PSEUDO CONSTRUCTOR CODE --->
<cfset My = structNew()>
<cfset My.DSN = "Employees">
<cfset My.resultSet = "">

<!--- BEGIN // INITIALIZATION FUNCTIONS --->
<cffunction            name="init"
                    access="public"
                    output="false"
                    Hint="Initializes Query object // always
returns this for method chaining"
                    returntype="struct">
    <cfargument name="department" required="false" type="string"
default="">
    <cfargument name="sortOrder" required="false" type="string" default="">
    <!--- This is the actual implemntation!!!! -->
    <cfquery name="qInit" datasource="#My.DSN#">
        Select *
        FROM Employees
        <cfif arguments.department neq "">
            WHERE Department = '#arguments.department#'
        </cfif>
        <cfif arguments.sortOrder neq "">
            ORDER BY #arguments.sortOrder#
        </cfif>
    </cfquery>
    // This is a demo implemntation!!!! --->
    <cfset var qInit = querynew("location,firstName,lastName")>
    <cfset queryaddrow(qInit, 3)>
    <cfset QuerySetCell(qInit, "location", "Philadelphia", 1)>
    <cfset QuerySetCell(qInit, "firstName", "Dave", 1)>
    <cfset QuerySetCell(qInit, "lastName", "Duncan", 1)>
    <cfset QuerySetCell(qInit, "location", "Louisville", 2)>
    <cfset QuerySetCell(qInit, "firstName", "Elmo", 2)>
    <cfset QuerySetCell(qInit, "lastName", "Smiggins", 2)>
    <cfset QuerySetCell(qInit, "location", "San Diego", 3)>
    <cfset QuerySetCell(qInit, "firstName", "Rita", 3)>
    <cfset QuerySetCell(qInit, "lastName", "Morebeno", 3)>
    <cfset My.resultSet = qInit>
    <cfreturn this>
</cffunction>

</cfcomponent>

```

Listing 7: IteratorExamples.cfm

```

<!--- Calling an Iterator for a Collection of Employees--->
<cfset SalesPeople =
CreateObject("Component","com.myCompany.EmployeeCollection") />
<cfset SalesPeople.init("Sales","lastName") />
<cfset SalesPeopleIterator = SalesPeople.Iterator()>
<cfoutput>
<cfloop condition="#SalesPeopleIterator.hasNext()#">
    <cfset salesPerson = SalesPeopleIterator.next()>
    #salesPerson.location#: #salesPerson.firstName#
    #salesPerson.lastName#<br>
</cfloop>
</cfoutput>

```

Download the Code...
Go to www.coldfusionjournal.com

MACROMEDIA

www.macromedia.com/go/devnet-cfdj

It's Not ColdFusion – It's J2EE!

Setting the record straight

Over the past few months I've had several conversations with developers that have all been pretty much the same. Here's a paraphrase of a recent one: "My management decided to standardize on *<insert your favorite J2EE server here>*, and my company isn't going to use ColdFusion anymore."

"They're telling me I have to convert my existing CFML applications to run on the J2EE server so they can retire the old ColdFusion 4.5 and 5.0 servers."

"The server admins say I have to package my applications into standard J2EE WAR files for deployment. I've looked at BlueDragon – the J2EE edition – and it seems to do everything I need. But when I told my management about it they said, 'Sorry, we've already said that you can't use ColdFusion – you'll have to rewrite it in JSP so that it'll be real J2EE.'"

"How am I going to rewrite everything in JSP? I don't know JSP! And it's going to take forever – I have several applications and some of them are made up of hundreds of CFML pages and custom tags! And how am I going to make all these enhancements they want while I'm rewriting everything? What am I going to do?"

My answer to this developer is to first educate his management. There are two important pieces of misinformation in their objection to the use of BlueDragon: (1) it's not ColdFusion (it's CFML); and (2) it's real J2EE. The rest of this column addresses these issues, particularly the second one, and provides details of the BlueDragon architecture to support the claim that it's "real J2EE."



By Vince Bonfanti

Having answered his managers' objections, he should point out to them that the time and costs of rewriting are unnecessary; BlueDragon allows them to migrate their CFML applications to their J2EE server relatively quickly and at a much lower cost than rewriting them in JSP. Finally, having migrated their CFML

applications to a standard J2EE environment, they're free to enhance the applications using either CFML or JSP, or even to slowly rewrite the application in JSP one page at a time if that's what they choose to do.

CFML is Not ColdFusion

This may seem obvious once you "get it," but it's very hard for some people to separate CFML from ColdFusion because they've been synonymous for so long. CFML is a tag-based markup and server-side scripting language for developing dynamic Web content. ColdFusion is the name of a server product from Macromedia, Inc., that implements CFML. They're not the same thing.

Consider this: there are two completely different implementations of CFML in server products from Macromedia, both named "ColdFusion." One is the original C/C++ implementation embodied in

ColdFusion 5.0, the other is ColdFusion MX, which is not only written in the Java programming language, but is based on a radically different architecture from the original C/C++ version. The fact that the inventors of CFML can create two very different implementations illustrates the separation of the CFML language from the ColdFusion server (even though Macromedia chose to name both of their implementations *ColdFusion*).

Similarly, BlueDragon is a server product that implements CFML using an architecture that differs from both the original C/C++ ColdFusion 5.0 and the Java-based ColdFusion MX. BlueDragon is CFML. BlueDragon is not ColdFusion. CFML is not ColdFusion.

Why is this distinction important? If we go back to the managers' objections, one is, "We've already decided that we're not using ColdFusion anymore." Often, this objection is to the ColdFusion server product and not to the CFML language, but the distinction between ColdFusion and CFML has not been clearly made. Therefore, the rejection of ColdFusion becomes a rejection of CFML. This doesn't have to be the case.

The BlueDragon architecture allows you to integrate CFML into J2EE Web applications as a legitimate, standard, portable, fully compliant presentation-layer alternative to JavaServer Pages (JSP). Therefore, because you can use BlueDragon to deploy CFML in "real J2EE" applications, a rejection of ColdFusion (the server) does not have to be a rejection of CFML (the language).

BlueDragon is Real J2EE

Java servlets are the core Java 2, Enterprise Edition (J2EE) platform tech-

nology for creating dynamic Web content. JSP pages are translated and compiled into Java servlets at runtime by the J2EE server; furthermore, on many J2EE servers, the translate-compile-execute processing of JSP pages is implemented by a Java servlet. J2EE Web applications, which are supported by all compliant J2EE servers, are defined by the Java servlet specification. Java servlets are as “real J2EE” as it gets.

The BlueDragon CFML runtime is implemented as a standard Java servlet. As such, it can be packaged into a standard J2EE Web application and deployed onto any compliant J2EE server. Web applications containing the BlueDragon servlet have been successfully deployed – without any modifications to the Web application, BlueDragon servlet, or J2EE server – to run CFML pages on BEA WebLogic, IBM WebSphere, Sun ONE Application Server, Oracle 9i Application Server, Borland Enterprise Server, JBoss, Macromedia JRun, New Atlanta ServletExec, and Apache Tomcat. This is what makes BlueDragon “real J2EE.” It is entirely standards-based, is fully compliant with all relevant J2EE specifications, and is completely portable across J2EE servers.

Here are some questions I’d ask someone who thinks CFML via BlueDragon isn’t real J2EE: if you write your own Java servlet and deploy it within a Web application onto a J2EE server, is that real J2EE? If you create a Web application using the servlet-based Struts Framework, is that real J2EE? If you create a Web application using a servlet-based templating system such as WebMacro or Velocity, or a servlet-based XML/XML transformer, is that real J2EE? If the answer to any of those questions is “Yes” (hint: the answer to all of them is “Yes”), then it becomes clear that the BlueDragon servlet-based CFML runtime is also “real J2EE.”

In order to understand this better, let’s take a more detailed look at what a Java servlet is, what a J2EE Web application is, and how to create standard J2EE Web applications that support CFML pages using the BlueDragon servlet.

Java Servlets

Java servlets are defined by the `javax.servlet.*` and `javax.servlet.http.*` packages (in Java, packages are collections of classes that perform similar or

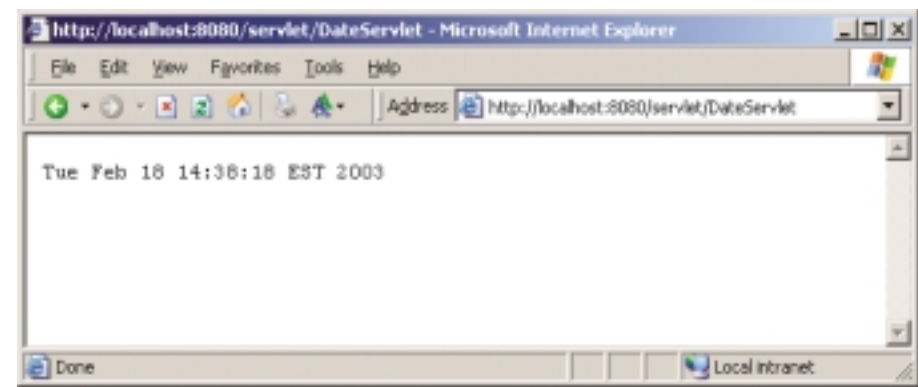


Figure 1: The output produced by the DateServlet in the browser

related functions, or that depend on each other to provide a common set of functions). In order to implement a Java servlet you need to do two things:

1. Create a Java class that extends `javax.servlet.http.HttpServlet`.
2. Implement the `service()` method of your class to receive HTTP requests and send HTTP responses.

The `service()` method of the `HttpServlet` class receives two parameters: an `HttpServletRequest` that contains all of the information about a request from the browser (including HTTP headers, URL arguments, and POST data from form submissions), and an `HttpServletResponse` that is used to send responses back to the browser. Here’s a complete Java servlet that simply returns the current date and time back to the browser:

```
import javax.servlet.*;
import javax.servlet.http.*;

public class DateServlet extends HttpServlet
{
    public void service( HttpServletRequest
        request,
        HttpServletResponse response )
        throws ServletException,
        java.io.IOException
    {
        // Set the response content type to
        // text/plain
        response.setContentType( "text/plain"
    );

        // Send the current date and time to
        the client
        response.getWriter().println( new
            java.util.Date().toString() );
    }
}
```

This column is about BlueDragon for J2EE servers, but if you’ve installed BlueDragon Server JX (the stand-alone server version of BlueDragon that includes servlet and JSP support), then you can see the output of the DateServlet by entering the following URL in your browser:

<http://localhost:8080/servlet/DateServlet>

The output produced by the DateServlet in the browser is illustrated in Figure 1.

If you don’t have BlueDragon Server JX installed, you’ll need to compile and deploy the DateServlet to a servlet container (instructions for which are beyond the scope of this column).

Because the BlueDragon CFML runtime is implemented as a standard Java servlet, it has the same characteristics as the DateServlet: it extends `HttpServlet` and implements the `service()` method. Of course, the BlueDragon `service()` method is a little more complicated than the DateServlet!

Some servlet containers, such as the one in BlueDragon Server JX, allow you to deploy Java servlets simply by copying the class files (the files generated by compiling Java source code) to a special “servlets” directory. However, the standard J2EE way of doing things is to deploy servlets (and other content) within “Web applications” that conform to a particular set of rules.

J2EE Web Applications

According to the Java Servlet 2.3 specification, “a Web application is a collection of servlets, HTML pages, classes, and other resources that make up a complete application on a Web server. The

Web application can be bundled and run on multiple containers from multiple vendors.” A J2EE Web application is characterized by a specific directory structure, and a configuration file named `web.xml` that is also referred to as the Web application deployment descriptor.

Content that is to be served to the client (HTML, JSP, and CFML pages; GIF and JPEG images; etc.) is placed directly in the top-level directory of a Web application. A Web application may contain subdirectories within the top-level directory; for example, it may contain an images sub-directory to hold GIF and JPEG files.

Within the Web application top-level directory is a special subdirectory named `WEB-INF`. The J2EE server will not serve any content from the `WEB-INF` subdirectory to the client; therefore, this is the place to put configuration files, Java class files, or other resources that need to be protected. The `web.xml` deployment descriptor is placed directly within the `WEB-INF` sub-directory.

The `web.xml` deployment descriptor contains configuration information used by the J2EE server to support the Web application. For example, the `web.xml` file provided with BlueDragon contains configuration information that tells the J2EE server how to process CFML files (specifically, it instructs the J2EE server to forward all URLs that end with the `.cfm` extension to the BlueDragon CFML runtime servlet).

There are two special subdirectories within the `WEB-INF` directory: the `classes` subdirectory that is used to hold unbundled Java `.class` files, and the `lib` subdirectory that is used to hold Java `.jar` archives. Any Java classes placed in these subdirectories are automatically available to the Web application (that is, they don't need to be added to the J2EE server's classpath or otherwise configured in any way).

In summary, the key features of a J2EE Web application are:

- Content that is to be served to the client (HTML, GIF, JPEG, CFM, JSP, etc.) is placed directly within the Web application top-level directory, or within subdirectories of the top-level directory.

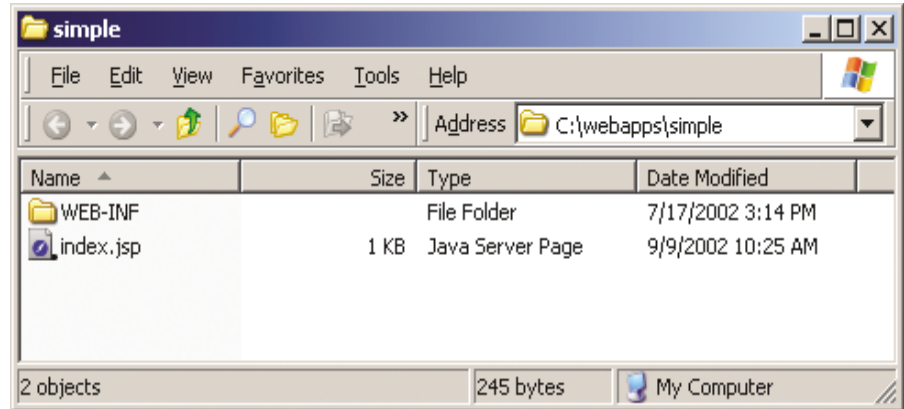


Figure 2: A simple J2EE Web application

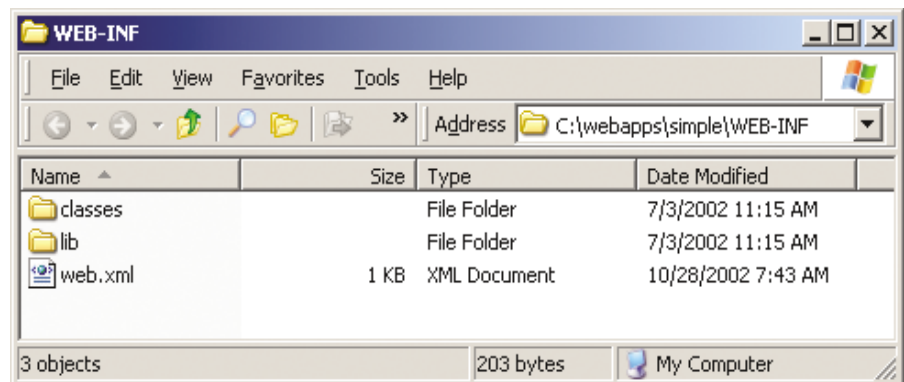


Figure 3: The `WEB-INF` subdirectory contains the `classes` and `lib` subdirectories and the `web.xml` deployment descriptor

- The `WEB-INF` subdirectory is located within the Web application top-level directory. The J2EE server will not serve any files from `WEB-INF` to the client.
- The `web.xml` deployment descriptor is located directly within the `WEB-INF` directory.
- The `classes` and `lib` subdirectories within `WEB-INF` are used to store Java `.class` and `.jar` files, respectively.

Deploying CFML in a J2EE Web Application

In order to deploy CFML pages onto a standard J2EE server, start with a basic J2EE Web application, add and configure the BlueDragon CFML servlet, then add your CFML pages and deploy!

Figures 2 and 3 illustrate a simple J2EE Web application. The Web application top-level directory is located at `C:\webapps\simple`, which contains the `WEB-INF` subdirectory and a single content file, `index.jsp`. The `WEB-INF`

subdirectory contains the `classes` and `lib` subdirectories (which are empty), and the `web.xml` deployment descriptor. If you were to examine the contents of `web.xml`, you'd find the following:

```
<web-app>
</web-app>
```

In other words, the `web.xml` deployment descriptor is empty (by definition, J2EE Web applications “know” how to process JSP pages, so no configuration is needed, see Figures 2 and 3).

This is a complete, valid J2EE Web application that is ready to be deployed on any compliant J2EE server. In order to process CFML pages within this Web application, we need to do two things:

1. Copy the `BlueDragonJ2EE.jar` file into the `WEB-INF\lib` subdirectory. The `BlueDragonJ2EE.jar` file contains the BlueDragon CFML servlet and supporting classes.

2. Add configuration information to web.xml related to the BlueDragon CFML servlet. The key configuration entries to be added to web.xml are:

```
<servlet>
  <servlet-name>tagServlet</servlet-name>
  <description>BlueDragon CFML
Engine</description>
  <servlet-
class>com.naryx.tagfusion.cfm.cfServlet</servl
et-class>
</servlet>

<servlet-mapping>
  <servlet-name>tagServlet</servlet-name>
  <url-pattern>*.cfm</url-pattern>
</servlet-mapping>
```

These web.xml entries tell the J2EE server that there's a servlet implemented by the class com.naryx.tagfusion.cfm.cfServlet and gives it the name "tagServlet". It then tells the J2EE server to forward all requests that end with ".cfm" to "tagServlet" for processing.

That's it! You can now add CFML pages (with the ".cfm" extension) to the

Web application and deploy them onto any standard J2EE server. (Well, that's not really all there is to it, but those are the main points. There are some additional JAR files needed by BlueDragon, such as JDBC drivers for database access, and some additional directories within WEB-INF, such as the "custom-tags" directory. If you really want to try it, go to New Atlanta's Web site – www.newatlanta.com – and download BlueDragon for J2EE servers).

WAR Files

Finally, a Web application can be deployed unbundled in an open (or exploded) directory structure, or bundled into a Web ARchive (WAR) file. A WAR file is simply a Web application directory structure bundled into a ZIP file and given the ".war" extension. WAR files can be created using any utility that can create a ZIP file, such as WinZip on Windows, gzip on UNIX, or the JDK jar utility. WAR files provide a convenient package for deploying J2EE Web applications as a single component (file).

Conclusion

The BlueDragon CFML runtime is implemented as a standard Java servlet that conforms to all of the relevant J2EE specifications. The BlueDragon CFML runtime servlet can be configured in a standard J2EE Web application, which can be used to deploy CFML pages onto any compliant J2EE server. BlueDragon: it's not ColdFusion, it's CFML as "real J2EE"!



About the Author

Vince Bonfanti is president and cofounder of New Atlanta Communications, developers of Java- and CFML-based server products. A charter member of Sun's JavaT Servlet API and JavaServer PagesT Expert Groups, Vince has been a JavaOne speaker and a contributor to Java trade magazines and online publications. He has also been a featured speaker at Toronto's CFNorth and Washington's CFUN conferences as well as at local ColdFusion User Groups around the country.

vince@newatlanta.com

CFDYNAMICS

www.cfdynamics.com

Undocumented ColdFusion MX – 2

Being able to tweak XML configuration files can be a lifesaver

Several months ago I wrote a column entitled “Undocumented ColdFusion MX – 1,” (*CFDJ*, Vol. 4, issue 9). I appended a “1” to the title in anticipation of there being a “2” at some later date. Well, that time has come. That column (which apparently was highly controversial and upset quite a few readers, but fortunately made even more happy) exposed and explained the use of the ColdFusion factory object. This time we’ll look at some of the configuration files used by ColdFusion MX – files that can be tweaked as needed (and no controversy this time, I think).

Caveat emptor: Undocumented means unsupported – you may access and tweak the files discussed here, but you do so at your own risk. Undocumented features and files may be changed or removed in future versions of products. That’s what distinguishes the documented from the undocumented – vendors are under no obligation to support anything undocumented. Use at your own risk.



By Ben Forta

Introduction

Unlike previous versions of ColdFusion that stored all sorts of configuration options and settings in the Windows registry (or a registry emulator on Unix machines), most ColdFusion MX configuration options are stored in XML files. There are lots of benefits to storing configuration information in XML files, but for ColdFusion developers, two benefits stand out:

- Tweaking the registry was painful (and rather scary); XML files are easily (maybe too easily) read and accessed (by both man and machine).
- Unlike the registry, XML files are suitable for use on all operating systems.

But before you even touch ColdFusion’s configuration files, here are a few rules to keep in mind:

- Always make backups.
- Some of the ColdFusion XML files are well documented (in embedded comments), others are not. If you can’t find the documentation you need, well, you probably won’t. With one exception, the files that configure the integrated JRun server (and HTTP server) are documented in the

JRun documentation (but not the ColdFusion documentation).

- If you do make changes you’ll usually need to restart ColdFusion for those changes to take effect.
- Use a real editor; XML is picky, if you drop an ending tag or mess up nesting or lose a quote, well, things will break. So, be safe, and use an editor that is XML aware.

One final note – the files and directories referred to here apply to ColdFusion MX. If you are using ColdFusion MX for J2EE, then paths and files will likely differ.

XML Files, Mostly

Ready to look at XML files? Great, but first, one non-XML file that you should be aware of. The lib directory under the ColdFusion root contains a file named password.properties. This file contains the ColdFusion Administrator and RDS

passwords. You can’t change the passwords in this file directly, but if you do forget your ColdFusion Administrator password, you can use this file to gain entrance. Here’s how:

1. Open the password.properties file in a text editor.
2. Delete the password and rdspassword values (all text after the =).
3. Change the value of encrypted to false.
4. Save the file.
5. Restart ColdFusion. You’ll now be able to access the ColdFusion Administrator to set a new password.

There are other .properties files in the lib directory, but you’ll find them to be of little interest.

The web.xml File

The web.xml file is an important one. It is found in the WEB-INF directory under the Web root. The top of the file contains several application parameters in <context-param> sections. One of the parameters is named coldfusion.compiler.saveJava and, as its name suggests, if set to true, will force the ColdFusion compiler to save the generated Java source code (instead of just the compiled bytecode). You’ll not want to ever edit the .java files directly, but if you are curious as to what the ColdFusion-generated Java code looks like (or if you need help proving to some naysayer that ColdFusion really is a Java compiler) then set this parameter to true.

This file also contains a series of servlet mappings, each in a section named <servlet-mapping>. These specify the file extensions that ColdFusion should process. You’ll not want to change any of the default mappings, but if you want ColdFusion to process additional file extensions then you’ll want to add those mappings here. The safest way to do this is to copy an entire section (from <servlet-mapping> to </servlet-mapping>) and then just edit the <url-pattern> as needed.

You can also use web.xml to completely disable ColdFusion RDS if needed (many administrators opt to do this on production servers as a security precaution). Prior to ColdFusion MX, RDS could be disabled by simply not running the RDS service; in ColdFusion MX it's a little trickier. To disable RDS, locate the `<servlet-mapping>` for the servlet named `RDSServlet`, and comment out the entire section using `<!--` and `-->` (from before `<servlet-mapping>` until after the matching `</servlet-mapping>`).

web.xml also contains a list of default files (or welcome files). These are the files that will be used if a URL contains a directory name but not an explicit file name. The default file names are `index.cfm`, `index.html`, and `index.htm`, and are listed in a section entitled `<welcome-file-list>` (each file is specified as a `<welcome-file>`). You may add file names to this list if needed. Just remember that files are located and used in the order defined.

Another interesting configuration option in this file is the `<error-page>` section, which is used to map specific HTTP errors to CFM files that are used when the error occurs. By default, only HTTP error code 500 is mapped, but you can add your own mappings if you so wish.

The jrun-web.xml File

The `jrun-web.xml` file is in the same directory as `web.xml`. You'll only want to edit this file if you change the location of the ColdFusion document root after installation. `jrun-web.xml` contains a series of `<virtual-mapping>` sections, and to change the document root, edit the `/*` mapping as needed.

If you do move the document root, be sure to move the `CFIDE` directory to the new document root or you'll not have access to the ColdFusion Administrator, the CFC viewer, `<CFFORM>` .js files, and more.

The jrun.xml File

Another extremely important file is the `jrun.xml` file, which is under the ColdFusion root in the `\runtime\servers\default\SERVER-INF` directory. As its name suggests, `jrun.xml` is used to configure ColdFusion's integrated JRun server. Be extremely careful in this file, you'll not want to make changes without really knowing what you're doing.

But there is one section that you should be aware of. The file contains `<service>` sections used to configure the various services used by JRun (even if you are not using JRun directly). One of those services is the integrated HTTP server that's configured in the section named `jrun.servlet.http.WebService`. If you want to change the port that the integrated HTTP server runs on (the default is 8500) or want to disable it altogether (deactivate it), this is where you can do that.

Another extremely important setting in `jrun.xml` is the `cacheRealPath` parameter for the proxy service. If you have multiple Web sites on a single host, you must turn this option off (or you will end up with the wrong files being served when file names conflict). Unfortunately, by default, `cacheRealPath` is true, so if you do have multiple Web sites (as most of us do), edit this file, locate the `<service>` named `ProxyService`, find the attribute named `cacheRealPath`, and set it to false.

The default-web.xml File


`default-web.xml` is in the same `SERVER-INF` directory as `jrun.xml`. This file defines servlets used by JRun, one of which is the `FileServlet`. This servlet has a configuration parameter named `browseDirs` that controls file browsing (the behavior whereby if a URL is requested

without an explicit file name specified, then a directory of files can be displayed for selection). The default is true (display file list for browsing and selection); set it to false to prevent file browsing.

What Next

As you can see, ColdFusion's configuration files are extremely important, and whether you need to tweak them or not, you should be aware of what they are and what they do. Of course, most of what they do is not for us mere mortals to know, but here's a hint – scan the Macromedia Support TechNotes for articles on JRun's configuration files. Much of what you'll find there applies to ColdFusion MX as well.

Summary

ColdFusion MX made the move from proprietary to standards based, and with that move came XML configuration files that control just about every aspect of ColdFusion (and its underlying JRun server). As long as you proceed with caution, being able to tweak these files can be a lifesaver. 

About the Author

Ben Forta is Macromedia's senior product evangelist and the author of numerous books, including ColdFusion MX Web Application Construction Kit and its sequel, Advanced ColdFusion MX Application Development, and is the series editor for the new "Reality ColdFusion" series. For more information visit www.forta.com.

ben@forta.com

**PACIFIC
ONLINE**
www.pacificonline.com

Reviving Super

Calling overridden parent component functions from a child component

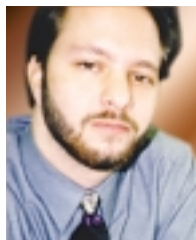
Components bring to ColdFusion MX developers much of the power and practices of object-oriented programming. The component implementation in ColdFusion MX introduces us to many of the principles of OOP, including encapsulation and inheritance. One aspect of these features is the ability to create new components that inherit or extend the functionality of other components.

In order to effectively extend functionality, developers often require the ability to override existing component functions as well as still call the original functions to incorporate their programming logic. Since ColdFusion MX doesn't currently provide this ability, we'll explore a technique to work around this limitation that enables us to call a parent component's overridden method.

Component-Based Development

With component-based development, related code can easily be grouped together in a consolidated entity, giving us the ability to encapsulate data and functionality. This consolidation can be further expanded with the ability to inherit functionality from a parent component, also called extension. When one component extends another, it can both reuse the functionality of the original component and add its own functionality.

Often, the new functionality added to a component will replace its original functionality. This is an advantage to developers who utilize the component



By Samuel Neff

in their code since the parent and child components will each have a similar call interface, or function names. By maintaining a consistent interface, developers can write code that can use either the parent or child component interchangeably without modifying the calling code.

For an example of this, see Listing 1, which sets up two components, GreetingBase.cfc and GreetingChild.cfc. GreetingChild extends the functionality of GreetingBase, and both provide a single function, sayGreeting, which returns a greeting to the caller. The caller can use either component and call the same function to retrieve the result appropriate for the component being used, a feature known in OOP as polymorphism.

Calling Overridden Parent Component Functions

We run into a limitation when we try to utilize the parent component functionality for a function that we've overridden, in this case sayGreeting. We want to add a function sayAllGreetings, which

returns the results of both the parent and the child components' sayGreeting functions. While you can create an instance of the parent component and call the new instance's sayGreeting method, this would not return accurate results in cases where the call depends on data in the component instance. Unfortunately, there is no mechanism in ColdFusion MX to call an overridden parent method and have it run in the context of the calling component instance.

Many other OOP languages, such as Java and even ActionScript, allow calling a parent class's methods through the keyword Super. While ColdFusion MX does not provide this ability, there is an often-overlooked feature that we can use to achieve the desired effect.

ColdFusion stores function references and variables equally. A function can be called by referring to the function name followed by parentheses, as you are accustomed. Additionally, the function reference can also be utilized by simply referring to the function name. The most common utilization of this feature is to copy a function to another scope, and here we'll use this ability to make a parent component's functions available to its children.

Any code that exists within the cfcomponent tags and outside of any cffunction tags is executed when the component is instantiated. This code is most often used to set up or initialize the component and is called the Constructor in OOP terms.

```
<cfcomponent>
  <!-- constructor code goes here -->
  <cffunction name="...">
  </cffunction>
</cfcomponent>
```

In Listing 2 we added some code to our original components. GreetingBase.cfc now has two lines of code in the constructor.

```
<cfset super = structNew()>
<cfset super.sayGreeting = sayGreeting>
```

What we've done here is set up a structure called "super", which we'll use as a holder for our parent component methods. After creating the structure we use cfset to copy a reference to the function to our structure. It's important to note here that when this code is run, the component will exist only as the parent component, so any functionality that is overridden in the child component has not yet been incorporated.

With this structure created and our parent functions safely stored, we can utilize them in the child, GreetingChild.cfc, as follows:

```
<cfset var greetings="Parent says '" &
    super.sayGreeting() &
    "', Child says '" &
    sayGreeting() & "'">
```

In this code we put together a string in which we call our parent component's sayGreeting function, through our super structure, and also call the current sayGreeting function as we normally would.

Super Structure Limitations

Through a little creative utilization of obscure features, we've been able to implement a very important feature common to OOP. However, our implementation is not comprehensive and does have limitations.

This technique can only be used to implement one level of parent-component calls. Since each component in the inheritance chain does not have its own data location where we can store the parent component's methods, we can only store the methods from a single parent component.

Additionally, we must recognize that this is a workaround to provide for functionality missing from ColdFusion MX. Since this feature is standard for OOP and commonly

requested from developers, Macromedia will probably implement this functionality natively in a future version of ColdFusion.

When adding support for parent-component calls, Macromedia is likely to use the keyword "super" in their own implementation since this is used in both Java and ActionScript. With this change, our code will stop working since "super" will become a keyword. While this may seem like a problem, it is actually an advantage.

Since we no longer need our workaround once parent component calls are implemented in ColdFusion, we want to ensure that we can easily remove this workaround when it becomes obsolete. To accomplish this, wrap all of the super workaround code, the parent-component constructor, inside consistent comments as shown in Listing 3.

With these comments, we can utilize the Regular Expression replace in either ColdFusion Studio or Dreamweaver to remove all of the super structures from our code. As long as the keyword super


HOSTMYSITE.COM

www.hostmysite.com

is used for this functionality natively, the rest of our code will continue to work.

Conclusion

Component-based development brings to ColdFusion MX many of the features of OOP. By using a structure and

function references, we were able to bring a common feature of OOP to ColdFusion, albeit with limitations. In recognizing these limitations, we can write our code in a forward-thinking manner and facilitate future changes to the code when they come in conflict with possible new functionality. 

About the Author

Samuel Neff is a senior software engineer with B-Line Express in the Washington, DC, area. He is Advanced ColdFusion 5.0 Certified and a Team Macromedia Volunteer for ColdFusion.

sam@blinex.com

Listing 1:

Parent and Child components each with their own sayGreeting function

```
<!--- GreetingParent.cfc --->
<cfcomponent displayName="Greeting Parent Component">
    <cffunction
        name="sayGreeting"
        returnType="string">
        <cfset var greeting="Hello">
        <cfreturn greeting>
    </cffunction>
</cfcomponent>

<!--- GreetingChild.cfc --->
<cfcomponent
    displayName="Greeting Child Component"
    extends="GreetingParent">
    <cffunction
        name="sayGreeting"
        returnType="string">
        <cfset var greeting="Hi there">
        <cfreturn greeting>
    </cffunction>
</cfcomponent>
```

Listing 2:

Parent and Child classes each with their own sayGreeting functions

```
<!--- GreetingParent.cfc --->
<cfcomponent displayName="Greeting Parent Component">
    <!--- set up a structure to hold super class methods --->
    <cfset super = structNew()>
    <cfset super.sayGreeting = sayGreeting>
    <cffunction
        name="sayGreeting"
        returnType="string">
        <cfset var greeting="Hello">
        <cfreturn greeting>
    </cffunction>
```

```
</cfcomponent>

<!--- GreetingChild.cfc --->
<cfcomponent
    displayName="Greeting Child Component"
    extends="GreetingParent">
    <cffunction
        name="sayGreeting"
        returnType="string">
        <cfset var greeting="Hi there">
        <cfreturn greeting>
    </cffunction>
    <cffunction
        name="sayAllGreetings"
        returnType="string">
        <cfset var greetings="Parent says '" &
                                super.sayGreeting() &
                                "', Child says '" &
                                sayGreeting() & "'">
        <cfreturn greetings>
    </cffunction>
</cfcomponent>
```

Listing 3:

Parent component with comments added to facilitate removing the super structure

```
<!--- GreetingBase.cfc --->
<cfcomponent displayName="Greeting Parent Component">
    <!--- start super hack --->
    <cfset super = structNew()>
    <cfset super.sayGreeting = sayGreeting>
    <!--- end super hack --->
    <cffunction
        name="sayGreeting"
        returnType="string">
        <cfset var greeting="Hello">
        <cfreturn greeting>
    </cffunction>
</cfcomponent>
```

Download the Code...
Go to www.coldfusionjournal.com

TERATECH

www.cfconf.org/cfun-03/

Extending ColdFusion

Here's a tag that can help speed up the performance of your Web site

Welcome to another edition of Extending ColdFusion. In the previous installments, we dealt with ColdFusion user-defined functions, or UDFs. In this month's article, we're going to talk about a custom tag. With the introduction of CFCs and the various methods of using UDFs in ColdFusion 5 and MX, custom tags have kind of been forgotten. However, they are still very powerful and can be a great help to your application.

The custom tag we're going to look at is called scopeCache.

By Raymond Camden

is based on a similar one used in the Spectra product, although this one is far simpler.

scopeCache takes three arguments:

- 1. name:** This is the name of the cached content. If you were caching the news portion of the home page, you may decide to use the name, "news/home."
- 2. scope:** This is the scope to store the cache in. The options are: session, application, and server. Obviously the scope you would choose would depend on where you needed to store the cached information.
- 3. clear:** This is an optional argument. It tells the tag to remove the cached information.

Using the tag is quite simple. Let's begin with Listing 1 (see page 28), which shows a simple page. In the middle of the page we have some code that generates a number:

```
<!-- Find Foo -->
<cfloop index="x" from=1 to=99999>
    <cfset a = x * 2 * sin(x/2)>
</cfloop>
<cfoutput>
<p>
Final result: a is #a#
</p>
</cfoutput>
```

This code doesn't relate to anything real – it just takes a long time to run. We also generate the total time the page takes to run by checking getTickCount() at the beginning and end of the dynamic content for the page. Run this template. It should take about 2–3 seconds to run. Obviously this is way too slow. One quick fix would be to add <cfcache> to the top of the template. However, notice we also output the current time. If we use <cfcache>, then the time value will get old. We can correct this by simply adding the scopeCache tag. Listing 2 (see page 28) is virtually the same code, except we have wrapped the slow process with a call to scopeCache.

```
<!-- Find Foo -->
<cf_scopeCache scope="application" name="foo">
<cfloop index="x" from=1 to=99999>
    <cfset a = x * 2 * sin(x/2)>
</cfloop>
<cfoutput>
<p>
Final result: a is #a#
</p>
</cfoutput>
</cf_scopeCache>
```

The custom tag code is pretty simple (and can be seen in Listing 3 [see page 28]). Let's take a look at how this tag works. Lines 11–16 simply handle attribute validation for the tag. We first check to make sure the name attribute was passed, and that it was passed in as a simple value (in other words, a string). The name attribute identifies the content we are caching and should be unique per cache. Next we check for the scope attribute. This time we not only check for a simple value, we use the listFindNoCase function to make sure one of the proper values for the attribute was passed in. The scope attribute should only be application, session, or server, which corresponds to ColdFusion scopes.

Query caching works great when you only want to cache a query. <cfcache> is simple to use and easy to implement – but it caches the entire page and caches based on the URL (and URL parameters). You can't tell <cfcache> to cache only one part of a page or store its cache in a session, for example.

The scopeCache tag not only allows for caching of a part of a page, it allows for storing the cached information in various scopes. Imagine a portal like my.yahoo.com. The home page of the portal contains news, weather, and other types of information. However, the news displayed is dependent on who is logged on.

Another problem is that stock prices are part of the information and can never be cached. That alone means we can't use the <cfcache> tag. Query caching would help, but it's not as easy to remove a query from the cache. What we want to do is cache the displayed content of a portion of the page. This tag

After validating the attributes passed in, we begin by creating a reference to the scope we'll be using. The structGet function allows you to pass in a string that corresponds to the name of a structure. The value of attributes.scope, application, session, or server, is a structure in ColdFusion MX. When we assign the result of structGet to the variable, ptr, we are simply saying the ptr equals the evaluation of the value of attributes.scope.

In other words, if attributes.scope was session, then ptr would equal session. Any change to ptr will be reflected in the session scope. What's the point of all this? By setting ptr equal to the scope, I don't have to litter my code with a large number of cfif blocks or evaluation commands. The tag stores all cache information in a substructure called scopeCache, so lines 21-23 simply make sure it exists. You will also notice that we don't use any cflocks. This tag was designed for use on a ColdFusion MX system and therefore the locks are not necessary. However, you could easily rewrite the code so that it works under ColdFusion 5 (or even 4.X).


Lines 25-28 check for the third possible argument, "clear." If passed, it removes the cached value and exits the tag. This is how we will remove content from the cache. Lines 30-38 do one of two things depending on the execution mode of the tag. Tags that wrap, like cfoutput, cfquery, and our custom tag, scopeCache, are used in pairs. For example:

```
<cf_scopeCache>stuff</cf_scopeCache>
```

Inside the custom tag, we can determine if it's being called at the beginning, or the end, by checking the value of thisTag.executionMode. Our logic is simple. If we are calling the tag in the "start" mode, we see if the content has already been cached. We do this by simply using the structKeyExists function. If it does exist, we output the value and exit the tag. This means any code between the beginning and end tags will not be run – and therefore will not slow down the page. (For more on custom tags, see *Mastering ColdFusion MX*, published by Sybex.)

The second portion of the tag cfif block, line 37, simply takes the content

generated between, then tags and stores it in the cache. Again, ptr refers to either the application, session, or server cache.

So, that's it. With a few lines of code and the use of ColdFusion's native persistence, we have a tag that can easily help speed up the performance of your Web site. (Code listings on next page) 

About the Author

Raymond Camden is co-technical editor of ColdFusion Developer's Journal, and a software engineer for Macromedia, Inc. A longtime ColdFusion user, Raymond is a former member of Team Macromedia and a co-author of the "Mastering ColdFusion" series published by Sybex. He also presents at numerous conferences and contributes to online webzines and CFDJ. He and Rob Brooks-Bilson created and run the Common Function Library Project (www.cflib.org), an open source repository of ColdFusion UDFs. Raymond formed and helps manage the Hampton Roads ColdFusion User Group (www.hrcfug.org).

jedimaster@mindseye.com

PAPERTHIN

www.paperthin.com

Listing 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Some Page</title>
</head>

<body>

<cfset initial = getTickCount()>

<p>
<cfoutput>
Some layout. The current time is #timeFormat(now())#.
</cfoutput>
</p>

<!-- Find Foo --->
<cfloop index="x" from=1 to=90000>
  <cfset a = x * 2 * sin(x/2)>
</cfloop>
<cfoutput>
<p>
Final result: a is #a#
</p>
</cfoutput>

<p>
<cfoutput>
This page took #getTickCount() - initial# ms to generate.
</cfoutput>
</p>

</body>
</html>
```

Listing 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Some Page</title>
</head>

<body>

<cfset initial = getTickCount()>

<p>
<cfoutput>
Some layout. The current time is #timeFormat(now())#.
</cfoutput>
</p>

<!-- Find Foo --->
<cf_scopeCache scope="application" name="foo">
<cfloop index="x" from=1 to=90000>
  <cfset a = x * 2 * sin(x/2)>
</cfloop>
<cfoutput>
<p>
```

Final result: a is #a#

```
</p>
</cfoutput>
</cf_scopeCache>

<p>
<cfoutput>
This page took #getTickCount() - initial# ms to generate.
</cfoutput>
</p>

</body>
</html>
```

Listing 3

```
<!--
  Name      : scopeCache
  Author    : Raymond Camden
  Created   : December 12, 2002
  Last Updated :
  History   :
  Purpose   : Allows you to cache content in various scopes.
-->

<!-- Attribute validation --->
<cfif not isDefined("attributes.name") or not
isSimpleValue(attributes.name)>
  <cfthrow message="scopeCache: The name attribute must be passed as a
string.">
</cfif>

<cfif not isDefined("attributes.scope") or not
isSimpleValue(attributes.scope) or not listFindNoCase("application,ses-
sion,server",attributes.scope)>
  <cfthrow message="scopeCache: The scope attribute must be passed as
one of: application, session, or server.">
</cfif>

<!-- create pointer to scope --->
<cfset ptr = structGet(attributes.scope)>
<!-- init cache root --->
<cfif not structKeyExists(ptr,"scopeCache")>
  <cfset ptr["scopeCache"] = structNew()>
</cfif>

<cfif isDefined("attributes.clear") and
structKeyExists(ptr.scopeCache,attributes.name)>
  <cfset structDelete(ptr.scopeCache,attributes.name)>
  <cfexit>
</cfif>

<cfif thisTag.executionMode is "start">
  <!-- determine if we have the info in cache already --->
  <cfif structKeyExists(ptr.scopeCache,attributes.name)>
    <cfoutput>#ptr.scopeCache[attributes.name]#</cfoutput>
    <cfexit>
  </cfif>
<cfelse>
  <cfset ptr.scopeCache[attributes.name] = thisTag.generatedcontent>
</cfif>
```

Download the Code...
Go to www.coldfusionjournal.com

Saturday Sessions

Alan Williamson
JDJ Editor-in-Chief

We understand the pressures of work and how difficult it can be to get time off. That is why we have designed this workshop to be held in one day and, as a special bonus, on the weekend, so no days off from work. **Your boss will be happy!**

JDJ Workshop with Alan Williamson

M	T	W	T	F	S	S
---	---	---	---	---	---	---

Coming to you...

April:
NEW YORK
WASHINGTON, DC

May:
BOSTON
TORONTO

June:
ATLANTA
RALEIGH

Performance > Efficiency > Reliability

This one-day intensive workshop is designed for developers who wish to increase the efficiency and reliability of their code development.

- 1) The day will begin by looking at the various hints and tips you can utilize at the code level to improve the quality and reduce the number of bugs you have to contend with.
- 2) The next part will look at Apache's Ant and how you can use this freely available tool for your own development, irrespective of your IDE.
- 3) Last, and most important, as the old saying goes: "You can never do enough testing." This session will look at JUnit and show you how to start building test harnesses for your code so you can begin your testing strategy.

>Performance

Java is a powerful language. While it offers a rich array of tools, the fundamentals mustn't be overlooked. Improving your code at the core layer will result in great improvements in efficiency and produce (hopefully) less bugs. We'll look at the do's and don'ts of programming and learn many hints and tips that will accelerate your Java coding.

>Efficiency with Ant

Apache's Ant is a powerful scripting tool that enables developers to define and execute routine software development tasks using the simplicity and extensibility of XML. Ant provides a comprehensive mechanism for managing software development projects, including compilation, deployment, testing, and execution. In addition, it is compatible with any IDE or operating system.

> Reliability with JUnit

A critical measure of the success of software is whether or not it executes properly. Equally important, however, is whether that software does what it was intended to do. JUnit is an open-source testing framework that provides a simple way for developers to define how their software should work. JUnit then provides test runners that process your intentions and verify that your code performs as intended. The result is software that not only works, but works in the correct way.

What you will receive...

- ✓ INTENSIVE ONE-DAY SESSION
- ✓ DETAILED COURSE NOTES AND EXCLUSIVE ONLINE RESOURCES
- ✓ JDJ CD ARCHIVE



To Register

www.sys-con.com/education

Call 201 802-3058

SPONSORED BY

JAVA DEVELOPER'S JOURNAL

PRODUCED BY
SYS-CON EVENTS

PRESENTED BY
i-TECHNOLOGY EDUCATION



10 Mistakes Fuseboxers Make

'IF YOU DON'T HAVE TIME TO DO IT RIGHT, HOW WILL YOU FIND TIME TO DO IT OVER?'

In the last year, I've seen a great number of developers make the commitment to learn Fusebox – and for good reason: the Fusebox framework and the Fusebox Lifecycle Process (FLiP) give developers both a framework and a methodology that work well in creating Web applications. Even so, there are still many pitfalls that can trip up Fusebox programmers. If you're learning Fusebox, here are some tips on avoiding common mistakes:

1. Asking too much of wireframes

The Fusebox Lifecycle Process (FLiP) begins with wireframes. A wireframe is a skeletal application meant to engage the client in the process of discovering what is required. I love wireframes. I find them tremendously helpful in the opening stages of a project, when we're trying to find out what the client wants. They're also very useful as a sales tool; clients see them and get excited by the ability to actually point and click – letting them browse through a skeletal application.

But many developers, eager to get coding, treat wireframes as all that's needed to fully explore what clients want. Wireframes work best when treated like a booster rocket – essential in getting us “off the platform,” but quickly becoming ballast that weighs down the project.

Wireframes work best when they're used to help jump-start the process of prototyping – the process of creating an HTML version of the finished application. Too often, I see developers treat the wireframe as a substitute for a prototype. This is almost always a mistake – one that, unfortunately, doesn't reveal itself until the very end of the project. Don't forget a basic truth of software development: clients can only tell us what they want after they see it. In short, wireframes aren't a substitute for prototypes.

2. Omitting the prototype

This is perhaps the worst mistake we can make. It comes in two varieties. In the first, we completely skip the prototype altogether. “Don't have time,” we tell ourselves.

Years ago, I worked as a cabinetmaker. One of the wonderful things about that experience is that it taught me patience and respect for my material and my tools. If the temptation in developing is to start coding too quickly, the temptation in cabinetmaking is to assemble too quickly. Hey, it's fun! I want to see my beautiful cabinet, chair, or curio. I want others to see it, too.

In working with wood, assembling the piece early in the process means that you're working with a lar-



By Hal Helms

gish object rather than working with what are essentially flat pieces of wood. When you do that, things like routing, drilling, and planing become much harder and you risk ruining not only a single piece of wood (a sad enough experience), but the entire piece. After a few ruined pieces, you learn to be patient, to delay gratification, to trust in the process. There's a reason they call things like cabinetmaking – and coding – disciplines.

Before I learned this, I often justified my zeal by telling myself, "I don't have time for all this process stuff. I need to see what I'm building." After the heartrending pain of ruining a piece or the heart-stopping fear of almost ruining a piece convinced me to slow down, I put a sign up in my shop to help me remember. It read, "If you don't have time to do it right, how will you find time to do it over?"

If you find yourself saying, "I don't have time to do a prototype," consider this: the work of the prototype must be done prior to delivering the application. What people usually mean when they say they don't have time to do a prototype is that they want to do it while they're writing the code. This is analogous to designing a cabinet while you're building it.

In the second variety of Mistake No. 2, we do a prototype, but don't do it completely. We whoosh through the process, putting up "sample" pages; we start coding before we know exactly what the content will be – in short, we don't have a true prototype.

But all of those details must be decided at some point in the process. My advice on this is the same I gave myself when I wanted to rush to assemble a cabinet: do it right. Understand that much of the value in prototyping comes from creating a context in which the inherent messiness of finding out what the client wants can work itself out.

If you do the prototype right, you'll find out about key people who weren't in the initial meetings (despite your insistence and the client's assurance that all key decision makers were present). You'll allow the office politics to work themselves out on your prototype – not your code. Finally, you'll arrive at a prototype that both you and your client can agree is exactly what the finished product should look like. Then you can start coding.

How long should a prototype take? Of course, this depends on the application, but I find that almost universally, developers tend to cut the prototype short; the desire to start coding is great. In looking back at jobs I've done, I usually find that the wireframe/prototype phases take up as much as 60% of the entire job. I'm tempted to say that the longer a prototype takes, the greater the chances for project success. Certainly, I've never found myself in a "post mortem" saying, "Gee, I wish we hadn't spent so much time on the prototype."

3. Not establishing acceptance tests up front

It's easy to start coding and – like assembling my woodworking project – it's fun. But when do you stop coding? When is "done" done? If you don't establish this up front, you're likely to find yourself forever chasing a moving goal line.

Did you build the application for a single machine, later to learn that the numbers of users would require a cluster? Did the client assure you they were going to use SQL Server, only to later (and without telling you) decide to adopt Oracle for all corporate projects? Did the client forget to tell you that the application must handle 5,000 transactions per second?

I've made some of these mistakes, so I can tell you that it is definitely not fun to try to convince your clients (after the fact)

that no, this had not been discussed beforehand and that no, the application was not built for this. Make expectations clear: put them into the project documents right after you freeze the prototype. Determine the environment in which you run your acceptance tests and make sure that you have access to that environment while coding.

4. Not using unit-testing code

Now I realize your code is flawless – a model by which to guide less-experienced coders. Mine too. But we do have to work in teams with other coders and those coders – well, they do make mistakes, don't they? If you don't unit-test code, you're going to find the flaws when you begin to assemble the project. At that point, finding out where the flaws occur is the old needle-in-the-haystack problem. Again, not fun.

Unit-testing code doesn't need to be difficult. I recommend building a simple test harness that creates an environment in which an individual fuse can run. This might include setting variables needed by the fuse, establishing state management, etc.

Your Fusedoc provides you with all the information you need to build a test harness, and I recommend building them and using them to test every fuse you write. These test harnesses can be saved and run at any point. This can be a great confidence booster when you're deeply into a critical project and could use a little encouragement. Oh, and since we don't want to make those other coders – the ones who make all the mistakes – feel bad, we might as well test our own code, too.

5. Waiting too late to integrate unit-tested code

This mistake is easy to make. There's so much to do while involved in a project – especially a large project – that it's easy to defer integrating the unit-tested code until very late in the project.

I recommend that, if you aren't already doing this, you try having regularly scheduled project "builds," where you integrate all of the code. Typically, after I've identified the fuses and written Fusedocs, I create a simple stub that says something like, "I'm home.main and I'm not implemented yet." With this, I can do a project build from the first day. Now, granted, it doesn't do much, but it does give me a framework for integrating working, tested fuses. I can build the project at regular intervals.

Daily builds are great confidence boosters. They also serve to alert us to areas where problems may arise. It's become something of a joke that on any project, the first 90% of the project is done in three months, while the remaining 10% takes another three months. Or six.

The problem, of course, is that we were never 90% done. We developers are nothing if not optimists. Daily builds are also helpful when working in team environments. Rather than trying to get a teammate to commit to a deadline, why not just use daily builds to see exactly how far along the project is.

The rule I use is that all completed fuses are checked in by midafternoon. Another rule is that only one fuse is worked on at a time. This means that at any given point, I should be able to see how the project is doing. If someone needs help, it becomes clear at an early stage and saves recrimination and blame later on.

The best way to ensure project success is to know exactly where we are. Then we can make needed changes and the project can get back on track. Depending on who your client is, you may want to give them access to a daily build site. This can alleviate a good deal of client nervousness. After all, they have rea-

son to fear if their project fails (as most do) and a partnership approach can make you both successful.

6. Starting at the database

If omitting the prototype is the most serious mistake we can make, starting a project at the database is probably the most common. “Let’s get started on the database schema,” someone will say, and others readily go along. It seems to make such good sense. Besides, it’s something we can do that feels concrete; we can show our managers the progress we’re making.

I suggest that a better approach is to begin by identifying the conceptual components of your application. If, for example, you’re dealing with a document management system, your conceptual components probably include a “Document” and an “Author.” These exist quite independently of any database. In fact, a database exists only to provide a physical storehouse for these conceptual components. Beginning with a database is truly putting the cart before the horse.

Once you’ve identified your conceptual components, determine the various properties for each component. If, for example, my application has the concept of an “employee,” I might identify its properties as `employeeID`, `firstName`, `lastName`, `dateOfHire` and so forth.

Part of writing good code is writing clear code. If you’re taking your cue from the database – especially if your company has database naming standards – then you’re likely to end up with variable names in your code like `PRO_CDR_RTN`. What does that stand for? Who knows? And once you find out, who will remember in six months?

Instead, you decide what your variables should be named. Make them clear so that in six months, you, or whoever is working on your code, will be able to tell the nature of the variable from its name. Then you can alias the actual database names so that they come from the database with the name that you want.

It’s not just a matter of naming schemes. Beginning with a database creates a needless dependency. Coding can’t begin until all the database issues are resolved. But in a complex application, this can take weeks or months. Do you want to wait this long to write your code? I sure don’t.

You’ll want to write code to work with the eventual database record set, of course. One of the great things about relational databases is that they always return tables (or relations in relational database lingo). What starts off as a seven-table many-to-many join gets resolved down to a simple record set. That means that you can use an artificially created record set and work with it while you’re writing code. Later, you can swap out the artificial for the real thing. Your code won’t break and it won’t have to be rewritten.

I wrote a custom tag called `QuerySim.cfm` to help with this. You can download it at www.halhelms.com.

7. Not using a standard

But wait – if we’re using Fusebox, we’re using a standard. How can this mistake apply to us? Surprisingly often; otherwise smart people recommend to others the practice of “adapting” a standard such as Fusebox. While the desire to put one’s own stamp on something is understandable, it’s often a mistake.

One of the great things about standards is that they are, well, standards. Having an “almost standard” won’t help much when you run into problems and need help either debugging or finishing code for a project. And unless you want to be stuck forever maintaining code you’ve written, you’ll want to write to a standard so that others can pick up what you’ve done quickly and easily.

Standards are especially important if you’re working on a team. One of Eli Whitney’s great contributions to the world was his demonstration that standardized parts could allow products to be built cheaper, faster, and better than the old practice of each person doing their own thing. Whitney was a genius; his advice is as true of coding as it is of manufacturing.

8. Not using projects to learn new technology

You’ve noticed that technology is whizzing past you? Why not use your projects to help both you and your company by increasing your knowledge? Now, far be it for me to discourage anyone from going to training classes (since I offer them!), but I do encourage you to also use the project you’re working on to do some experimenting. I’m not suggesting that you turn your project into a science project, but rather select a very specific and relatively small technology (or a small portion of a large technology) to play with. Besides, have you noticed that companies tend to send their best people off to training – people who have taken initiative and actively seek out learning experiences?

For example, you might decide to experiment with XML on a project. ColdFusion has some wonderful tags and functions that make this process fast and painless. Or you might decide to try implementing the Model-View-Controller design pattern on your application (or a small part of it). (I’ve written about the MVC design pattern on my site, www.halhelms.com.) If you’re using a good methodology like FliP, you’ll have time to do this without endangering the project at all.

A friend remarked to me recently that “the difference between a programmer and a plumber is the programmer’s knowledge.” If you don’t invest in yourself – in your knowledge – you’re likely to find yourself without current skills, putting both you and your company in jeopardy. So be strategic: select a specific area that won’t consume an inordinate amount of time and go to school on it!

9. Fixing problems

We’ve all done this one: pressure on, little time left, deadline looming, and we have a bug. What do you do? Why, reach for the can of Lucky Brand Magic Fixit Powder, of course. This may take the shape of finding someone who you think may know more than you, shipping off all of the code for the entire project with an accompanying e-mail that goes something like this: “My code isn’t working. Please help! I’ve got a presentation in two hours!!! TIA”

Or you might find yourself commenting out variables that are giving you problems, or adding in variables just to make the code “work.” Believe me, I am sympathetic. I know the frustration and panic that such a situation causes. But the solution is not reaching for the Magic Fixit Powder. Once simple syntax errors are resolved, you’re not dealing with a bug. Instead, you’re getting valuable feedback telling you that your understanding of the logic of your program is incomplete. That’s all.

The solution, therefore, is to gain a complete understanding.

How? Well, one thing I recommend without fail is to use Fusedocs to document your code. You can easily see what variables are supposed to be entering and exiting your code. Are they doing what they're supposed to? Use test harnesses to ensure that what you think is happening really is. You can set self to the value of a page that simply calls Dan Switzer's wonderful Debug.cfm custom tag (www.pengoworks.com [<http://www.pengoworks.com>](http://www.pengoworks.com)), which will give you lots of information about the state of your application.

The important thing to realize is that you're in a situation that is not a singular occurrence. For as long as you're programming, you're going to be dealing with code that doesn't do quite what you expected it to, so craft a long-term strategy. Isolate the problem, use snippets of code running in a test directory to make sure that each individual piece of your code works. The downside of this advice is that you are likely to end up getting those e-mails thanking you in advance for "fixing" someone else's problem!

10. Not asking for help

Wait a minute. I just told you not to rely on others to save you. Now I'm saying that one of the deadly mistakes that Fuseboxers make is not asking for help? Well yes, that is what I'm saying. Do your homework, for sure. Isolate the problem so that you know exactly where the problem is occurring, but if you still can't figure out where the problem is, ask for help.

Innumerable times, I've found that printing out the code in order to go over it with someone else reveals the problem quick-

ly. I know someone who has a stuffed animal on their monitor that they talk with when running into problems. The point is to break a mental logjam by changing the environment.

Programming, sadly, is often a culture of one-upmanship, where lack of knowledge is a sign of weakness and revealing any weakness is seen as shameful. That's bunk. Our job is to produce a successful software deployment, not to prove to someone that we're like Al Franken's character, Stuart Smally, who needed constant self-assurance that he was "smart enough, good enough, and – doggone it – people like me."

Like the ColdFusion community, where Fusebox had its start, one of the great things about the Fusebox community is its collegiality and closeness. The forums on the fusebox.org site offer assistance from people who are more than willing to help. Lists like *cf-talk* at Michael and Judith Dinowitz's houseoffusion.com are invaluable. I write an occasional newsletter that deals with software engineering issues. (Sign up at halhelms.com.)

Well, that ends our tour of "10 Deadly Mistakes." If you've got any others you'd like to share, send me an e-mail at hal@techspedition.com. 

About the Author

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

hal@fusebox.org

CFXHOSTING

www.cfxhosting.com

Exploring Amazon Web Services with ColdFusion MX

They're there to serve us all

Amazon.com, the "earth's biggest selection," has exposed their entire catalog of products, product reviews and ratings, user-created book lists, and more via Web services. Whether you're new to Web services or not, and even if you don't care so much about accessing Amazon's data, it presents an interesting case study of doing Web services in CFMX. You might even be able to turn the opportunity into profit!

In this month's **Journeyman** column, I'll walk you through using Amazon Web Services (AWS), and along the way will point out some of the key points you need to know about CF (and Dreamweaver) to be able to use them. If you've not explored this feature of CFMX yet, the good news is that it's really quite easy. And if you have begun using Web services in CFMX, you might learn a thing or two along the way.

ColdFusion MX supports both the consumption and creation of Web services. In this article, we'll focus only on consuming them. There have been several articles both in this magazine and at the Macromedia DevNet site that cover more details about creating Web services (using CFCs). You may want to look for and refer to those for more background after reading this if you're entirely new to the process of working with Web services in general, or in CF.

But any CF developer should be able to follow along here quite easily. CF makes this just extraordinarily trivial to set up compared to many other environments.

A Quick Example

I want to start off quickly with an example of something that you can run right away without having to write any code. Save the code in Listing 1 to your CFMX server environment, placing it in a directory where you can execute other



By Charlie Arehart

CFMX templates. This might be a subdirectory of your webroot, such as c:\cfusionmx\www-root. Save it as amazon-search-simple.cfm.

Don't worry if you don't understand entirely what's going on. Just save it and run it, then take a look at the code.

Sidebar

Before you try to browse this URL to execute the request, note that Amazon expects developers to obtain their own "developer tag." This is free and can be done easily at www.amazon.com/websevices. You'll get one on the spot. Apply that value in the one line of code:

```
aKeywordRequest.devtag="yourtag";
```

This simple example will retrieve a selection of books from Amazon's database, based on a "keyword" provided in the query string. In other words, to search for a book, enter ?keyword=somevalue on the URL when calling this template. I've designed it so that if you don't supply any, it will search for CF books by default. (Certainly, a more complete example would provide a form for the user to enter values instead.) Figure 1 shows a screenshot of how the results might appear.

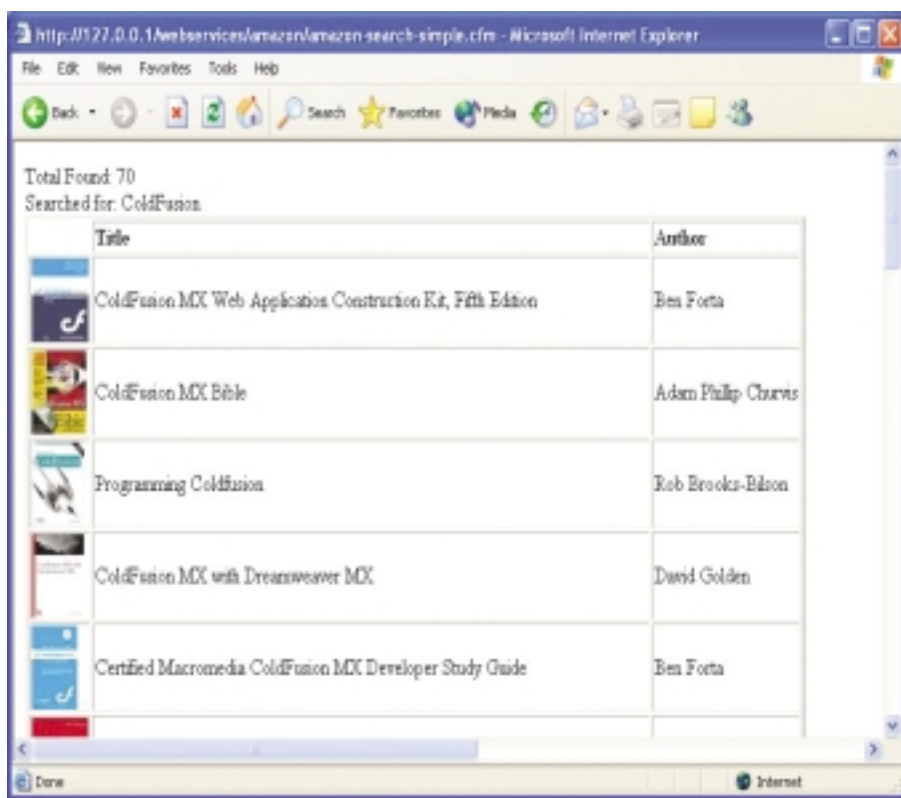


Figure 1

This is a very simplistic example that leaves a few questions unanswered: Why were only 10 records returned? How do you request additional records? What's the sort order and how can that be changed? What if there are multiple authors? What other data is returned about the books? What other sort of searches can I do? We'll address those later.

For now, let's take a look at some fundamentals of setting up the code to call the Web service and process its results.

The Printed Documentation About the Web Service

Notice that the CFINVOKE is calling the Web service at the URL: <http://soap.amazon.com/schemas2/AmazonWebServices.wsdl>. Where did I get that value? It's provided in the documentation about the Web services, which is available via download as part of the "Developer's Kit" available at www.amazon.com/webservices.

That document, the "API and Integration Guide," offers a pretty complete discussion about many aspects of working with AWS, both in the SOAP Web service approach (that we're using here) as well as a pure XML approach for those who might prefer that. For a discussion of processing Web services in XML in an older form of AWS, see the sidebar about Jeffrey Houser's article in the Macromedia DevNet.

Sidebar

Jeffrey Houser has done another article on integrating Amazon Web services with CF, at www.macromedia.com/devnet/mx/coldfusion/articles/wsamazon.html. That article is based on an older form of obtaining Amazon Web services as pure XML. Though the service he calls and the code he wrote still work, the Amazon Developer's Kit describes a newer approach to requesting pure XML responses from AWS. There's also an option to pass in an XSLT file to modify the output from that XML-based mechanism. In either case, the approach I describe in this article saves you having to perform any XML parsing or processing. That's one benefit of SOAP-based Web services processing!

The Guide also lists all the available types of Web service methods that can be called, each of which enables different kinds of searches, such as by author name, artist/musician name, actor name, directory name, ASIN (Amazon's unique product IDs, which for a book is the same as its ISBN), UPC code, and more. Some of the listed methods include:

- KeywordSearchRequest
- AuthorSearchRequest
- ArtistSearchRequest
- ActorSearchRequest

You see the pattern to their names.

The Data Passed in to the Web Service Method

The Guide also explains that each method further requires that a specific argument be passed in on the request to execute that method, and it documents the argument name expected for each method, as well as the properties that the argument should include.

For instance, in the case of the KeywordSearchRequest used in Listing 1, the Guide shows that the expected name of the argument to be passed in is also KeywordSearchRequest, and it's to be composed of the following properties (with their data types):

- keyword (string)
- page (string)

- mode (string)
- tag (string)
- type (string)
- sort (string, optional)
- devtag (string)

The Guide further explains the meaning of each of those properties. I'll discuss some of them later.

For now, let's take a look again at Listing 1. In ColdFusion, the way to pass those properties as an argument called "KeywordSearchRequest" is to package them as a structure, which I do in the first few lines of code. Then in this example that structure is passed on the CFINVOKEARGUMENT tag:

```
<cfinvokeargument name="KeywordSearchRequest"
    value="#aKeywordRequest#"/>
```

While you could call the actual structure any name you want (I called it "aKeywordSearchRequest"), you must pass KeywordSearchRequest as the value in that tag's NAME attribute since that's what the Web service is expecting. (Note that you could also avoid the CFINVOKEARGUMENT tag by placing the attribute=value pair, KeywordSearchRequest="#aKeywordSearchRequest#", on the CFINVOKE tag itself. The two approaches achieve the same result.)

The Data Returned from the Web Services Method

Finally, the Guide also shows what results are returned, including the names of fields, arrays, and other objects that

COLDFUSION- HOSTING.COM

www.coldfusionhosting.com

might be returned. In the case of a `KeywordSearchRequest`, the method returns a `productInfo` structure (referred to as a “node” in the Guide), whose keys describe the set of search results. The Guide explains that the `productInfo` node/structure contains the following data:

- **TotalResults:** Number of results (products) generated by search
- **ErrorMsg:** The error message that is returned if the call fails
- **Details:** An array describing the details about each specific product

You'll notice that in Listing 1, I provide a name to refer to the ProductInfo results by way of the RETURNVARIABLE attribute of CFINVOKE, and I named it "aProductInfo." I could have called it anything I like. This is available as a structure in ColdFusion. I then use that to create output about the results in general, as in the first line inside the CFOUTPUT, which shows how many records were found (aProductInfo.TotalResults).

As stated above, the information about each record is found in a Details array within that ProductInfo result variable (in our case, the aProductInfo structure), and each element in the array will be a structure describing each product. The guide further describes the information returned for each product:

- **ASIN:** Amazon.com Standard Item Number
- **ProductName:** Name or title of product
- **Catalog:** Product category (book, music, etc.)
- **Artists:** Musician(s) of a CD or cassette
- **Authors:** Author(s) of a book
- **URL:** Link to product detail page at Amazon.com
- **ReleaseDate:** Date that the product was released to market
- **Manufacturer:** Manufacturer of product; can represent publishers (for books), labels (for music), or studios (for films)
- **ImageUrlSmall:** Small product image or cover art
- **ImageUrlMedium:** Medium product image or cover art
- **ImageUrlLarge:** Large product image or cover art

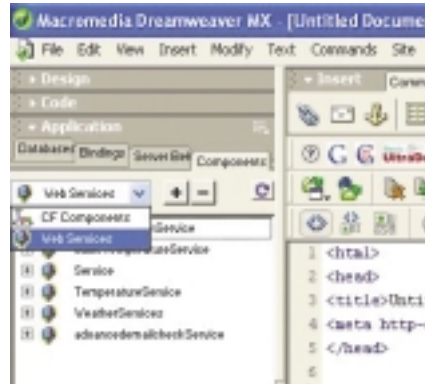


Figure 2

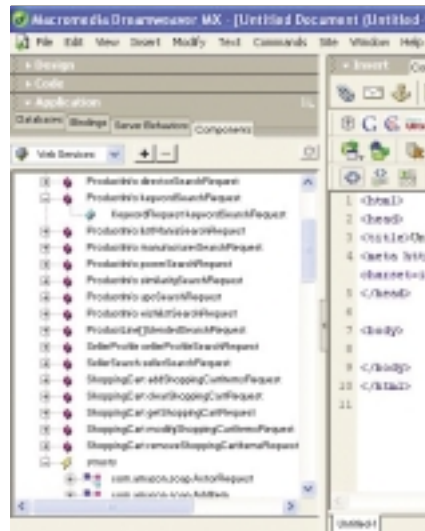


Figure 3

- **ListPrice:** Usually represents the price of the product suggested by the manufacturer
- **OurPrice:** The Amazon.com product price
- **Status:** Indicates if we encountered any errors while processing the service request

That's quite a bit of detail. Do you see now that this whole process is like performing a remote query against the Amazon database? We really get a lot of data back, including links to pictures, actual current prices, and more.

And this set of data is just what Amazon calls the “lite” set of data. The Guide also describes a “heavy” set of data that can provide much more information, such as sales rank, ratings, availability, features, tracks on CDs, reviews, similar products, and more!

You'll notice that in the simple example I have a CFLOOP looping over each element in the aProductInfo.details array, and for each I'm outputting just the product name and authors. Note, too, that the authors' value is itself another array, holding the name of each author, if any.

You might think that it would be useful to simply perform a CFDDUMP on this chunk of data returned from the Web service. Sadly, you won't like the result. Because CF sees this `aProductInfo` variable as an object, rather than a structure with an array of other structures, if you try to dump it, you just get a Java-oriented description of the object.

This may bother you, though, because it means it can be difficult to know what exactly is returned by the Web service call. There's good news, however. Dreamweaver offers a mechanism to at least be able to see what kind of data will be returned, to help you build the code to output or otherwise process that returned data.

Using Dreamweaver MX to Explore the Web Service Definition

It can be cumbersome referring to the printed documentation to find out such information about the Web service as its available methods and their expected arguments, and the names and types of the returned results from each. Fortunately, Web services are self-documenting.

If you execute the URL of the Web service itself (the value entered in the CFINVOKE WEBSERVICE attribute) via a browser, that will generate an XML stream in return that does document how the Web service can be executed and the type of data that it will return. See Listing 2 for a small portion of the 1,000+ lines of XML returned that describe the AWS. You have to be pretty savvy in both XML and SOAP (and patient) to be able to read through that document to discover needed information.

Fortunately, Dreamweaver MX (DWMX) has a very useful feature for exploring Web service methods, arguments, and return results. From the Application panel, select the Components tab. Choose “Web

Services” from the drop-down next to the “+” sign. See Figure 2:

Finally, select that “+” sign to the right of the drop-down menu where you just selected “Web services.” At the prompt, paste in the URL of whatever Web service you care to browse. In our case, it’s the URL in the CFINVOKE tag. In this same screen, be sure that “ColdFusion MX” is selected for the “Proxy Generator” drop-down.

When you select “ok,” DWMX will go out to the Web service and examine that same XML stream we just discussed. But more important, when it’s finished a few moments later, it will now have registered the Web service within DWMX for whatever “site” you had open at the time. Figure 2 happens to show a few Web services already having been registered on my site.

Further, if you expand any one of these Web services in DWMX, you will be shown all the Web service methods, as well as the arguments expected within each. In Figure 3, you can see I’ve selected the keyWordsearchRequest method, and its expected

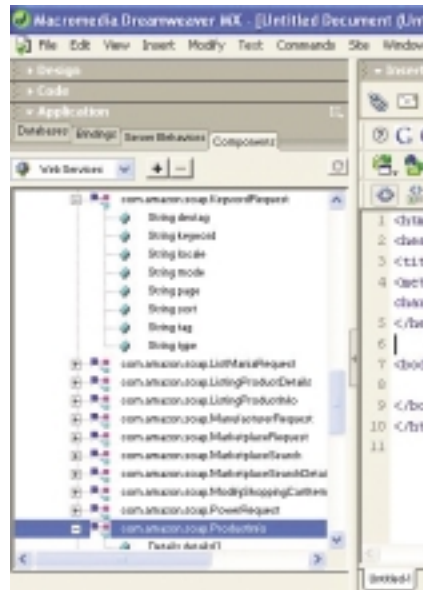


Figure 4

KeywordSearchRequest argument shown underneath it. A couple of things worth noting are that the methodname and the argumentname are preceded by other words.

In the first case, the ProductInfo refers to the data type of what’s returned from the method. In the second case, the KeywordRequest refers to the data type of what’s to be passed in the argument. But what are those and where do you find out more about each of them?

At the bottom of this list of methods for the Web service, as seen in Figure 3, there is another element listed as “structs.” Expanding that, you’ll see several entries, with one for each kind of data type (or node or structure) that is mentioned in that display of data types for methods and request arguments. Figure 4 shows the KeywordRequest and (the beginning of) the ProductInfo structs, expanded to show their properties.

The list in the KeywordRequest (technically, is com.amazon.soap.KeywordRequest, at the top of Figure 4) maps to the same KeywordSearchRequest argument discussed at the opening of “The Data Passed in to the Web Service Method,” above. And the ProductInfo item at the bottom of Figure 4 is the very

EKTRON

www.ektron.com/cfdj

ProductInfo described in “The Data Returned from the Web Service Method,” above.

It's worth noting that the name of these structs is the same as the data type name indicated before the names of methods and arguments in the list above “structs.” These struct names do not necessarily equate to the names of the arguments to be passed to the methods. In our example, Figure 3 showed that the method KeywordSearchRequest expected to be passed an argument of the same name. But that argument was indicated to be of a data type called simply KeywordRequest. That's what is listed in the “structs” section. And we need to then create a structure whose key-names map to those properties listed for KeywordRequest in Figure 4. It can be a little confusing. Hopefully this example will help get you straightened out as you explore this further.

Building Code with Dreamweaver

Finally, it's important to note that you can use all this information for more than just documentation. Click and drag a method name onto the code editing area in DWMX. If you're in “code view” of DWMX (View>Code), you'll see that DWMX will build a skeletal CFINVOKE and nested CFARGUMENT for you, with all the method names and appropriate argument names all properly filled in. Nifty!

Indeed, the CFINVOKE code in Listing 1 was generated with this very feature. All I needed to change was the VALUE attribute in the CFINVOKEARGUMENT, to specify the name of the structure I had built in the first few lines. And the keys in that structure were obtained by dragging and dropping the properties in the aforementioned KeywordRequest “struct.”

Even dragging the KeywordRequest itself will create a line of code to create a new structure, since that is indeed what you must do to create this input argument structure. Unfortunately, the code generated is a CFSET doing a struct-new(). I changed it in Listing 1 to all take place in a CFSCRIPT operation. Do whatever makes you comfortable. At least you now know that some of the code can be created for you.

When it comes time to build the output processing aspect, go to the ProductInfo “struct.” Again, you can drag and drop the totalResults property name into your CFOUTPUT code. And when you see that there's a details[] array listed, that suggests you need to find the element of the same name in the “structs” area to find out what elements are within that data type.

If that shows that there's possibly an array of tracks[] within that, you'd then find an element describing those. Just note that in the case of an array like authors[] inside of ProductInfo, it shows that the data type for that is simply a string. That means there will be no further breakdown of elements for authors. It indicates that this is just an array of author names.

Before concluding this article, let's take just a brief look at some of the ways you might modify and enhance the example in Listing 1.

Enhancing the Example

We pointed out previously that the simple example we showed in Listing 1 left quite a bit unanswered. Let's address a few of these points quickly. First, did you notice that the Amazon search mechanism returned only 10 records at a time? That's by design, and it currently cannot be changed. What we can do is cause it to return another “page” of 10 records by modifying the page property in the KeyWordSearchRequest argument:

```
aKeywordRequest.page="1";
```

Of course, you'll want to change that programmatically, to facilitate paging or scrolling through the results. A greatly enhanced example in Listing 3 addresses that as well as a few more points.

Did you wonder what sort order the results were shown in? The Web service defaults to sorting the results in “Featured Item” order for the keyword search we were doing. You can change that sort order if you'd like. I've commented out a line of code that would cause an alternative sort by sales rank:

```
aKeywordRequest.sort="+salesrank";
```

Note also that the aKeywordRequest structure is also where the “lite” or “heavy” type of result can be requested in the “type” property. And finally there's a “mode” property that has significance in indicating the kind of product you're searching for. In our case, it's set to “books.” The AWS Guide offers more information on finding out valid values for the “mode” property. Just be careful not to make a mistake in that sort of designation. If you mistakenly left it as “book” rather than “books,” you'd get a rather unclear error: “Could not perform Web service invocation 'KeywordSearchRequest' because AxisFault fault”.

That's a fault in the underlying Axis processing, not from Amazon it seems, but it's an error to be careful about either way. It's worth noting that CF's Web services foundation is built upon the Apache Axis project, and while the base version of CF and the first two updaters continued to use a .9 release of that Axis project, the Updater 3 has resolved that, giving us the full version. It may not impact using AWS, but it's worth noting for other Web services.

Be aware also of some possible surprises about the data as you're processing results from a Web service, such as in the Details array in our example. For instance, we've said before that the authors field was an array. In the simple example, I was processing only the first author. You should be prepared to handle all of them. The enhanced code in Listing 3 does that.

Also note that a book might not have any authors at all. This is an interesting situation, because while you may think you can use CF's isdefined() function to check for that, it will not work. Sadly, you really need to test for a null value in that property, and CF offers no means to do that. You'll notice that I use CFTRY/CFCATCH processing so that if I refer to a field that has no values, instead of getting an error, I just continue processing.

The enhanced version of the example, in Listing 3, does that and more, performing scrolling through records and showing how to handle multiple or zero authors. There are still more things that could be done, such as having a form for inputs and controlling other

aspects like the page, etc. It could also perform caching of the Web service, so as not to visit it too often when data may not change. This isn't a trivial problem, but it is one worth exploring.

One last point to note is that if you want to quickly and easily explore the data that would be returned from the Amazon Web services, you can always use the XML service I alluded to above. Again, see the API for more information, but with it you can enter a URL such as this in your browser:

```
http://xml.amazon.com/onca/xml2?dev-t=yourdev-
tag&t=syndic8-20&KeywordSearch=ColdFusion&page=1&
mode=books&type=lite&f=xml
```

Conclusion

So now you know how to access the Amazon Web service, as well as find documentation both in print and electronically to understand better what is possible with that service and its methods. And you've learned how to take advantage of Dreamweaver MX to make browsing Web service properties, and even building code, easier.

Now you just need to explore all the possibilities that Amazon Web Services provides. We've really only scratched the surface. As I alluded to previously, there is also a mechanism that would allow


you to enable one-click and other shopping cart functionality on your site that can be executed against the Amazon servers to allow your visitors to make purchases. And if you obtain and use that Associates ID, you can earn from 5–15% on their purchases. Wow! Check out the Guide for more information.

If you're more a developer than a budding retailer, consider how you might still turn AWS to profit. First of all, you could build vastly enhanced mechanisms to help retailers show off certain products. Remember that there are reviews, ratings, similar products, and a whole lot more that you can tap into with AWS. Apply a little Flash and Flash Remoting, and you can build something that may truly be unique.

Consider also that you can perform searches by seller, so that you could help a seller create better inventory or other product management and integration systems to sync their own data with that about their transactions at Amazon. The sky really is the limit.

Be sure to read the API and Integration Guide. It's not too long and offers lots of useful information. You'll also want to download that Developer's Kit as it has lots of examples (though none currently in CF, which I hope to address with them.) There is also a Discussion Board at that

www.amazon.com/webservices site that's friendly, supportive (including AWS staffers), and not generating more than several posts a day. Sometimes there are very useful discussions that take place.

Finally, I'll remind you to seek out some other articles (and books) on the topic of Web services in CFMX, and Amazon Web Services in particular. Don't be afraid to do a Google search and see where it takes you. Even a resource shows using them in ASP or PHP may still share new perspectives that could apply to your using Web services in CFMX. That's the cool thing about Web services: they don't care what kind of client you are. They're there to serve us all. Enjoy. 

About the Author

Charlie Arehart is co-technical editor of ColdFusion Developer's Journal. He's also a certified Macromedia trainer/developer, Team Macromedia member, and CTO of SysteManage. He contributes to several CF resources, is a frequent speaker at user groups throughout the country, and provides training, coaching, and consultation services. He is also now a partner in CommunityMX.com.

carehart@systemanage.com

Listing 1: Amazon-search-simple.cfm

```
<!---
Name: amazon-search-simple.cfm
Author: Charlie Arehart, in ColdFusion Dev Journal, Apr 2003
Desc: Uses a URL variable called "keyword" to do a search of the Amazon
site using Amazon Web Services. If none is provided, presumes a search of
"ColdFusion" by default.
Notes: This is a very simple example. A more complete example is offered
as amazon-search-enhanced.cfm.
--->
<cfparam name="url.keyword" default="ColdFusion">
<cfscript>
    aKeywordRequest = structnew();
    aKeywordRequest.devtag="yourtag";
    aKeywordRequest.keyword=url.keyword;
    aKeywordRequest.mode="books";
    aKeywordRequest.page="1";
    // aKeywordRequest.sort="+salesrank";
    aKeywordRequest.tag="webservices-20";
    aKeywordRequest.type="lite";
</cfscript>
<cfinvoke
    webservice="http://soap.amazon.com/schemas2/AmazonWebServices.wsdl"
    method="KeywordSearchRequest"
```

```
returnvariable="aProductInfo">
    <cfinvokeargument name="KeywordSearchRequest"
        value="#aKeywordRequest#" />
</cfinvoke>

<cfoutput>
Total Found: #aProductInfo.TotalResults#<br>
Searched for: #url.keyword#
</cfoutput>
<table border="2">
    <tr><td></td><td><strong>Title</strong></td><td><strong>Author</strong></td></tr>
    <cfoutput>
    <cfloop index="i" from="1" to="#arraylen(aProductInfo.details)#"
        step="1" >
        <tr>
            <td></td>
            <td>#aProductInfo.details[i].ProductName#</td>
            <td>#aProductInfo.details[i].authors[1]#</td>
        </tr>
    </cfloop>
    </cfoutput>
</table>
```

Listing 2: Some of the SOAP XML for AWS

```
<wsdl:definitions xmlns:typens="http://soap.amazon.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xml-
soap.org/wsdl/" targetNamespace="http://soap.amazon.com"
name="AmazonSearch">
  <wsdl:types>
    <xsd:schema xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://soap.amazon.com">
      <xsd:complexType name="ProductLineArray">
        <xsd:complexContent>
          <xsd:restriction base="soapenc:Array">
            <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="type
ns:ProductLine[]" />
          </xsd:restriction>
        </xsd:complexContent>
      </xsd:complexType>
      <xsd:complexType name="ProductLine">
        <xsd:all>
          <xsd:element name="Mode" type="xsd:string" minOccurs="0"/>
          <xsd:element name="ProductInfo" type="typens:ProductInfo"
minOccurs="0"/>
        </xsd:all>
      </xsd:complexType>
      <xsd:complexType name="ProductInfo">
        <xsd:all>
          <xsd:element name="TotalResults" type="xsd:string"
minOccurs="0"/>
          <!-- Total number of Search Results -->
          <xsd:element name="TotalPages" type="xsd:string"
minOccurs="0"/>
          <!-- Total number of Pages of Search Results -->
          <xsd:element name="ListName" type="xsd:string" minOccurs="0"/>
          <!-- Listmania list name -->
          <xsd:element name="Details" type="typens:DetailsArray"
minOccurs="0"/>
        </xsd:all>
      </xsd:complexType>
    </xsd:schema>
  </wsdl:types>

  <!-- There is much more. View it by entering
http://soap.amazon.com/schemas2/AmazonWebServices.wsdl into a browser-->
```

Listing 3: An Enhanced Example

```
<!--
Name: amazon-search-enhanced.cfm
Author: Charlie Arehart, in ColdFusion Dev Journal, Apr 2003
Desc: Uses a URL variable called "keyword" to do a search of the Amazon
site using Amazon Web Services. If none is provided, presumes a search of
"ColdFusion" by default.
Notes: This is an enhanced example as compared to the simpler example
offered as amazon-search-simple.cfm.
```

This version performs scrolling through records and shows handling multiple or zero authors. There are still more things that could be done, such as having a form for inputs and controlling other aspects like the page, etc. It could also perform caching of the Web service, so as not to visit it too often when data may not change. This isn't a trivial problem, but it is one worth exploring.

-->

```
<cfparam name="url.keyword" default="ColdFusion">
<cfparam name="url.page" default="1">

<cfscript>
  aKeywordRequest = structnew();
  aKeywordRequest.devtag="yourtag";
  aKeywordRequest.keyword=url.keyword;
  aKeywordRequest.mode="books";
  aKeywordRequest.page="#url.page#";
  // aKeywordRequest.sort="+salesrank";
  aKeywordRequest.tag="webservices-20";
  aKeywordRequest.type="lite";
</cfscript>

<cfinvoke
  webservice="http://soap.amazon.com/schemas2/AmazonWebServices.wsdl"
  method="keywordSearchRequest"
  returnvariable="aProductInfo">
  <cfinvokeargument name="KeywordSearchRequest"
    value="#aKeywordRequest#" />
</cfinvoke>

<cfoutput>
Total Found: #aProductInfo.TotalResults#<br>
Searched for: #url.keyword#
</cfoutput>

<table border="2">
  <tr><td></td><td><strong>Title</strong></td><td><strong>Author</
strong></td></tr>
  <cfoutput>
    <cfloop index="i" from="1" to="#arraylen(aProductInfo.details)#"
      step="1" >
      <tr>
        <td></td>
        <td>#aProductInfo.details[i].ProductName#</td>
        <td><cftry><cfloop from="1"
          to="#arraylen(aProductInfo.details[i].authors)#"
          index="z">#aProductInfo.details[i].authors[z]#<br></
          cfloop><cfcatch>&nbsp;</cfcatch></cftry></td>
        </tr>
      </cfloop>
    </cfoutput>
  </table>

  <cfset numpages = aProductInfo.TotalResults/10>
  <cfif url.page gt 1>
    <cfoutput>
      &lt;<a href="#cgi.script_name?page=#url.page-1#">prev</a>&gt;
    </cfoutput>
  </cfif>

  <cfif numpages gt url.page>
    <cfoutput>
      &lt;<a href="#cgi.script_name?page=#url.page+1#">next</a>&gt;
    </cfoutput>
  </cfif>
```

Download the Code...
Go to www.coldfusionjournal.com

cf community

—continued from page 8

customer issues; the reality is that we apply a very methodical process to work toward accurate and timely resolutions.

CFDJ: How is your department working with other Macromedia departments to offer the best possible service?

Christian: One thing that has really impressed me about Macromedia is that although we are a relatively large company, we really tend to operate much more like a small entity. One of the ways we maintain a small-company feel is by communicating often and efficiently. I am in constant contact with the product team, customer and technical support, DevNet authors and editors, other community managers, and the community itself to make sure the right information is reaching the right people at the right times.

Sarge: Professional Services works directly with Sales to provide customers with end-to-end solutions. Providing on-site customer support affords my team the opportunity to interface with

clients and to see how our products respond in their environments. We communicate with other product support engineers to learn the history of customer incidents and work with them while on site to resolve the issues. We provide feedback and detailed analysis to Macromedia Engineering, which often results in hot fixes for updaters and feature enhancements for new releases. I am also responsible for Macromedia Flash Remoting support, so I often work with the Tools support. Finally, we provide customer feedback to product marketing. Again, customer advocacy is also a component of the job and we have the opportunity to provide the product teams with intimate details of how customers are using our software and what enhancements they want to see in future releases.


CFDJ: Sarge, what would you like to tell the general public about Macromedia technical support that they may not already know?

Sarge: The team is dedicated and loyal to our customers. We work to ensure

that our clients' enhancement requests and bug fixes are addressed and implemented in a timely way.

CFDJ: Christian, What would you like to tell the general public about Macromedia's developer community support that they may not already know?

Christian: I would like to make sure that everyone in the ColdFusion community realizes what a unique community they have. I spend a lot of time reading posts and following threads, and I am constantly amazed at how creative, responsive, and accepting the ColdFusion community generally is. Developers who haven't spent significant amounts of time in other communities may not realize what a good thing they have going here.

One other thing I would like to make sure everyone understands is how seriously not just the ColdFusion community, but all the Macromedia communities, are taken within the company. Although at the end of the day the decisions Macromedia makes have to be the right business decisions, we take the positions of our communities very seriously. 

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.ACTIVEPDF.COM	866.GO.TOPDF	4
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.233.9626	21
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX.HOST	35
COLDFUSIONHOSTING.COM	WWW.COLDFUSIONHOSTING.COM	888.250.8100	37
COOLFUSION	WWW.COOLFUSION.COM	631.737.4668	13
EDGE WEB HOSTING	WWW.EDGEWEBHOSTING.NET	1.866.EDGEWEB	15
EKTRON	WWW.EKTRON.COM/CFDJ	603.594.0249	39
FUSETALK	WWW.FUSETALK.COM/NEW	866.477.7542	2
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	25
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	52
IVIS TECHNOLOGIES	WWW.IVIS.COM	602.346.5045	3
JDJ WORKSHOP	WWW.SYS-CON.COM/EDUCATION	201.802.3058	31
MACROMEDIA	WWW.MACROMEDIA.COM/GO/DEVNET-CFDJ	877.460.8679	17
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CFDJ_TRAINING	877.460.8679	51
NETQUEST	WWW.NETQUEST.COM		9
NEW ATLANTA COMMUNICATIONS	WWW.NEWATLANTA.COM		6
PACIFIC ONLINE	WWW.PACIFICONLINE.COM	877.503.9870	23
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	29
SYS-CON SUBSCRIPTIONS	WWW.SYS-CON.COM/SUBOFFER.CFM	888.303.5282	46-47
TERATECH	WWW.CFCONF.ORG/CFUN-03/	301.881.1440	27

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

Don't Miss CFDJ's Next Issue!



Text File Parsing with CFML...

How to use text file parsing to automate processes or allow Web site users to make "bulk uploads."

Screenshots Made Easy...

Utilize ColdFusion and a third-party tool to make your screenshots a snap.

Design Patterns in the CF Universe...

The Strategy Pattern.

Flash and CFMX...

Putting Flash and CFMX on the desktop (thick client) using Screenweaver.

Site Executive 3.0 Content Management System

Web content management with fine style control

Site Executive 3.0 promises ease of use, standardization, rapid site implementation, rights- and role-based security, versioning, workflow, and easy custom module implementation. It largely succeeds in these areas. Load balancing and failover are also supported. This version does not have true localization support, but it's planned for future versions. Double-byte characters are supported.

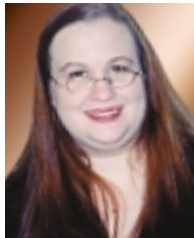
The installation process sets up the required database in your system. CFMX and the target database must be working and communicating properly and DNS must be responding. Parts of the installation process will fail if the server doesn't respond properly. Additional sites may be added within the Web application, but DNS must be handled externally.

Administration and Site Security

It is extremely straightforward to add new users and new user groups or to add permissions to documents and directories. The three permissions sets (read, write, and admin) can be applied to individual content, or recursively starting at any level of the site. The read role may preview unapproved pages and generate history reports. Write permission provides access to all content management features, and admin permission provides additional template, user, and site management functions.

Content Management Interface

The ActiveX WYSIWYG template and page editor is somewhat basic, choosing to encourage ease and stylesheet use over extended functionality. The familiar Office-like dropdowns allow the user to select text, then select a style from the



By Jennifer Larkin

menu. The inability to change text at will encourages standardization across templates but may irritate users who want nonstandard text. The editor also contains an easy-to-use table builder and pop-up windows that stay on top until the user closes them.

There is no ability to paste HTML into the template builder, so the Web designer will have to re-create the templates in the administrator. Also, the body content is not surrounded by the template in the interface, requiring the user to close the editing screen to preview the content within the template. Uploading images must be done with the editor closed, so if an expected image is missing, the user must complete several steps in order to upload the image.

The system allows separate stylesheets for the template and page content, encouraging standardization by limiting style differences in the page content. Only the default stylesheet can add new styles, and these cascade to other stylesheets. The stylesheet builder does not handle link styles, which are handled per template. Applied styles will not override link colors, making it virtually impossible to enable links in more than one color per template. Figure 1 shows the template builder displaying several content mod-

ules and the styles dropdown, and Figure 2 shows the page content editor with an "open and on top" module customization form.

Versioning and Workflow

Each time the user saves changes to a template or to page content, a new version is created. This becomes the editable version, with other versions being archived. An archived version can be reinstated by selecting it from the Versions dropdown. This will archive the current version and copy the selected archive version into a new editable version.

The approval workflow is set by the administrator and may be set per site, per directory, per page, or per template. Workflow is normally set recursively, applying site-wide workflow rules, which can be overridden per item. An unlimited number of ordered approval steps can be implemented, with final publication by the original author. The reporting tool lists what is ready for approval or publication.

Features and Modules

Custom 404 pages can be applied by server, by site, and by directory, recursively or individually. Some modules, such as the HTML Passthrough and Object Inclusion, are available by page for the inclusion of static HTML pages, media files, and multimedia files. Site Map, Trail of Breadcrumbs, and Email a Friend are available by template, and some modules are available at both levels. The Flash presentation module takes advantage of some of CFMX's Flash integration, and a site search module is also included.

The custom form builder allows administrators to create a wide variety of forms, the results of which can be e-mailed or exported in delimited text, Excel, or XML file formats. The exports can be downloaded by the user or

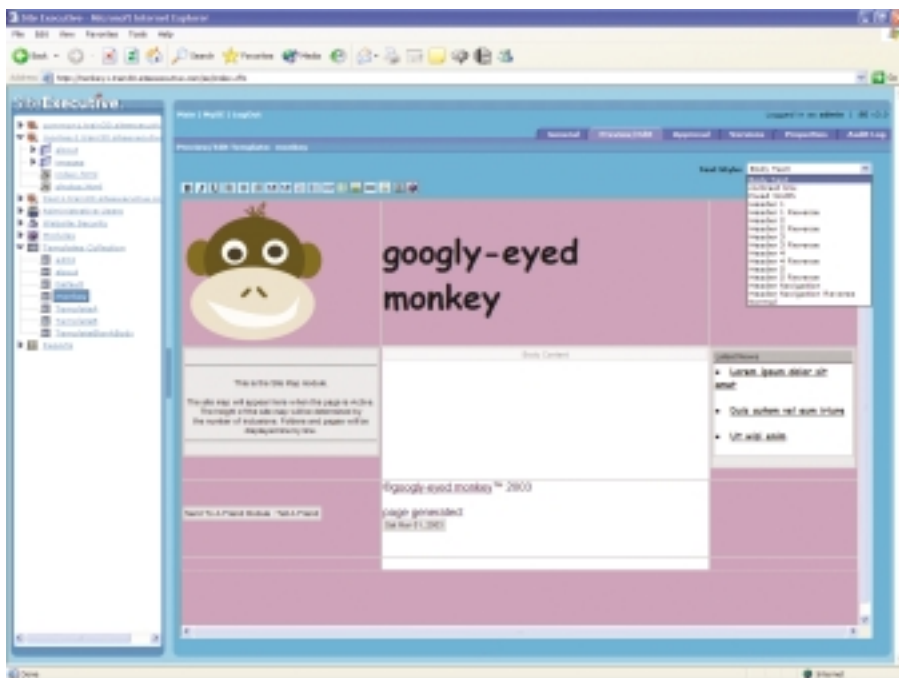


Figure 1: Template builder

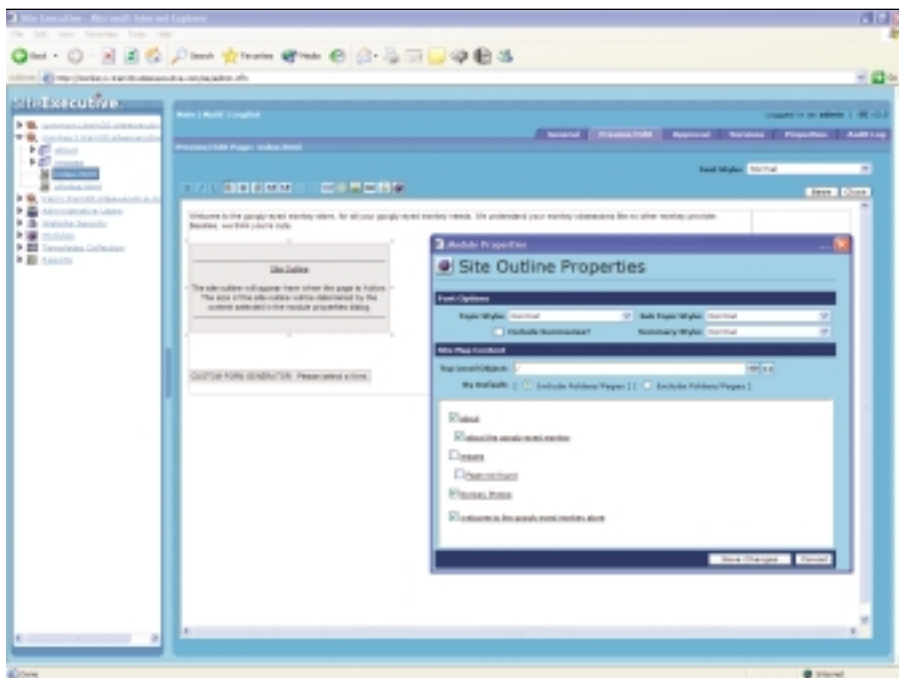



Figure 2: Page content editor

processed by a custom module. Despite the Site Map and DHTML menu modules, there is no easily implemented standard sidebar menu. Lack of automatic updates in the DHTML module requires the administrator to individually add new content links.

The custom module API allows structured module insertion while allowing

technical users to create any CFMX-supported functionality. Custom modules are registered at startup and on demand, with functionality much like custom tags. The API allows access to existing data and modules, easily handling versioned and approved data. However, there is a learning curve for creating complex modules.

Should You Buy It?

In essence, Site Executive 3.0 protects against site inconsistency at the expense of ease in creating a site with complex design. Some of these problems could be alleviated with custom modules. Some administration functions are easy to use, but others are difficult to understand or implement. Since administrative functions are largely for more technical users, this isn't a big problem. Easy access to approved version data makes custom extensions easy once you've gotten over the initial hump. As with all software, the purchase decision depends on the needs of the user. Site Executive 3.0 seems appropriate where there is significant disparity between the skill levels of administrators and content managers, and where fine style control is required. 

About the Author

Jennifer Larkin is the manager of the Bay Area ColdFusion User Group and a coauthor of ColdFusion 5.0 Certified Developer Study Guide. She is currently developing a ColdFusion intranet application for the city of San Francisco.

jlarkin@biliviv.com

Vitals

Site Executive 3.0

Systems Alliance, Inc.
34 Loveton Circle, Suite 102
Sparks, MD 21152
Phone: 410 584-0595, ext. 222
Toll Free: 877 797-8554
Fax: 410 584-0594
Web: www.siteexecutive.com

Specs:

CFMX only

Platform: Windows 2000, Red Hat Linux 7.2,
Solaris 2.5.1, Windows 2003 support planned
Databases: SQL 2000, Oracle 8i/9i

Pricing: Pricing per server with flexible bulk licensing. Starting price is \$35,000 with average multiserver installation at \$70,000. VAR programs and educational/nonprofit pricing discounts are available.

Test Environment:

Dell PowerEdge 1650, Pentium 3 1.13GHz processor, Windows 2000 Server OS, 512MB RAM, SQL 2000 on same server

SUBSCRIBE TODAY TO MULTI

Go To WWW.SYS-CON.COM/suboffer.cfm



and receive your
FREE CD Gift
Package via
Priority Mail



Each CD is an invaluable developer resource packed with important articles and useful source code!

Pick the CDs to go with your Multi-Pack order

► Pick one CD with your 3-Pack order

► Pick two CDs with your 6-Pack order

► Pick three CDs with your 9-Pack order



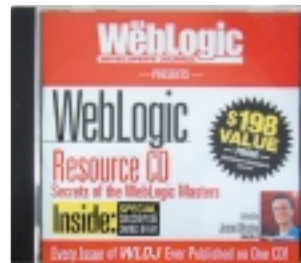
More than 1,400 Web services and Java articles on one CD! Edited by well-known editors-in-chief Sean Rhody and Alan Williamson, these articles are organized into more than 50 chapters on UDDI, distributed computing, e-business, applets, SOAP, and many other topics. Plus, editorials, interviews, tips and techniques!

LIST PRICE \$198



The most complete library of exclusive JDI articles compiled on one CD! Assembled by JDI Editor-in-Chief Alan Williamson, more than 1,400 exclusive articles are organized into over 50 chapters, including fundamentals, applets, advanced Java topics, Swing, security, wireless Java... and much more!

LIST PRICE \$198



The most complete library of exclusive WLDJ articles ever assembled! More than 200 articles provide invaluable information on "everything WebLogic", including WebLogic Server, WebLogic Portal, WebLogic Platform, WebLogic Workshop, Web services, security, migration, integration, performance, training...

LIST PRICE \$198

- ☐ Web Services Resource CD
- ☐ Java Resource CD
- ☐ WebLogic Resource CD
- ☐ ColdFusion Resource CD
- ☐ XML Resource CD
- ☐ WebSphere Resource CD
- ☐ Linux Resource CD

Your order will be processed the same day!



The most complete library of exclusive CFDJ articles on one CD! This CD, edited by CFDJ Editor-in-Chief Robert Diamond, is organized into more than 30 chapters with more than 400 exclusive articles on CF applications, custom tags, database, e-commerce, Spectra, enterprise CF, error handling, WDDX... and more!

LIST PRICE \$198



The largest and most complete library of exclusive XML-Journal articles compiled on one CD! Edited by well-known Editors-in-Chief Ajit Sagar and John Evdemon, these articles are organized into more than 30 chapters containing more than 1,150 articles on Java & XML, XML & XSLT, <e-BizML>, data transition... and more!

LIST PRICE \$198



The most up-to-date collection of exclusive WSDJ articles! More than 200 articles offer insights into all areas of WebSphere, including Portal, components, integration, tools, hardware, management, sites, wireless programming, best practices, migration...

LIST PRICE \$198



An up-to-the-minute collection of exclusive Linux Business & Technology articles plus Maureen O'Gara's weekly LinuxGram, which keeps a finger on the pulse of the Linux business community. Edited by LBT's Editor-in-Chief Alan Williamson, these articles cover migration, hardware, certification, and the latest Linux-related products. Available June 2003!

LIST PRICE \$198

Subscribe Online Today

PLE MAGAZINES ONLINE

AND SAVE UP TO \$400 AND RECEIVE UP TO 3 FREE CDs!



TO ORDER

- Choose the Multi-Pack you want to order by checking next to it below.
- Check the number of years you want to order.
- Indicate your location by checking either U.S., Canada/Mexico or International.
- Then choose which magazines you want to include with your Multi-Pack order.

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.

Linux Business & Technology

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 /	Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

Java Developer's Journal

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

Web Services Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

.NET Developer's Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

XML-Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

WebLogic Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

ColdFusion Developer's Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

Wireless Business & Technology

U.S. - Two Years (24) Cover: \$144	You Pay: \$89 /	Save: \$55 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$49.99 /	Save: \$22
Can/Mex - Two Years (24) \$192	You Pay: \$139 /	Save: \$53 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$79.99 /	Save: \$16
Intl - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

WebSphere Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

RECEIVE
YOUR DIGITAL
EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to **\$275⁰⁰**
Pay only \$175 for a 1 year subscription plus a **FREE CD**

- 2 Year - \$299.00
- Canada/Mexico - \$245.00
- International - \$315.00

6-Pack

Pick any 6 of our magazines and save up to **\$350⁰⁰**
Pay only \$395 for a 1 year subscription plus 2 **FREE CDs**

- 2 Year - \$669.00
- Canada/Mexico - \$555.00
- International - \$710.00

9-Pack

Pick 9 of our magazines and save up to **\$400⁰⁰**
Pay only \$495 for a 1 year subscription plus 3 **FREE CDs**

- 2 Year - \$839.00
- Canada/Mexico - \$695.00
- International - \$890.00

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

ColdFusion User Groups

For more information go to...

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

New England

New Hampshire
Northern N.E. MMUG
www.mmug.info

Massachusetts
Boston, MA CFUG
www.cfugboston.org

Rhode Island
Providence, RI CFUG
www.ricfug.com

Vermont, Montpelier
Vermont CFUG
www.mtbytes.com/dfug/index.htm

Midatlantic

New Jersey, Raritan
Central New Jersey CFUG
www.freecfm.com/cjcfug/index.cfm

New York
Albany, NY CFUG
www.anycfug.org

New York
Long Island, NY CFUG
www.licfug.org

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

New York
Rochester, NY CFUG
www.roch-cfug.org

New York
Syracuse, NY CFUG
www.cfugcny.org

Pennsylvania, Harrisburg
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Philadelphia, PA CFUG
www.pacfug.org

Pennsylvania
Pittsburgh, PA CFUG
www.orbwave.com/pgchcfug

Pennsylvania
State College, PA CFUG
www.cfug-sc.org

Southern

Alabama
Birmingham, AL CFUG
www.bcfug.org

Alabama
Huntsville, AL CFUG
www.nacufug.com

Delaware, Dover
Delaware CFUG
www.decfug.org

Delmarva, Dover
Delmarva CFUG
www.delmarva-cfug.org

Florida
Gainesville, FL CFUG
<http://plaza.ufl.edu/aktas>

Florida
Orlando, FL CFUG
www.cforlando.com

Florida
Tallahassee, FL CFUG
www.tcfug.com

Florida
Tampa, FL CFUG
www.tbcfug.org

South Florida
South Florida CFUG
www.cfug-sfl.org

Georgia, Atlanta
Atlanta, GA CFUG
www.acfug.org

Georgia, Atlanta
Georgia CFUG
www.cfugorama.com

Georgia
Columbus, GA CFUG
www.vcfug.org

Kentucky
Louisville, KY CFUG
www.loulexcfug.com

Louisiana
New Orleans, LA CFUG
www.nocfug.org

Maryland
Annapolis, MD CFUG
www.ancfug.com

Maryland
Baltimore, MD CFUG
www.cfugorama.com

Maryland
Broadneck H. S. CFUG
www.cfug.broadneck.org

Maryland, Bethesda
Maryland CFUG
www.cfug-md.org

Maryland
California, MD CFUG
<http://smdcfug.org>

North Carolina
Charlotte, NC CFUG
www.charlotte-cfug.org

North Carolina
Fayetteville, NC CFUG
www.schoolink.net/fcfug/

North Carolina
Raleigh, NC CFUG
www.ccfug.org

Oklahoma
Oklahoma City, OK CFUG
<http://idgweb4.ouhsc.edu/cfug>

Oklahoma
Tulsa, OK MMUG
www.tulsacfug.org

Tennessee
Memphis, TN CFUG
<http://cfug.dotlogix.com>

Tennessee
Nashville, TN CFUG
www.ncfug.org

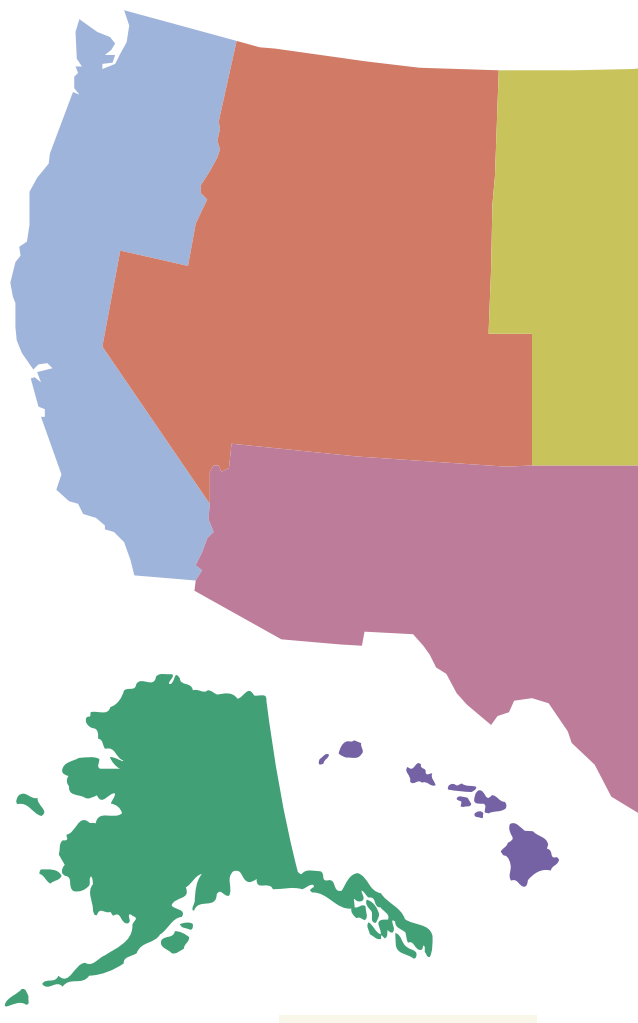
Virginia
Hampton Roads CFUG
www.hrcfug.org

Virginia
Northern Virginia CFUG
www.cfugorama.com

Virginia
Richmond, VA CFUG
<http://richmond-cfug.btgi.net>

Virginia, Roanoke
Blue Ridge MMUG
www.brmug.com

Washington D.C.
Washington, D.C. CFUG
www.cfugorama.com



Midwest

Northern Colorado
Northern Colorado CFUG
www.nccfug.com

Colorado Hamilton M.S. CFUG
<http://hamilton.dpsk12.org/teachers/team-c/intro.html>

Illinois, Champaign
East Central Illinois CFUG
www.ecicfug.org

Illinois, Chicago
Chicago, IL CFUG
www.cfugorama.com

Indiana
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana, South Bend
Northern Indiana CFUG
www.ninmug.org

Iowa
Des Moines, IA CFUG
www.hungrycow.com/cfug/

Michigan, Dearborn Mmaniacs
CFUG <http://ciwebstudio.com/mmania/mmania.htm>

Michigan, East-Lansing
Mid Michigan CFUG
www.coldfusion.org/pages/index.cfm

Minnesota, Minneapolis
Twin Cities CFUG
www.colderfusion.com

Rockies

Montana, Helena
Montana CFUG
<http://montanacfug.site-o-matic.com>

Nevada
Las Vegas, NV CFUG
www.snfcug.com

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

Wyoming, Jackson
Wyoming MMUG
www.wyfcug.org

West Coast

California
Bay Area CFUG
www.bacfcug.net

California, Fresno
Central California CFUG
www.cccfcug.com

California, Inland Empire
Inland Empire CFUG
www.sccfcug.org

California
Los Angeles CFUG
www.sccfcug.org

California
Orange County CFUG
www.sccfcug.org

California
Sacramento, CA CFUG
www.saccfcug.org

California
San Diego, CA CFUG
www.sdcfcug.org

Southern California
Southern California CFUG
www.sccfcug.org

Oregon
Eugene, OR CFUG
www.EugeneCFUG.org

Oregon
Portland, OR CFUG
www.pdxcfug.org

Alaska

Alaska
Anchorage, AK CFUG
www.akcfug.org

Hawaii

Hawaii, Honolulu
Hawaii CFUG
<http://cfhawaii.org>

Southwest

Missouri
Kansas City, MO CFUG
www.kcfusion.org

Missouri
St. Louis, MO CFUG
www.psiwebstudio.com/cfug

Nebraska
Omaha, NE CFUG
www.necfug.com

Ohio, Columbus
Ohio Area CFUG
www.oacfcug.org

Ohio
Mid Ohio Valley MMUG
www.movcfug.org

Wisconsin
Milwaukee, WI MMUG
www.metromilwaukee.com/usr/cfug

Arizona
Phoenix, AZ CFUG
www.azcfug.com

Arizona
Tucson, AZ CFUG
www.tucsoncfug.org

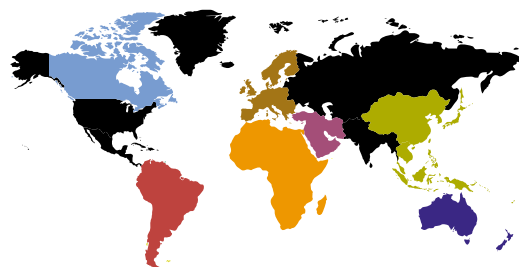
Texas
Austin, TX CFUG
<http://cftexas.net>

Texas
Dallas, TX CFUG
www.dfwcfug.org

Texas
San Antonio, TX CFUG
<http://www.samcfug.org>

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.



Africa

South Africa, Cape Town
Cape Town, South Africa CFUG
www.coldfusion.org.za

South Africa, Johannesburg
Johannesburg, South Africa CFUG
www.coldfusion.org.za

Central Europe, Munich
Central Europe CFUG
www.cfug.de

Finland, Helsinki
Finland CFUG
www.cfug-fi.org

France, Valbonne
France CFUG
www.cfug.fr.st

Germany, Frankfurt
Frankfurt CFUG
www.coldfusion-frankfurt.de

Ireland
Belfast, Ireland CFUG
www.cfug.ie

Ireland
Cork, Ireland CFUG
<http://viewmylist.com/cork>

Italy, Bologna
Italy CFUG
www.ingenium-mmug.org

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

Switzerland
Martin Bülmann, Switzerland CFUG
www.cfug.ch

United Kingdom, London
UK CFUG
www.ukcfug.org

United Kingdom
Northern England CFUG
www.cfug.org.uk

Asia

Malaysia, Kuala Lumpur
Malaysia CFUG
www.coldfusioner.com

Thailand
Bangkok, Thailand CFUG
<http://thaicfug.tei.or.th>

Japan, Urayasu-city Japan CFUG
<http://cfusion.itfrontier.co.jp/jcfug/jcfug.cfm>

Australia

Australia-Melbourne
Australian CFUGs
www.cfug.org.au

Canada

Canada
Edmonton, AB CFUG
<http://edmonton.cfug.ca>

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Montreal, QC CFUG
www.kopanas.com/cfugmontreal

Canada
Ottawa, ON CFUG
www.cfugottawa.com

Canada, Ottawa (HS group)
Ecole Secondary CFUG
www.escgarneau.com/gug

Canada
Toronto, ON CFUG
www.cfugtoreto.org

Canada
Vancouver, BC CFUG
www.cfug-vancouver.com

Canada
Vancouver Island CFUG
www.cfug-vancouverisland.com

Middle East

Pakistan Edu, Lahore Cantt
Pakistan Educational CFUG
www.cfugpakistan.org

Pakistan, Lahore
Pakistan CFUG
www.cfugpakistan.org

Saudi Arabia, Riyadh
CFUG Saudia
www.cfugsaudia.org

Turkey, Izmer
Turkey CFUG
www.cftr.net

Europe

Belgium, Brussels
Belgium CFUG
www.cfug.be

South America

Brazil
Rio de Janeiro CFUG
www.cfugrio.com.br

NTT DoCoMo to Embed Macromedia Flash Technology into i-mode Service
(Tokyo) – Japanese mobile phone operator NTT DoCoMo, Inc. (the world's leading mobile communications



company with more than 45 million customers), and U.S. software maker Macromedia, Inc., have announced

that the two companies have reached an agreement to jointly deliver Macromedia Flash technology to i-mode, DoCoMo's proprietary 2G/3G mobile Internet platform, in Japan. The technology will be embedded into the new i-mode handsets, starting with the 505i series, which is scheduled for release later this year.

"NTT DoCoMo is pleased to be able to implement Macromedia Flash technology into the i-mode service," said Takeshi Natsuno, managing director of the i-mode planning department at DoCoMo. "Because content providers are already familiar with Macromedia's widely used Internet technology for developing Web content and applications, we expect dynamic new content to be created for the i-mode service, which will further enhance the mobile-Internet user experience."

By incorporating Macromedia Flash in its i-mode service, DoCoMo will provide i-mode users with access to a much broader range of rich content and applications.

Easy-to-use graphical interfaces will allow users to simply click on one screen to go directly to specific information, eliminating the need to navigate through multilayered menus.

"With more than 98% penetration, Macromedia Flash has become a staple on the Internet because it has enabled memorable customer experiences that have delivered true business results," said Rob Burgess, chairman and CEO, Macromedia. "Working with leaders like DoCoMo, we are now setting the standard for amazing mobile experiences which bring relevant, engaging information to consumers no matter where they are."

The agreement was finalized with the signing of a license agreement between DoCoMo and Macromedia Netherlands BV, a wholly owned subsidiary of the U.S. parent company.

Native Deployment of CFML on .NET Offered
(Atlanta) – New Atlanta Communications, a leading producer of server-side technologies, has announced BlueDragon for .NET, the newest edition of its product family for deploying ColdFusion Markup Language (CFML) Web applications onto the Java 2, Enterprise Edition (J2EE) and Microsoft .NET platforms.

New Atlanta has also announced that BlueDragon Server, its stand-alone CFML application server implemented in Java, is now available, free, for development and deployment.

New Atlanta, through the BlueDragon product family, is committed to helping ensure that substantial investments in CFML code and skill sets will not become

marginalized or devalued due to corporate adoption and standardization on the .NET and J2EE technology platforms.

"Most organizations today are faced with choosing between J2EE and .NET as their strategic technology platform," said Vince Bonfanti, president of New

Atlanta Communications.

"This decision can be particularly troublesome for organizations with significant resources invested in CFML.

BlueDragon gives these organizations the freedom to choose either J2EE or .NET with confidence that their CFML applications can be integrated and deployed on either platform."

BlueDragon for .NET will be available for production use in the second quarter of 2003.



i-Technology's First Full-Time Canary?

By *CFDJ News Desk*

What do the following Internet technologists all have in common: Steve Wozniak, Raymond Ozzie, and Jeremy Allaire? The answer is that all fathered a pioneering technology solution...took it to worldwide market leadership...and then moved on.

Just as "Woz" moved beyond Apple, just as Ozzie moved beyond Lotus, so Jeremy Allaire has – after an astonishingly successful stint at the technical helm – moved beyond Macromedia, and beyond ColdFusion, the child he reared and nurtured from its birth through infancy, childhood, adolescence, and into adulthood where it is now comfortably ensconced as ColdFusion MX.

The announcement didn't come entirely out of the blue, since any i-technologist of the caliber of Allaire is always going to be tempted by fresh pastures, new horizons, and greater challenges. But the exact details of his next stopping place on one of the most successful technology journeys of the Internet era weren't known until last month.

When he spoke exclusively to *CFDJ*, it was – literally – on his first day at his new haunt, which is Cambridge, MA-based General Catalyst Partners, an early-stage private equity firm that henceforth will be the gymnasium in which he keeps his technological wits in shape, as "technologist-in-residence."

"It's a great opportunity to learn a lot," says Allaire with characteristic modesty. "Allaire Corp. and Macromedia, Inc., were great but it was only one experience, whereas the greater technology landscape is very broad."

Scoping out the contours of that landscape, particularly with regard to the future, will be Allaire's special mission. Does that make him the i-technology world's equivalent of a miner's canary, *CFDJ News Desk* asks him, as in the bird that they took down into the coal mines in days gone by, to check if the mine shaft at its furthest extremes was safe to work in for everybody else? (If it wasn't, the canary in the cage was the early indicator. It would expire.)

Allaire chuckles. He recognizes the analogy, and accepts that he may well go deeper down the i-technology mine shaft, and further along its tunnels, than many. But evaluating emerging technologies, he says, will hopefully be less risky to life and limb.

Or he won't be around to participate in "firm building," the term he uses for his role at General Catalyst Partners in which he'll be helping early-stage companies with things such as their product road maps, actively leveraging Allaire's expertise as a technology founder.

CFDJ wishes him the best of luck with his future endeavors and is happy to report that he will remain involved with

Macromedia as a Founder Emeritus. www.generalcatalyst.com



Jeremy Allaire

MACROMEDIA

www.macromedia.com/go/cfdj_training

INTERMEDIA.NET

www.intermedia.net