

COLD FUSION Developer's Journal

ColdFusionJournal.com

December 2002 Volume: 4 Issue: 12

Announcing...

web services **EDGE**
conference & expo

March 18-20, 2003

Boston, MA

HYUNDAI CONVENTION CENTER

SEE PAGE 39 FOR DETAILS

Editorial

DevCon 2002

Robert Diamond page 5

Q&A

Ask the Training Staff

Bruce Van Horn page 7

CF Tutorial

Extending ColdFusion

Raymond Camden page 28

ColdFusion News

Macromedia

Contribute Makes

Web Site Updates Easy

Montara and New

Atlanta Partnership

page 50

RETAILERS PLEASE DISPLAY
UNTIL FEBRUARY 28, 2003

\$8.99US \$9.99CAN



SYS-CON
MEDIA



See
pg.8

Development Tools: Flash Remoting with Macromedia's DesDev Feed

A simple way to pull dynamic content into Flash

Dennis Baldwin

16

<BF on CF>: 'But It's Free!'

Time to clear the air and set the record straight

Ben Forta

20

CF and MS: Serving Word

HTML, XML, and CSS make it easy

Samuel Neff

24

Show Report: DevCon 2002

Face to face with the CF Community

Simon Horwith

32

Foundations: All About Arrays

They're not too geeky

Hal Helms

36

Product Review: BlueDragon

Another option for CF developers

Phil Cruz

40

CFCs: Creating and Accessing Collection Files

ColdFusion components make it simple

Steve Parks

44

New Atlanta Communications

www.newatlanta.com

CFXHosting

www.cfxhosting.com

activePDF
www.activepdf.com

editorial advisory board

Jeremy Allaire, *CTO, macromedia, inc.*
 Charles Arehart, *CTO, systemanage*
 Michael Dinowitz, *house of fusion, fusion authority*
 Steve Drucker, *CEO, fig leaf software*
 Ben Forta, *products, macromedia*
 Hal Helms, *training, team 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

Charles 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

Dennis Baldwin, Raymond Camden,
 Phil Cruz, Ben Forta, Hal Helms, Simon Horwith,
 Samuel Neff, Steve Parks,
 Courtney E. Payne, Bruce Van Horn

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

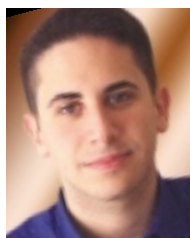
Copyright © 2002 by SYS-CON Publications, Inc.
 All rights reserved. No part of this publication may
 be reproduced or transmitted in any form or by any
 means, electronic or mechanical, including photocopy
 or any information, storage and retrieval system,
 without written permission.
 For promotional 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.



DevCon 2002: A Positive Indicator for CF

The ColdFusion world traveled down to Disney World this October – without even having to win a major sporting event first. The occasion was Macromedia DevCon 2002 – the world's largest gathering of all things Macromedia – which took place October 27–30. It was a very successful conference on all fronts, with a keynote by Macromedia chairman and CEO Rob Burgess, and appearances by Ben Forta, Jeremy Allaire (via Flash video), and other Macromedia executives and product managers.

The theme of the conference was “Architecting a New Internet Experience,” and many of the sessions stuck to that theme, showing how you can combine Macromedia's products to build better Internet applications and Web sites. The MX product line is now in full swing with one third of Macromedia's revenue coming from the new set of products.



By Robert Diamond

Many conference attendees who I spoke with were advanced ColdFusion developers who had been slowly dragged into the worlds of Dreamweaver and Flash and, now there, didn't know how they had survived before.

Things are going very well in the world of ColdFusion these days. More than 70% of the 2,500+ conference attendees were there to learn all they could about CF in three action-packed days. An interesting statistic is that more than half of the attendees had never been to a conference or user group meeting before, so there was a lot of new blood, and a high level of excitement about all that could be done with CF.

Almost 30% of ColdFusion MX purchases are by new users, with 70% coming from upgrades. That 30% number is a sign that ColdFusion is doing very well, and Macromedia is committed to continuing that success. They are the number-two software company in research and development spending, and the reasons are evident with their recent and upcoming releases.

Conference attendees were able to see a lot of real-world examples showcasing the new

Flash server, as well as take a sneak peek at Macromedia's next release – Contribute. It's a product that we'll be covering more in the upcoming months because it's designed to make developers' lives easier – and who of us doesn't like that?

I've always advocated attending conferences, and I think everyone at DevCon would agree. It provided fantastic opportunities for learning, networking, sharing ideas, solving problems, and even taking some time to have fun at a Macromedia special event at Universal Studios Florida. Next year's DevCon has been announced for September 7–10, 2002, at the San Diego Convention Center – I hope to see many of you there!

This month we've got another information-packed issue of **CFDJ**, and we're working very hard to make 2003 our best year yet. There is one feature in this month's issue that I wanted

—continued on page 42

About the Author

Robert Diamond is vice president of information systems for SYS-CON Media, and editor-in-chief of both **CFDJ** and **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



Rob Burgess, chairman and CEO, Macromedia (left)
 Robert Diamond, editor-in-chief, CFDJ (right)



Phil Costa, Macromedia's senior product manager (left)
 Ben Forta, Macromedia's senior product evangelist, ColdFusion (right)

FuseTalk

www.fusetalk.com/new

Ask the Training Staff

A source for your CF-related questions

Can you believe it's December already? 2002 was certainly a year of big changes for many CFers as Macromedia released the new MX products. Many of us scrambled to come up to speed on the new changes in CFMX. However, the fundamentals of CF programming didn't really change. The newcomer to CF still has to learn the basic CF tags, functions, and logic that were in previous versions before tackling the new features. With that in mind, here are two questions that came in recently that address some basics that CFers deal with every day.

Q: I'm having trouble with a nested loop. The code shown in Listing 1 should – if I understand the way CF does loops – output both “house” and “mouse” in that order. However, when I run the code, it only outputs “house”. Any insight?



By Bruce Van Horn

Listing 1

```
<cfloop list="car, boat, house, plane,
mouse" index="outer">
    <cfloop list="mouse, house, dog, cat"
index="inner">
        <cfif #outer# IS
#inner#><cfoutput>#inner#</cfoutput></cfif>
    </cfloop>
</cfloop>
```

A: Great question. The answer is easy, but elusive. The problem is the spaces in your lists. In the first (outer) list, the value of element 1 is “car”, but the value of element 2 is “boat” (notice the space that follows the comma). CF includes the spaces as part of the value of the list element. In the first list, the value that you think is “mouse” is actually “ mouse”, whereas it is “mouse” in the second list. Therefore “ mouse” will never equal “mouse” and thus doesn't appear in the output.

The solution could simply be to remove all spaces from your lists, but that often isn't practical if you aren't in control of the list's source. The better solution is to use the Trim() function when evaluating your list elements. The Trim() function eliminates any leading or trailing spaces from a value. As a side note, the #s

are not necessary inside a CFIF tag (or any CF tag or function for that matter) when evaluating the value of a variable, and your CFOUTPUTs should be outside of the loops. Listing 2 will give you the result you desire.

Listing 2

```
<cfoutput>
<cfloop list="car, boat, house, plane, mouse"
index="outer">
    <cfloop list="mouse, house, dog, cat"
index="inner">
        <cfif Trim(outer) IS
Trim(inner)>#inner#</cfif>
    </cfloop>
</cfloop>
</cfoutput>
```

Q: I'm looking for a way to do a database search from a form page where the user can input more than one item in a text input box (separated by commas) and have the query search of each item in the list. For example, in

—continued on page 42

About the Author

Bruce Van Horn is president of Netsite Dynamics, Inc., a Certified ColdFusion Developer/Instructor, and a member of the CFDJ International Advisory Board.

bruce@netsitedynamics.com

Super Wizards

Creating wizards to collect well-defined data is pretty straightforward. But sometimes we're confronted with the task of creating a set of forms to collect data where the final structure of that data is unknown or inconsistent. Here, I'll describe a method I currently use to effectively collect such data by dynamically building and managing an underlying ColdFusion data structure to store and manipulate the data. Later on, we'll discuss a simplified example that demonstrates the method I used.

I once received a request from a client to build an application that would allow data collection by way of HTML form submissions. "No sweat," I thought. Then they began to go further into detail. The more they talked, the more concerned I became.

The client explained that the data elements were to be interrelated in a "multilevel, top-down, one-to-many hierarchy" and that the user supplying the data should have the ability to easily add, modify, and remove elements from this hierarchy. Then we realized that there would be too much data to fit on one page. It would

have to be broken up. "OK," I thought, "this is a bit more complex than I expected. Still, I should be able to build a relational database to represent the hierarchy and, while the user steps through the multipage form, or wizard, I can pull data from the database for display and modification. Piece o' cake."

I left the meeting with my requirements and a preliminary plan. As I sat in my (very) meager cube, I began to think about the possible implications of this plan. I figured that hitting the database to retrieve data every time the user moved from one form page to another wouldn't be a very efficient solution. And how would I record intermediate changes to the data? Would I check to see if the data had been modified and update the database with changes between pages? Maybe store the intermediate changes in temporary tables? It became clear that this "solution" would provide for a poor user experience and definitely wouldn't scale. I realized that what I needed to do was to initially load the entire data structure into ColdFusion

Server memory and manipulate that interim or temporary data structure there, in memory, without touching the database. I would then be able to commit changes to the database only when the user had completed all modifications to the data structure, not between pages.

The Building Blocks

Before we get into our scenario, it's assumed that you have a good understanding of ColdFusion arrays and structures. They will serve as the building blocks that we will use to represent our data structure.



By Courtney E. Payne

For our purposes here, we'll only be dealing with single-dimension arrays. Here is an example of basic array instantiation and manipulation:

```
<cfscript>
    variables.siteUsers = ArrayNew(1);
    variables.siteUsers[1] = "Jose Parilla";
    variables.siteUsers[2] = "Marco Koers";
    variables.siteUsers[3] = "Tina Paulino";

    variables.arrayLength = ArrayLen(variables.siteUsers); // = 3
    temp = ArrayDeleteAt(variables.siteUsers,2); // Delete Array
    Element 2
    variables.arrayLength = ArrayLen(variables.siteUsers); // = 2 & No
    More Marco
</cfscript>
```

Structures allow us to create and maintain key-value pairs so that we can group related data together under a single variable name. Structure keys can be referenced as associative arrays, or in “dot” notation, which we'll use here. Here is an example of basic structure instantiation using dot notation:

```
<cfscript>
    variables.sportsCar = StructNew();
    variables.sportsCar.year = "2003";
    variables.sportsCar.make = "Nissan";
    variables.sportsCar.model = "350Z";
</cfscript>
```

Data Persistence

In order for us to maintain our structure while the user is making changes to the data without storing it in a database, we're going to store it in the session scope. Doing so will cause ColdFusion to hold our in-work structure in server memory across requests of our users' sessions.

We all know how important it is to fully lock all shared-scope variable references. It's no different here. Not doing so will create the perfect condition for your session, application, or server variables to be read from and written to simultaneously, possibly corrupting server memory and eventually resulting in memory leaks and unexpected server behavior. We must lock. Here are basic examples of “exclusive” and “readonly” locks in use:

```
<cflock scope="session" type="exclusive" timeout=10
throwonetimeout="yes">
    <cfset session.data = ArrayNew(1)>
</cflock>

<cflock scope="session" type="readonly" timeout=10
throwonetimeout="yes">
    <cfset variables.array_length = ArrayLen(session.data)>
</cflock>
```

When an “exclusive” lock is encountered, ColdFusion always single-threads all access to the code within it. When a “readonly” lock is encountered, ColdFusion checks before proceeding to make sure that there isn't code executing within an “exclusive” lock of the same locking scope (or with the same lock name). If, indeed, there is an “exclusive” lock of the same scope processing

at that time, ColdFusion waits until that code completes before allowing access to the “readonly” lock. Otherwise, the code within “readonly” locks processes as normal, with negligible performance penalty. The timeout parameter specifies the maximum number of seconds to wait to gain access to that lock, not the maximum number of seconds to complete the code within it.

We're going to use a session variable to hold the structure in our example. If you're developing for a load-balanced Web server configuration, or if you think you may eventually move to one, use a database-stored client variable instead so your users won't lose data in the event that their requests are handled by more than one server. Of course, if you decide to use client variables, remember that you'll need to serialize your structure after each update using WDDX before storing as a client variable. We'll look at the use of the CFWDDX tag a bit later.

The Scenario

Let's say that we're running a small business and we need to track each of our employees' progress toward assigned objectives. In this simplified example, our tracking system will need to record information on three different entities: employees, roles, and objectives, each representing a progressively deeper “level” in our structure. A high-level view of this data structure would look something like Figure 1, with our company being the root node of our data structure.

Listing 1 creates a structure to hold our data (the source code for this article can be found at www.sys-con.com/cfdj/sourcec.cfm). In our example code, we have two employees, Danny and Kevin. Danny has two roles assigned to him, “Network Administrator” and “Coffee Preparer.” Kevin has only one role assigned to him, “Web Developer.”



Figure 1: High-level view of data structure

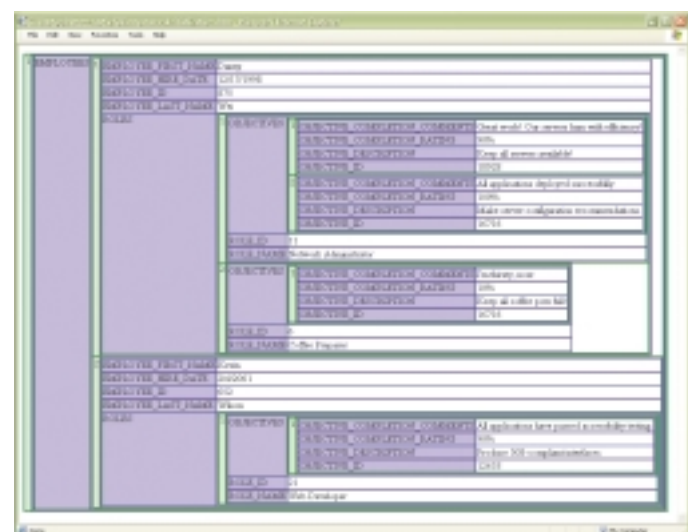


Figure 2: Organization of structure



Figure 3: Employee page



Figure 4: Role page



Figure 5: Objective page

There are two objectives assigned to the “Network Administrator” role and one objective assigned to the “Coffee Preparer” role, so Danny has three objectives. With his sole role of “Web Developer,” Kevin has only one objective. Using ColdFusion’s CFDUMP tag, we can get a better idea of how our new structure is organized in server memory (see Figure 2).

Remember, the whole point of managing our data in server memory is that we can divide our input fields into logical pages and allow users to modify data without having to save it to the database between pages. Let’s divide our form into three pages: an “Employee” page (see Figure 3 and Listing 2), a “Role” page (see Figure 4 and Listing 3), and an “Objective” page (see Figure 5 and Listing 4).

We’ll be able to move between pages freely, adding, deleting, and modifying elements, and those changes will be effected by directly manipulating our structure in ColdFusion server memory.

Building Our Basic Structure

Before we can do anything, we have to initialize our structure. We first need to create our first employee (see Listing 1 for full source code).

```
<cfscript>
variables.company = StructNew();
variables.company.employees = ArrayNew(1);
variables.company.employees[1] = StructNew();
variables.company.employees[1].employee_hire_date = "";
variables.company.employees[1].employee_first_name = "";
variables.company.employees[1].employee_last_name = "";
</cfscript>
```

Now that we have our root and first-level elements, company and employee, we need to create our first role.

```
<cfscript>
variables.company.employees[1].roles = ArrayNew(1);
variables.company.employees[1].roles[1] = StructNew();
variables.company.employees[1].roles[1].role_name = "";
```

```
variables.company.employees[1].roles[1].role_description = "";
</cfscript>
```

We now have our root, first-level, and second-level elements defined. Remember, each role has objectives assigned to it. So we need to create our first objective, the third-level element.

```
<cfscript>
variables.company.employees[1].roles[1].objectives =
    ArrayNew(1);
variables.company.employees[1].roles[1].objectives[1] =
    StructNew();
variables.company.employees[1].roles[1].objectives[1].objec-
    tive_description = "";
variables.company.employees[1].roles[1].objectives[1].objec-
    tive_completion_rating = "";
variables.company.employees[1].roles[1].objectives[1].objec-
    tive_completion_comments = "";
</cfscript>
```

We now have our basic structure defined. As an example, to access objective 7’s description, under role 3, belonging to employee 9, we could use this code.

```
#variables.company.employees[9].roles[3].objectives[7].objective_de-
scription#
```

To better visualize where we would be in our structure, see Figure 6.

Building Our Switchboard

We can now begin to build our forms and structure manipulation templates. We’re going to use self-submitting forms that post our data to a central, “switchboard” template that manages the manipulation of our structure and determines which of our three forms to display to the user.

Let’s use the form variable form.showPage to determine which form to display to the user and another form variable,

NetQuest

www.nqcontent.com

form.updatePage, to determine which piece of our structure needs to be updated when the user moves away from a page. We'll pass them between pages as hidden form fields. Values of 1, 2, or 3 will tell our switchboard to display or update the employee, role, or objective data, respectively.

We'll also keep track of our employee, role, and objective array indices with hidden form variables, form.employeeIndex, form.roleIndex, and form.objectiveIndex, respectively (for complete source code, see Listing 5).

As you can see in Listing 5, we have "Employees," "Roles," and "Objectives" form selector INPUT buttons on each page.

```
<input type="button" value="Employees"
onclick="document.forms.theForm.showPage.value=1;
document.forms.theForm.submit();">
<input type="button" value="Roles"
onclick="document.forms.theForm.showPage.value=2;
document.forms.theForm.submit();">
<input type="button" value="Objectives"
onclick="document.forms.theForm.showPage.value=3;
document.forms.theForm.submit();">
```

Each button uses JavaScript to update the showPage hidden form field (to tell our switchboard which form to display) and then submits the form. We're using a CFSWITCH/CFCASE construct to include the appropriate template based on the value of our form variable showPage.

Adding Elements to Our Structure

In each of our three form templates, we have an add button.

```
<input type="submit" value="Add New Employee" name="add">
```

The existence of the variable form.add in our switchboard template indicates that the user wants to add an element. But which one, an employee, role, or objective? The value of form.updatePage tips us off (for complete source, see Listing 5, index.cfm : Add An Element).

```
<cfif IsDefined("form.add")>
    <cfswitch expression="#form.updatePage#">
        <cfcase value=1>
            ...
        </cfcase>
        <cfcase value=2>
            ...
    </cfswitch>
```



Figure 6: View of structure

```
</cfcase>
<cfcase value=3>
    <cflock scope="session" type="exclusive" timeout=10
    throwtimeout="yes">
        <cfscript>
            variables.nextIndex = ArrayLen(session.company.
            employees[form.employeeIndex].roles[form.
            roleIndex].objectives) + 1;

session.company.employees[form.employeeIndex].roles[form.roleIndex].ob
jectives[variables.nextIndex] = StructNew();

session.company.employees[form.employeeIndex].roles[form.roleIndex].ob
jectives[variables.nextIndex].objective_description = "";

session.company.employees[form.employeeIndex].roles[form.roleIndex].ob
jectives[variables.nextIndex].objective_completion_rating = "";

session.company.employees[form.employeeIndex].roles[form.roleIndex].ob
jectives[variables.nextIndex].objective_completion_comments = "";

            form.objectiveIndex = variables.nextIndex;
        </cfscript>
    </cflock>
</cfcase>
</cfswitch>
</cfif>
```

Let's say we're adding a third-level element to our structure, an objective. The existence of the variable form.add and a form.updatePage variable value of 3 would tell our switchboard to execute the code in the third CFCASE block, in which the length of the current objectives array would be retrieved, incremented, and stored in the variable nextIndex. Then, just as we did when we initialized our structure, our new objective would be created, using nextIndex as the new objective index.

Deleting Elements from Our Structure

To delete elements, we also have a delete button in each of our form templates.

```
<cfif ArrayLen(variables.company.employees) GT 1>
    <input type="submit" value="Delete Employee" name="delete">
</cfif>
```

In our switchboard template, we use the combination of the existence of form.delete and the value of form.updatePage to determine whether we're deleting an element, and if so, which element to delete (for complete source, see Listing 5, index.cfm : Delete An Element). Here, we use the ArrayDeleteAt function to delete the current objective array element.

```
<cfelseif IsDefined("form.delete")>
    <cfswitch expression="#form.updatePage#">
        <cfcase value=1>
            <cflock scope="session" type="exclusive" timeout=10
            throwtimeout="yes">
                <cfscript>
                    if (ArrayLen(session.company.employees) GT 1)
```

```

        {garbage = ArrayDeleteAt(session.company.
            employees, form.employeeIndex);}
    </cfscript>
</cflock>
</cfcase>
<cfcase value=2>
    <cflock scope="session" type="exclusive" timeout=10
        throwontimeout="yes">
        <cfscript>
            if (ArrayLen(session.company.employees[form.
                employeeIndex].roles) GT 1) {garbage =
                ArrayDeleteAt(session.company.employees
                    [form.employeeIndex].roles, form.roleIndex);}
        </cfscript>
    </cflock>
</cfcase>
<cfcase value=3>
    <cflock scope="session" type="exclusive" timeout=10
        throwontimeout="yes">
        <cfscript>
            if
(ArrayLen(session.company.employees[form.employeeIndex].roles[form.
roleIndex].objectives) GT 1) {garbage = ArrayDeleteAt(session.com-
pany.employees[form.employeeIndex].roles[form.roleIndex].objec-
tives, form.objectiveIndex);}
        </cfscript>
    </cflock>
</cfcase>
</cfswitch>

```

If, for example, a user were to delete the 10th element in the array and the array had 10 total elements, the current array index (10) would become invalid, which would cause ColdFusion to throw an error if we tried to access that now nonexistent element in our form. As a safeguard against this, we always check the validity of each index in our switchboard template and reset where necessary (see Listing 5, index.cfm : Adjust Out Of Range Indices).

```

<cfscript>
    if (form.employeeIndex GT ArrayLen(variables.company.employees))
        {form.employeeIndex = ArrayLen(variables.company.employees);}
    if (form.roleIndex GT
        ArrayLen(variables.company.employees[form.employeeIndex].roles))
        {form.roleIndex =
        ArrayLen(variables.company.employees[form.employeeIndex].
            roles);}
    if (form.objectiveIndex GT
        ArrayLen(variables.company.employees[form.employeeIndex].
            roles[form.roleIndex].objectives)) {form.objectiveIndex =
        ArrayLen(variables.company.employees[form.employeeIndex].
            roles[form.roleIndex].objectives);}
</cfscript>

```

If an index is greater than the length of its array, we reset the index to the length of that array. Now we know that our array indices will always be valid.

Ektron

www.ektron.com/free

Don't Miss CFDJ's Next Issue!



A CFDJ Fifth Anniversary Retrospective...

In a special, expanded Journeyman column, Charlie Arehart will take a look at what's happened in the past 5 years in the world of ColdFusion – how CFDJ has evolved, the Allaire days, the merger, etc., complete with photos from our archives. Don't miss this issue that is sure to become a CF collector's item!

Understanding Anonymous Access and Integrated Windows Authentication and Applying It to Fusebox...

Excellent advice for developers and site administrators working in an environment using Macromedia CF running on a Windows server using Internet Information Server 4.0 or better.

ColdFusion MX Hardware Performance...

Find out the answer to the question "What kind of server should I set up to run ColdFusion MX?"

Web Services in a Flash...

Here's how to access a Web service using Flash Remoting and display the information in a rich Flash UI.

Design Patterns for ColdFusion...

The first of a series will target beginning CF developers.

Switching Between Elements

In addition to add and delete buttons on each form page, we also have an element selector SELECT box that allows users to switch to a particular employee, role, or objective element (for complete source, see Listing 4, `form_objectives.cfm` : Element Selector).

```
<select name="selectedObjectiveIndex" size=1
  onchange="document.forms.theForm.submit();">
<cfloop from=1
  to=#ArrayLen(variables.company.employees
    [form.EmployeeIndex].roles[form.Role
    Index].objectives)# index="i">
  <option <cfif form.ObjectiveIndex EQ i>SELECTED</cfif>
    value="#i#">Objective #i#
</cfloop>
</select>
```

The current objective array is looped over, creating an option in our objective element selector SELECT box, `selectedObjectiveIndex`, for each objective element in the current objective array. When the objective index is changed by the user, we use JavaScript to submit the form to our switchboard so that we can grab and display the selected element form to the user. In our switchboard template, we check for the existence of `selectedObjectiveIndex`, `selectedRoleIndex`, or `selectedEmployeeIndex` form fields. If they exist, we overwrite the appropriate element index so that the correct element is displayed (Listing 5, `index.cfm` : Switch Element Index).

Making the Data Stick

Now we have to make sure that when the user moves from one form page to another, the data that was entered persists. This is easy with the use of our element array indices, `employeeIndex`, `roleIndex`, and `objectiveIndex` and `updatePage` form variables (for complete source, see Listing 5, `index.cfm` : Update Data). Again, we use the value of `form.updatePage` to determine which block of data needs to be updated, then we simply update the data structure directly with the data entered by the user.

Committing Our Data

To store our data for later use outside of our current session, we can either serialize our structure using WDDX and stuff it in a database table or text file, or we can loop through our structure, inserting the data into a relational database. If you have requirements to run searches against your data, you'll definitely want to store your data in a relational database. Otherwise, it's simpler to just save and retrieve WDDX packets as whole units.

Even if we choose to store our data in a relational database, we can still leverage WDDX by maintaining a data archive that tracks each version of our data structure. Each time a user saves data, we can serialize our data structure with the CFWDDX tag, and stick it in an "archive" table with a timestamp, and possibly the ID of who committed the data. If the need ever arises to revert to a previous version, we can always restore an archived packet from the database.

For the purpose of our example, we're creating a dummy initial data structure to work with (Listing 1). In practice, you'll

be pulling your existing data structure from a database table or text file to view or make changes to it. For example, if you were to serialize and save your flattened data structure to a text file, you could read it back into ColdFusion and deserialize it using CFWDDX so that users could update it.

```
<cffile action="Read" file="c:\web\ourPacket.txt"
variable="wddxText">
<cflock scope="session" type="exclusive" timeout=10 throwtime-
out="yes">
  <cfwddx action="wddx2cfml" input="#wddxText#"
    output="session.company">
</cflock>
```

Pulling It All Together

We've just created a "Super Wizard" for the collection of our fictional company's employee objectives. With the relatively small amount of code we've written, we can build very complex representations of our company's personnel and objectives structure.


ODBC Limitations

If you're inserting and selecting WDDX packets from a database and your data structures grow to be very wide or very deep, you'll need to go into ColdFusion Administrator to select "Enable Retrieval of Long Text" and to raise the "Long Text Buffer Size" value under your ODBC data source settings to keep packets from being truncated when queried from the database.

Even after doing so, some database drivers may have limitations on the number of characters that can be returned in one query. To overcome such limitations, you may need to execute multiple queries to retrieve packets piece by piece from the database (with SQLServer's SUBSTRING function, for example) and concatenate the pieces in your data retrieval template before deserializing.

Wrapping Up

This technique can be applied to any application where structured data is collected. With it, you can produce and manage very complex hierarchies with structures and arrays nested many levels deep without bogging down your database server under load.

ColdFusion is very efficient at diving deep into and looping over complex data structures. Moving as much data manipulation and processing as possible over to ColdFusion, and away from your database, may help you avoid database-related performance bottlenecks in your applications. 

About the Author

Courtney E. Payne is a senior Web developer and project lead for Digicon Corporation's Web Services team currently working at the U.S. Department of Education. He also cofounded Communico Site Server, a producer of ColdFusion-based dynamic online communities.

cpayne@communicoserver.com

CoolFusion

www.coolfusion.com

Flash Remoting with Macromedia's DesDev Feed

A simple way to pull dynamic content into Flash

Iwanted to write an article that would be useful to the developer community. Since Flash is gaining ground as a powerful development tool, this article will focus on using Flash to create a rich UI hooked up to a ColdFusion component.

An effective Web site cannot exist without content, so this article will show you how to pull dynamic content into Flash. Pulling content from across the Web into Flash may be much easier than you expected. We're going to look at utilizing Macromedia's Designer & Developer (DesDev) XML feed. This XML feed will be pulled locally to our server, and then loaded into Flash. To accomplish this we'll be utilizing ColdFusion Components (CFCs), Flash Remoting MX, and Macromedia Flash MX components.

Before starting, please be sure to download the necessary files from www.devmx.com/cfdj/mm_desdev.zip. Once you've downloaded the files, unzip them to a directory named "cfdj" under the "wwwroot" of your Web server.

The ColdFusion Component

Let's take a look at the CFC we'll be building. We're going to create a file named "feeds.cfc" and save it to the "cfdj" folder under our Web server root "wwwroot". You can either create this file from scratch or use the file contained in the download. If you'd like to write the code yourself, then open your HTML editor of choice – Dreamweaver MX, Notepad, or something else. First off you need to know that the XML feed is located at the

By Dennis Baldwin

following URL: www.macromedia.com/desdev/resources/macromedia_resources.xml. It's a good idea to look at the structure of the XML document before you attempt to load it into Flash and parse it. The document has the structure shown in Listing 1.

Each resource node has a type attribute that specifies what type of content is contained within the node. This type attribute can consist of articles, components, extensions, tips, tutorials, and so on. There are four child nodes that consist of title, author, URL, and product. The product node has a name attribute that displays which software product the content relates to (ColdFusion, Macromedia Flash, Dreamweaver, Fireworks, etc.). We'll loop through each resource node of the XML document and parse the nodes inside of Flash. Before we tackle this we need to develop our CFC that will pull the feed from Macromedia.com to our local server. The CFC is very simple and can be seen in Listing 2.

First, make sure that the access attribute of CFFUNCTION is set to remote. This will allow Flash Remoting (or any remote service for that matter) to access the getDesDevFeed method of our component. This method of the CFC calls the XML file from Macromedia's server using CFHTTP. The CFPROCESSINGDIRECTIVE tag is

used to trim any whitespace from the XML document. Flash tends to fuss about whitespace within XML documents so we'll also be sure to handle this in our ActionScript code too.

Next we set the MIME type of the page using CFHEADER. The MIME type is automatically determined by CFHTTP and, in our case, is "text/plain." The XML contents are then returned to the caller (Flash in our case) using the CFRETURN tag.

That's it! It's as simple as that. Now let's take a thorough look into the Flash side of things.

The Flash Movie

Due to security limitations, the XML document cannot be loaded directly into Flash unless the file resides in the same domain. This is why we created our CFC, which pulls the XML file locally using CFHTTP. Now we can access and load the content as if it existed on our own server. This is a very handy technique and can be used to pull other third-party news feeds such as ones from Slashdot, Moreover, and any feed that comes in an XML format. The CFC could actually be developed to incorporate other news feeds and Web services. Take a look at the ActionScript code in Listing 3.

The first two lines of code are extremely important when developing Flash Remoting applications and components. These contain the necessary classes to communicate with ColdFusion and make calls to CFC methods. If Flash complains that these includes don't exist, then the Remoting components are not installed. By default these are included with the



Figure 1: The properties panel for the ScrollBar component

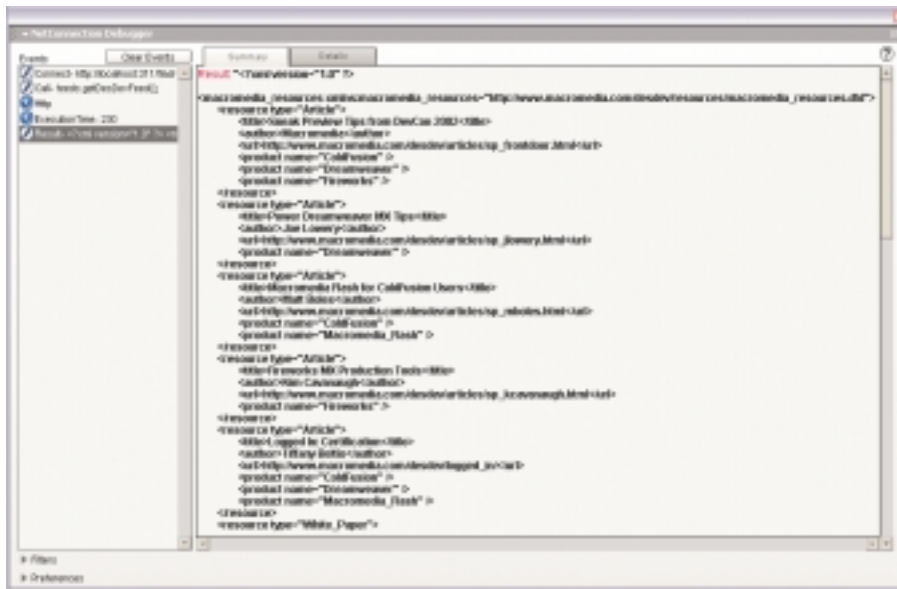


Figure 2: The NetConnection Debugger displaying the DesDev XML document

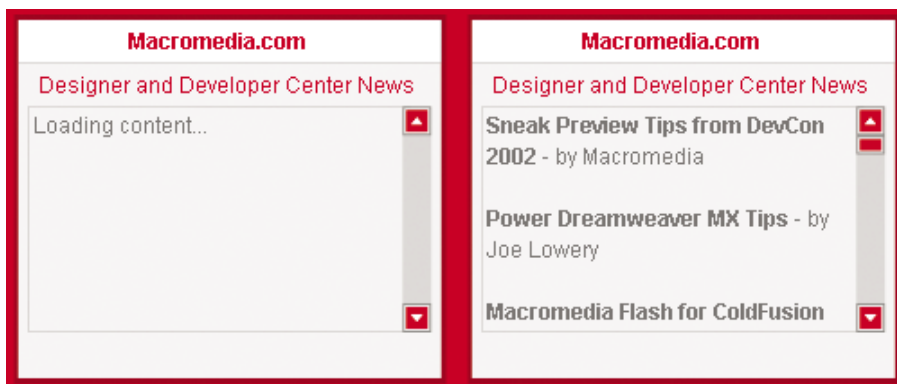


Figure 3: The finalized application being initialized and then displaying the DesDev feed

CFMX installation. If for some reason you don't have them, you can grab them from Macromedia's Web site: www.macromedia.com/software/flashremoting/downloads/components.

Also note that the second line of code (#include "NetDebug.as") is unnecessary in a production environment. This is only used for debugging purposes such as tracing properties, methods, and results on the client and server. All of this information is contained within the NetConnection debugger (Window > NetConnection Debugger).

For more information about using and understanding the NetConnection Debugger, see my article, "Get Connected with Flash Debugging" (CFDJ Volume 4, issue 10). Be sure to remove the debug include or comment it out when publishing your file to a production server. This will save about 6KB in file size, which isn't a lot, but every little bit helps.

Now let's get into initializing the application. You'll see that the init code is set to run only once, which is all we'll need to establish the connection with

the server. This is just a failsafe practice that is used to prevent any additional connections from being established with the server. Within this code we display a simple "Loading" message within our text field on the stage. We've given the text field an instance name of "body" so that we can make reference to it when sending information to be displayed in the field.

This instance reference is also used by the ScrollBar component, and is set as the target TextField for the component. This tells the ScrollBar which text field to control, and will automatically control the scrolling of any text within the field. To set the target TextField be sure to select the ScrollBar on the stage and view the properties panel (Window > Properties). Figure 1 shows the properties panel for the ScrollBar.

Next we create the "gwURL" variable and set it as the defaultGatewayURL. This is important when we're testing within the IDE so that Flash knows where to establish the Remoting connection. The last part of the initialization code is to create a reference to the service we're calling (cfdj.feeds) and then call the service method we created earlier (getDesDevFeed).

Now that the method has been called, we need a way to handle the result sent from the server. Remember, the getDesDevFeed method pulls the XML document from Macromedia's Web site and then returns it to the caller. In Flash we call this method the default responder, which takes the format of "function Name_result". In our case, the default responder is "getDesDevFeed_Result". The XML contents are passed into the default responder and we run a custom routine to parse the XML and display it in the "body" text field.

We basically create an XML object and make sure to remove any whitespace in the document. This is good practice since Flash will treat whitespace as empty nodes, which can ultimately complicate the parsing process. After the XML is loaded and parsed, we grab the number of resource nodes and enumerate through each node. This allows us to append the title, author, and URL to our text field. We use the "body.htmlText" property to allow the text field to accept HTML contents. Figure 2 shows the XML document passed from CF to Flash in the NetConnection Debugger.

If you haven't dealt with the Flash XML object in the past, it's good to note that XML parsing has increased dramatically in Flash MX. In Flash 5 the XML parser was fairly slow when parsing large XML documents. I would say that on average the Flash MX parser is 5-10 times faster than the Flash 5 version - call it XML parsing on steroids!

You should now be able to test your movie and see the results of your hard work (CTRL-Enter or Control > Test Movie). You should initially see the "Loading content..." message until the XML is parsed and loaded into the text field. You can then use the scroll bar to view the different articles and click the titles which link to the corresponding

content on Macromedia's Web site. Figure 3 shows the finished product.

Conclusion

As you can see, it's a fairly simple task to pull dynamic news into Flash using Remoting and CFCs. The power of this approach lies in the fact that this is very extensible and can be used in future components and applications. For instance, if the user wanted to access the feed component and display it in a CF page, this would be just as simple to do.

If you didn't notice, the ScrollBar was re-skinned to match the color scheme of the movie. Try to play around by extending the movie with changing colors, UI, and even a bit of animation. Stay tuned as we'll

cover this in the future as well as hook up Flash and CF with Web services. 

About the Author

Dennis Baldwin is the lead developer for Eternal Media, a company that provides information and technology solutions for nonprofit organizations and ministries. He's also the co-author of Reality ColdFusion: Flash MX Integration, a book that focuses on applying Flash Remoting to real-world applications. Dennis also helps maintain a couple of online resources for Flash and ColdFusion developers at www.devmx.com and www.flashcfm.com.

dennis@devmx.com

Listing 1: Macromedia DesDev XML Document Structure

```
<?xml version="1.0" ?>
<macromedia_resources>
<resource type="Article">
  <title>Sneak Preview Tips from DevCon 2002</title>
  <author>Macromedia</author>
<url>http://www.macromedia.com/desdev/articles/sp_frontdoor.html</url>
  <product name="ColdFusion" />
</resource>
<resource type="Article">
  <title>Power Dreamweaver MX Tips</title>
  <author>Joe Lowery</author>
<url>http://www.macromedia.com/desdev/articles/sp_jlowery.html</url>
  <product name="Dreamweaver" />
</resource>
</macromedia_resources>
```

Listing 2: ColdFusion Component that pulls the Macromedia DesDev feed locally

```
<cfcomponent displayname="feeds">
<cffunction name="getDesDevFeed" access="remote" returntype="any">
  <cfprocessingdirective suppressWhiteSpace="Yes">
    <cfhttp url="http://www.macromedia.com/desdev/resources/macromedia_resources.xml" method="get" redirect="yes" timeout="10"/>
    <cfheader name="Content-type" value="#cfhttp.mimeType#">
    <cfreturn cfhttp.filecontent />
  </cfprocessingdirective>
</cffunction>
</cfcomponent>
```

Listing 3: Flash ActionScript code to load and parse XML from the CFC

```
// include the necessary remoting classes
#include "NetServices.as"
#include "NetDebug.as"
// initialize the application only once
if(inited == null){
```

```
  inited = true;
  // display the loading message in the text field
  body.text = "Loading content...";
  // create the gateway url variable and set the defaultGatewayURL
  var gwURL = "http://localhost/flashservices/gateway";
  NetServices.setDefaultGatewayURL(gwURL);
  // establish the connection to the gateway
  gw = NetServices.createGatewayConnection();
  // make reference to the feeds service
  cfService = gw.getService("cfdj.feeds", this);
  // call the getDesDevFeed method of the service
  cfService.getDesDevFeed();
}
// handle the XML result passed from our CFC
function getDesDevFeed_Result(result) {
  // create the XML object
  var myXML = new XML();
  // strip any whitespace in the XML
  myXML.ignoreWhite = true;
  // parse the XML
  myXML.parseXML(result);
  // set the length so we can enumerate through the XML nodes
  var len = myXML.childNodes[0].childNodes.length;
  var pointer = myXML.childNodes[0].childNodes;
  // clear out the text field
  body.text = '';
  // loop through the xml nodes and populate the text field
  for(var i = 0; i < len; i++) {
    var title = pointer[i].firstChild.firstChild;
    var author = pointer[i].firstChild.nextSibling.firstChild;
    var url = pointer[i].firstChild.nextSibling.nextSibling.firstChild;
    body.htmlText += "<b><a href='" + url +
    "' target='_blank'>"+ title + "</a></b> - by " + author + "<br>";
  }
}
```

EDGE Web Hosting

www.edgewebhosting.net/cfdj

'But It's Free!'

Time to clear the air and set the record straight

There was once a time, not that long ago, when I seemed to be spending a significant portion of my life explaining why ColdFusion was indeed worthy of consideration even though other products were free. Then I stopped hearing that objection for the most part, and life was good (well, better).

But in the past two weeks I have had to defend ColdFusion against the “but [fill in the blank here] is free” objection no less than five times. So it looks like I need to address this issue one last time to clear the air and set the record straight.

What Exactly Is ColdFusion?

Let's start with the question – what is ColdFusion anyway? Is it an application server? Some think so, others vehemently disagree with that description. Is it a language? Definitely. But is it only a language? No, not at all. So, what is ColdFusion then?

There is actually a reason for me starting with this question. Because ColdFusion is not easily defined, it is not easily compared to other products and technologies. And this inability to perform comparisons usually results in one of two things happening – either ColdFusion gets compared to products that it honestly should not be compared to, or only one aspect of ColdFusion gets looked at and compared, but not ColdFusion as a whole.

So, once again, what exactly is ColdFusion? Actually, ColdFusion is many things; it really depends on your perspective, what you are trying to do, and how you are trying to do it. But at a



By Ben Forta

minimum:

- **First and foremost, ColdFusion is a language:**

After all, it is CFML (the ColdFusion Markup Language) that makes ColdFusion ColdFusion.

What we love about the product, what makes us productive, what distinguishes

ColdFusion from anything else out there is CFML.

- **ColdFusion is also a set of application services, complete subsystems that you can leverage in your applications – things like full-text searching, graphing, database query caching, and more:** These are not language features (although they have language elements associated with them); they are runtime services that are included with ColdFusion so that you do not have to start from scratch.
- **And yes, for many of us, ColdFusion is still an application server:** With ColdFusion MX, and its new ability to run on top of industry-standard application servers (like IBM WebSphere and Sun ONE), ColdFusion has been split in two. It can be used as a scripting engine with supporting runtime services on top of an application server of your choice, or it can be used as an application server in its own right.

All of which makes defining exactly what ColdFusion is a little tricky.

It's More Than a Language

As I already explained, part of ColdFusion is a series of runtime application services. These are included with ColdFusion in all versions (although there are some differences based on the product edition being used), and all at no extra cost. It has been this way almost since day one.

Why is this important? Well, we take many of these included services and technologies for granted. As ColdFusion we use them all the time, and never even realize that had we been using other popular languages, we'd have had to pay for these, buying them individually.

Now, I don't usually take potshots at other vendors or products (well, not often, and usually not in public and in print), but... consider Microsoft ASP, the most used scripting language out in Web-land. ASP is free, right? Maybe, but look at the chart on the next page.

The chart is not even a complete list of ColdFusion features missing from ASP, but it's enough for the discussion at hand.

Just to be clear, and in the interests of fairness, when a third party add-on was needed (because a feature is not included in ASP itself) the lowest-cost functionally equivalent add-on was used. Where multiple options were available, the one that most closely matched the feature set native to ColdFusion was used. However, unsupported free add-ons were not included.

Now before I get flooded with e-mails accusing me of rigging the data, yes, I am sure I could have searched and found cheaper options too. And yes,

Macromedia
www.macromedia.com/go/cfmxmlight

there is stuff out there for free. And yes, some of those add-ons might have features and options that ColdFusion does not support. Agreed. This is not official data, nor is it an apples-to-apples comparison. I did not put tons of time into this research, that was not the intent.

What this is, however, is proof positive that you get what you pay for. Or in other words, pay now or pay later. And if you do opt to pay later, you'll likely pay a lot more. (Plus the integration will not be as clean or as simple, it just can't be.)

This is not just true of ASP – it is true of every one of the free options. ColdFusion is not free, but from a raw cost of goods standpoint it is an absolute bargain – matching the features in any other development product will cost you a lot more than ColdFusion.

And It's an Incredible Language Too

From a cost perspective, just by looking at what you get in the box, ColdFusion is already a far better option. And we haven't even mentioned the cost associated with the language – or rather, the cost associated with using the wrong language.

CFML is not a general-purpose language; it was never intended to be one. But CFML is the very best language for

integrating with Internet clients and technologies. Take something as simple as a database query – in ColdFusion we use a single tag named `<CFQUERY>` and it just works. Have you ever looked at what it takes to do the same in ASP or PHP or even JSP? Or how about something as trivial as sending an SMTP e-mail message?

Or even a simple thing like making data persist in a shared scope? A colleague recently shared with me some ASP code that he was so proud of – code that helped him eliminate unnecessary database reads by caching results in the application itself. When he was finished with his lengthy and very detailed code walkthrough and explanation I showed him `<CFQUERY NAME="APPLICATION.result">` and `<CFQUERY CACHEDWITHIN ...>`. He just stared and said nothing (although I think he was trying to respond – his lips moved but no sound escaped).

At DevCon in October I discussed Web services in detail, and had the opportunity to demonstrate ColdFusion Web services being consumed by .NET (using C#) as well as .NET Web services being consumed by ColdFusion MX. One of the most blatantly obvious differences was the amount of code it took to consume a Web service – .NET is built


around Web services technologies, and ColdFusion puts it to shame with something as simple as consuming a Web service (a .NET Web service at that) – one tag compared to pages of code.

I am not going to go into when CFML should be used and when it shouldn't; I've written and talked about that many times before. But one thing is a given, an indisputable fact. For the right parts of your application, CFML is the cleanest, the simplest, the most productive, and simply the best solution available to you.

What does that have to do with other products being free? Not only do you get more product for your money as already explained, but by using CFML, your development costs drop dramatically. Using CFML, your development will be simple and will take less time, and time is money. It's as simple as that.

Summary

So, why buy ColdFusion when ASP (or JSP or PHP or Perl or...) is free? Well, if they were indeed free that would be a very good question. But, as you can see, they are not free, and thus it's not a good question at all. The truth is that the question should be exactly the opposite – with ColdFusion being such a bargain, offering such value for the money and ongoing savings in development costs, why would you ever want to use a supposedly free product?

That's not to say that no one should be using those products. I have found one scenario where they make perfect sense – if you are a contract developer getting paid by the hour, then you may benefit from legitimately having to take so much longer on your projects. But for everyone else, the choice is clear. Enough said. 

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

	ColdFusion Professional		ASP	
Product		\$799.95		\$0.00
Verity full text searching	Included at no extra cost	\$0.00	Verity K2 Bronze	\$31,000.00
Charting	Included at no extra cost	\$0.00	Not included, Dundas Chart	\$699.00
POP support	Included at no extra cost	\$0.00	Not included, AspPOP	\$69.95
Server-side HTTP support	Included at no extra cost	\$0.00	Not included, AspHTTP	\$49.95
Server-side FTP support	Included at no extra cost	\$0.00	Not included, Dart FTP Tool	\$249.00
File uploads	Included at no extra cost	\$0.00	Not included, AspUpload	\$149.00
COST		\$799.95		\$32,216.90

Macromedia

www.macromedia.com/go/usergroups

Serving Word

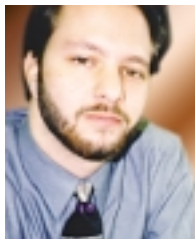
HTML, XML, and CSS make it easy

ColdFusion developers often include in their applications the functionality to serve up dynamic Microsoft Office documents. The most common technique for doing this is through automation – the process of controlling one application from another. While there have been numerous articles, sites, presentations, and examples of using Office automation with ColdFusion, it is wholly inappropriate without extreme precautions. Office is not safe for unattended execution; unanticipated content or programming errors can cause an Office application to pop up a prompt on the server or otherwise halt the process.

In this article I'll describe how to use technologies and techniques you're already familiar with to safely generate dynamic Word documents and serve them to clients over the Internet. To do this I'll use ColdFusion to generate MS Word-compatible HTML, XML, and CSS and instruct the browser to open the document in MS Word. This approach doesn't require any software on the server and works with both Netscape and Internet Explorer on the client side. Users can view the document in MS Word 2000 or 2002, if they have it installed, or save the file to disk.

Basic Word Document

While we're going to use the same HTML to generate Word documents that we've always used to display Web pages in the browser, there are fundamental differences in the implementations, as well as some additional syntax. The most basic difference to remember is that an MS Word document is a page-based medium geared primarily toward being printed, while Web pages are



By Samuel Neff

screen-based and designed to be viewed in flowing text on the screen. Additionally, MS Word documents are divided into sections, and each section can have its own page-formatting settings.

All of the examples referred to in this article are available online at www.sys-con.com/coldfusion/sourcecec.cfm.

To start with, look at the HelloWorld.cfm example. It contains only basic HTML elements and two cfheader tags. When run in a browser, the HTML text is sent to MS Word and displayed to the user. Notice that the HTML is being processed by only MS Word, not by the browser, which relieves us of cross-browser compatibility concerns. The only involvement that the browser has in this particular request is as a conduit, accepting the content from the Web server and sending it directly to MS Word.

The Hello Word example is very small and simply demonstrates the technique used to send HTML documents to MS Word. To demonstrate some of the Word-specific syntax, let's review Letter.cfm.

The first new item is on the root HTML tag. In order to use any of the MS Word XML grammar, we need to add the namespace declaration `xmlns:w="urn:schemas-microsoft-com:office:word"` to the HTML element. With this declaration we can set Word-specific options such as the initial view and optimizations, as shown within the XML element on lines 14–24.

In lines 31–37 we use the standard CSS @page directive to declare the size, layout, and margins for the pages in each section. By declaring the size as "8.5in 11in" we're creating a standard size document in portrait layout; reversing the numbers would create a landscape layout. The margins are set as "1.0in 1.25in 1.0in 1.25in"; the numbers correspond to the top, right, bottom, and left margins, respectively. After defining a page setup, we declare a div class, which corresponds to this page and will be used later to define the sections of our document.

With the page styles declared, we'll turn to paragraph formatting in lines 53–61. We'll specify that the paragraphs should not have any margin, which eliminates the extra space exhibited in most Web pages, and that we want to turn on orphan control through the Microsoft-specific `mso-pagination: window-orphan` style. Orphan control keeps a paragraph from breaking across a page such that only one line is left on a page. We also declare a style for indented paragraphs that have a three-inch left margin.

Within the body we use a div tag to associate the text with the section defined above. The text of the letter is included within paragraph tags, with the first three using the indented style. Notice that empty paragraphs must contain the ` ` entity or they will not be rendered. Additionally, we'll instruct

Word not to collapse the double space after the end of each paragraph, using the Microsoft specific mso-spacerun: yes style.

Finally, we use the same two cfheader tags to declare that the document should be sent to MS Word as an attachment.

Labels

Labels require exact precision and formatting that can't be achieved with traditional HTML displayed in a browser. Generating labels with MS Word is easily accomplished through standard HTML tables, with care taken regarding column and row sizing, cell and paragraph margins, and a few MS Office-specific styles.

In Labels01.cfm we generate dynamic content compatible with Avery 5160 address labels. Lines 9–15 set up the page dimensions and lines 16–45 set up table, row, cell, and paragraph spacing. All items are set to the exact specifications required for the target label. This information is gathered by creating a labels document through the MS Word Labels wizard and then saving as HTML. Once the

HTML document is created, the label specifications can be extracted for creating dynamic labels.

The rest of the document queries the database for addresses and creates a standard table for the labels. Each row contains three labels and two spacer cells. After the loop, the template adds extra cells as necessary to fill out the row. Finally, it uses the same cfheader tags to instruct the browser to open the document in MS Word.

When creating labels, it's useful to separate the code to generate the labels themselves from the label content. This makes your code maintainable and allows easy switching to a different label style.

When writing reusable code to generate any style label, one issue that developers is that labels have different content – some labels are address labels on which you'll want to place a full address, and others may just be file folder labels for names or diskette labels for product information.

ColdFusion MX simplifies this code reuse through components. By creating a base component that contains the logic and information to create labels,

and extending that component for each type of content, you can maintain a minimal set of templates to produce many different labels. To demonstrate this, first look at WordLabelGenerator-Base.cfc.

In lines 22–124 we include code within the cfcomponent tag but outside any cffunction tags, which serves to initialize the component and create a structure containing the label layout details for a variety of Avery labels. This data was all obtained through MS Word's Save as Web Page feature. In real-world use, it would be preferable to store this information outside the component in a more maintainable mechanism, either a database or an XML file.

Lines 141–426 define the createLabels function, which is the main worker function for the component. It accepts the label number, query, and some font information as arguments. The logic within createLabels allows for labels that require spacer cells, spacer rows, or both. It uses cfsavecontent to generate HTML code very similar to the previous example, but instead of outputting the generated content it returns it to the caller.

CFDYNAMICS

www.cfdynamics.com

Notice on line 196 that we take the query passed to our function and store it in a component-level variable so it is available to other functions within the component. Component-level data in CFMX components is stored in the Variables scope, but due to a bug that confuses this data with the Caller scope, we created a structure called `private` within the component constructor and use this to contain all component-level private data. This improves readability and allows us to store arguments as component-level data without changing the variable name.

The next line of interest is 363. This is where we output the actual label content. Since this is a base component that can be used to create a variety of labels containing different content, we call the component function `createLabelContent` to actually output the content. The function accepts a single parameter, which is the index into the query for the current record.

The function `createLabelContent` is declared in lines 437–458. The base component `WordLabelGeneratorBase` is not intended to be used directly in any application but is instead intended to be extended for each type of content generated. This is similar to an abstract class in object-oriented languages. For this reason, the `createLabelContent` function in the base component simply throws an error.

With the base component created, we can extend it to create labels with different content types, such as an address or product information. For these components review `WordLabelGeneratorAddress.cfc` and `WordLabelGeneratorDiskette.cfc`. Both components extend `WordLabelGeneratorBase` and contain only the `createLabelContent` function. Notice that this function is private and is used only by the `createLabels` function inherited from the base component. The address component formats the query data as a three-line address, while the diskette component formats product information.

In examples `Labels02.cfm` and `Labels03.cfm`, we utilize our components to easily create labels from the information stored in our database. These two files create addresses and diskettes, respectively. Each template

starts with a URL parameter for the label style, then queries the database and invokes the appropriate component. Finally, they output the content returned from the component and declare to the browser that the document should be sent to MS Word as an attachment.

Readers with ColdFusion 5 can utilize this same functionality by replacing the base component with a custom tag. The variable content can be achieved by creating a user-defined function for content formatting and passing the function as a parameter to the custom tag.

Multifile Documents

Some features of MS Word documents require multiple files to render in HTML. These include headers, footers, and embedded images. To send multiple files to the browser in a single request they must be packed inside a Multipart-MIME message, which Microsoft refers to as a Web Archive. Users with Word 2000 will need to download and install the Web Archive add-in for Office 2000, available from Microsoft's Office Web site, <http://office.microsoft.com>.

Review `MultipartMIME_Example.txt` for an example of a Multipart-MIME message. It begins with the MIME version declaration and then declares the content type as `multipart/related` and a boundary, lines one and two. The boundary is any unique sequence of characters pre-pended with two minus signs.

Lines 4–8 are the message header. These are ignored when processed by a Multipart-MIME-aware agent such as MS Word and are used to display a message to the user in noncompliant agents.

Line 10 is the first boundary and indicates the start of the first message part. Lines 11–13 declare the location, type, and encoding. The location is not a physical location but a named location, which can serve several purposes. The original intent of the content location header was to indicate to agents when to use cached versions, but MS Word uses this header to name the individual sections. The name consists of a file-based URL but is never used as the

actual physical location, only for the relative location. It's important to include a very random set of characters as the root of all locations so the documents are properly processed by MS Word.

The first part is the main HTML document and is encoded using the Quoted-Printable standard. Quoted-Printable is a means of encoding text where nonsafe characters are encoded with an equals sign and two-digit Hex code, and soft line breaks (for lines exceeding 75 characters) denoted by an equals sign.

Line 289 is another boundary and indicates the start of the second message part, which consists of an XML document also encoded using the Quoted-Printable standard. The XML document is a file list that follows a simple format consisting of an XML element, a `MainFile`, and `File` elements. The list identifies the main file as well as all related files. The exact relation between the files is declared within the content, explained later.

The boundary at line 300 starts the third message part, which is a binary image encoded using Base64. Base64 encoding maps three bytes of binary data to four safe characters and is easily accomplished in CF through the `ToBase64` function. The data is slightly modified from the raw `ToBase64` output by adding a line break after each 76th character.

The message ends with a final boundary including an extra two minus signs.

Creating Multipart-MIME Messages in ColdFusion MX

To create Multipart-MIME messages we'll use a ColdFusion Component, `MultipartMIME.cfc`. Review `Columns03.cfm` to see how to create a document that contains a header, footer, and embedded image. `Columns03.cfm` is a small template that loads the individual message parts into variables and passes them to the `MultipartMIME` component to construct the message. Then it sends the message to the browser and instructs it to open in MS Word.

`Columns03.cfm` uses four support files to create the final multipart docu-

ment. The first file is Columns03_MainDoc.htm. While this is similar to the earlier examples, notice the link element on line 9. This link is used to associate the main document with the file list.

Lines 40 and 41 associate headers and footers with the document using MS Office-specific styles added to the page declaration. Notice that both the header and footer are contained in the same document and are identified by the ID attribute of div tags, in this case h1 and f1.


On line 116 we include an image by adding a standard img tag. The image itself is included in the Multipart-MIME message and the file list.

Columns03_header.htm contains both the header and footer. At the top is a link reference back to the main file. Within the body are div tags for each header and footer. The ID attribute on the div corresponds to the ID used in the main document's style settings. Finally, the div tags should also have a style attribute

identifying the section as a header or footer.

Columns03_FileList.xml is the file list for our multipart document. It contains a line for the main document and each related file.

Summary

Using HTML, XML, and CSS is much faster and more stable than using COM for creating Microsoft Word documents. It's also easy to do and builds upon developers' existing skill sets. This technique can be used to create virtually any MS Word document and should be utilized whenever application specifications call for this functionality. 

About the Author

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

sam@blinux.com

Errata



Dear Readers,
In my October article, "Precompiling CFML Templates in CFMX," I offered a precompile.bat file that was missing the critical first portion of its second line. The correct file is offered below:

```
set MX_INSTALL=d:\cfusionMX
MX_INSTALL%\runtime\jre\bin\java -classpath
MX_INSTALL%\lib\cfusion.jar
coldfusion.tools.Compiler -webroot %1
-webinf %MX_INSTALL%\wwwroot\WEB-INF %1
```

As indicated in the article, that's just two lines of code. The first is just "set MX_INSTALL=d:\cfusionMX" and the rest is all one line.

Charlie

HostMySite.com

www.hostmysite.com

Extending ColdFusion

Dealing with users

Welcome to another installment of Extending ColdFusion. This month I'm going to talk about a UDF (user-defined function) that will help deal with users. Users are wonderful. Without people visiting your site, your Web page may as well not exist. On the other hand, users have a way of doing the most stupid... err crazy things that you can (or most likely cannot) imagine.

You should always assume that your site visitors will not do what you intend; in other words, you should build your Web site to handle every situation – a goal that is not easy to achieve. One of the (many) ways that users can “do the wrong thing” is when you give them a chance to enter information. Almost all Web sites allow users to enter information via forms. However, as soon as you let the user enter any kind of information, you need to make sure the information isn't something that will throw your site into a tailspin.

Consider forums. Many Web sites have a forum of some type that allows users to have threaded discussions. When posting to the forum, the Web application may ask for the user's name, the subject of his or her post, and then the body of the message. This information is then stored in a database and displayed on the Web page. However, what would happen if the user entered this for a name?

```
Jacob <b>Camden</b>
```

Instead of just entering his name, the user entered HTML as well. In this case, the user wrapped his last name with bold tags. This isn't horrible, but when the forum posting is viewed, his name

By Raymond Camden

will not appear in the same manner in which other forum postings appear. Next, consider this input:

```
Jacob <b>Camden
```

This time, the user entered incorrect HTML. When the Web page is viewed, the layout could be totally off from what it should be. If the user had entered a `<table>` tag instead, the entire page content may actually disappear in Netscape. In this case, the user just made a stupid mistake, but consider this input:

```
Jacob Camden<script>document.location.href=
"http://www.whitehouse.gov";</script>
```

This time the input will do something far worse. When the page is viewed, it will execute JavaScript code that will send visitors of your Web site to the official Web page of the U.S. president. However, if the “.gov” was switched to “.com”, they would be sent to a porn site instead. Visitors may actually think that your site was the one hosting the porn.

So, how do we fix this? ColdFusion provides us with two functions that can easily and quickly “clean” HTML code input: `HTMLCodeFormat` and `HTMLEditFormat`. The only problem

with these two functions is that they will remove all code. What if you want to allow some HTML like the bold or italics tags? The first UDF we will look at will allow you to specify the tags you don't want, and thereby allow other tags that are safe for your Web site.

The UDF, `SafeText` (www.cflib.org/udf.cfm?ID=56), was written by Nathan Dintenfass (who was featured in my last article as well; see *CFDJ* Vol. 4, issue 10). The UDF takes one required argument, text (the text to be cleaned), and three optional arguments. You can specify if bad code should be escaped or removed (defaults to being removed). You can specify a list of bad tags, although the default list is intensive. Finally, you can create a list of bad “events”. This allows you to remove JavaScript events like `onClick` or `onMouseOver`. Listing 1 contains the UDF.

The UDF begins with a set of var statements. These are variables that exist only in the UDF itself. The important ones are `badTags` and `badEvents`. These values are the defaults that define which tags and JavaScript events to strip from the string. After the var statements, we have a code branch that splits between replacing the bad tags or simply removing them. If we need to replace them, a while loop is used to search over the text. If bad tags should simply be removed, one regular expression call is used.

Why is it easier to just remove them? Replacing means calling `htmlEditFormat` on the particular tag that was matched. If this could be done with a simple regex, please e-mail the author (and me!). However, you can completely remove the bad tags with one call. Next, a regular expression call is used to remove the bad JavaScript events. Last, the cleaned text is removed. Listing 2 contains a simple example. I use `<cfsavecontent>` to store

text into a variable. Notice how I use some good HTML, along with some bad HTML and JavaScript events. The good HTML will remain; the bad will be stripped out.

That's it for this installment. In the next one we'll look at another UDF that we can use to deal with user input. In the meantime, if you have any ideas for UDFs, custom tags, or CFCs that you would like to

see discussed, please send them to me at ray@camdenfamily.com.



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). ray@camdenfamily.com

Listing 1

```
<cfscript>
/**
 * Removes potentially nasty HTML text.
 * @param text      String to be modified.
 * @param strip      Boolean value (defaults to false) that deter-
 *                   mines if HTML should be stripped or just escaped out.
 * @param badTags     A list of bad tags. Has a long default list.
 *                   Consult source.
 * @param badEvents   A list of bad HTML events. Has a long
 *                   default list. Consult source.
 * @return Returns a string.
 * @author Nathan Dintenfass (nathan@changemedia.com)
 * @version 1, July 17, 2001
 */
function safetext(text) {
    //default mode is "escape"
    var mode = "escape";
    //the things to strip out (badTags are HTML tags to strip and
    badEvents are intra-tag stuff to kill)
    //you can change this list to suit your needs
    var badTags = "SCRIPT,OBJECT,APPLET,EMBED,FORM,LAYER,ILAYER,FRAME,
    IFRAME,FRAMESET,PARAM,META";
    var badEvents = "onClick,onDblClick,onKeyDown,onKeyPress,onKeyUp,
    onMouseDown,onMouseOut,onMouseUp,onMouseOver,onBlur,onChange,
    onFocus,onSelect,javascript:";
    var stripperRE = "</?(" & ListChangeDelims(badTags,"|") &
    ")[^>]*>";
    //set up variable to parse and while we're at it trim white space
    and deal with "smart quotes" from MS Word, etc.
    var theText = trim(REReplace(text,"(f|&)", "'", "ALL"));
    //find the first open bracket to start parsing
    var o Bracket = find("<",theText);
    //var for badTag
    var badTag = "";
    //var for the next start in the parse loop
    var nextStart = "";
    //if there is more than one argument and the second argument is
    boolean TRUE, we are stripping
    if(arraylen(arguments) GT 1 AND arguments[2]) mode = "strip";
    if(arraylen(arguments) gt 2 and len(arguments[3])) badTags = argu-
    ments[3];
    if(arraylen(arguments) gt 2 and len(arguments[4])) badEvents =
    arguments[4];

    //if escaping, run through the code bracket by bracket and escape
    the bad tags.
    if(mode is "escape"){
```

```
//go until no more open brackets to find
while(o Bracket){
    //find the next instance of one of the bad tags
    badTag = RERFindNoCase(stripperRE,theText,o Bracket,1);
    //if a bad tag is found, escape it
    if(badTag.pos[1]){
        theText = Replace(theText,Mid
        (TheText,badTag.pos[1],badTag.
        len[1]),HTMLEditFormat(Mid(TheText,
        badTag.pos[1],badTag.len[1]),"ALL");
        nextStart = badTag.pos[1] +
        badTag.len[1];
    }
    //if no bad tag is found, move on
    else{
        nextStart = o Bracket + 1;
    }
    //find the next open bracket
    o Bracket = find("<",theText,nextStart);
}
//if not escaping, assume stripping
else{
    theText = REReplaceNoCase(TheText,StripperRE,"","ALL");
}
//now kill the bad "events" (intra tag text)
theText = REReplace(theText,(ListChangeDelims(badEvents,"|")),
"","ALL");
//return theText
return theText;
}
</cfscript>
```

Listing 2

```
<!-- include the udf -->
<cfinclude template="listing1.cfm">

<cfsavecontent variable="test">
This text will contain both <b>bad</b> and <i>good</i> html. The
<form> bad html will be stripped out. We will
also strip out any <a href="http://www.macromedia.com" onClick="docu-
ment.location.href='http://www.whitehouse.gov';">bad</a>
JavaScript events.
</cfsavecontent>

<cfset cleaned = safeText(test,true)>

<cfoutput>#cleaned#</cfoutput>
```


SUBSCRIBE TODAY TO MULTIPACK

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

- ☐ Web Services Resource CD
- ☐ Java Resource CD
- ☐ WebLogic Resource CD
- ☐ ColdFusion Resource CD
- ☐ XML Resource CD
- ☐ WebSphere Resource CD
- ☐ CF Advisor Complete Works CD

Your order will be processed the same day!



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 *JDJ* articles compiled on one CD! Assembled by *JDJ* 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



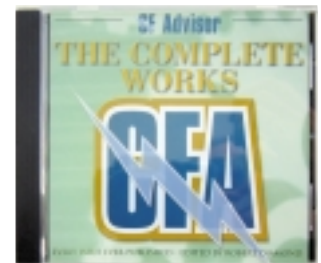
The most complete library of exclusive *CFDJ* articles on one CD! This CD, edited by *CFDJ* editor-in-chief Robert Diamond, is organized into more than 30 chapters with more than 400 exclusive articles on CF applications, custom tags, database, e-commerce, Spectra, enterprise CF, error handling, WDDX...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



The complete works of *CF Advisor* are now on one CD! Check out over 200 exclusive articles on topics such as e-commerce, custom tags, Fusebox, databases, object-oriented CF, WDS, upgrading CF, wireless, Verity, and more. Plus, find interviews, editorials, and source code!
LIST PRICE \$198

Subscribe Online Today

LE MAGAZINES ONLINE

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

Pick a
3-Pack, a
6-Pack or a
9-Pack ☒



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 US, Canada/Mexico or International. Then choose which magazines you want to include with your Multi-Pack order.

☐ **3-Pack** ☐ 1YR ☐ 2YR ☐ US ☐ Can/Mex ☐ Intl.
☐ **6-Pack** ☐ 1YR ☐ 2YR ☐ US ☐ Can/Mex ☐ Intl.
☐ **9-Pack** ☐ 1YR ☐ 2YR ☐ US ☐ Can/Mex ☐ Intl.

☐ 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) \$180 You Pay: \$149 / Save: \$31
 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
- Can/Mex - \$245.00
- All Other Cnty. - \$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
- Can/Mex - \$555.00
- All Other Cnty. - \$710.00

9-Pack

Pick all 9 of our
magazines and save
up to \$400⁰⁰
Pay only \$400 for a
1 year subscription
plus 3 FREE CDs

- 2 Year - \$839.00
- Can/Mex - \$695.00
- All Other Cnty. - \$890.00

DevCon 2002

Face to face with the CF Community

Another ColdFusion Developers Conference has officially come and gone. All in all, this year's conference, held October 27-30 in Orlando, Florida, was a huge success, both in terms of attendance (well over 2,000) and content. I'll begin with my personal view of the conference and then share some thoughts from CFDJ List members I had a chance to talk with.

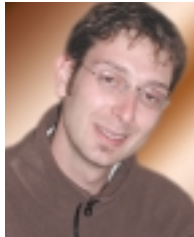
My first observation was that this year's conference was all about the MX suite – not just ColdFusion but the entire gamut of MX products. Past DevCons were very much more CF-centric: discussing the ColdFusion Server, CFML, CF Studio, and even integrating CF with other technologies.

While this year's conference offered little for designers, Macromedia did a good job familiarizing attendees with their many new products – all of which integrate very well with each other, and result in an increase in efficiency and creativity in finding solutions to business requirements.

More specifically, Dreamweaver seemed to show up everywhere, Flash Remoting and the Flash Communication Server were highly visible, and one (very good) morning's special event highlighted the new version of Director and its integration with Flash. On the CF side, ColdFusion Components, Flash Remoting, and Web services were definitely what most of the talk was about, though there were sessions that examined most of the new features in the CFML language.

To me, the most important thing at DevCon was that the direction in which Macromedia is taking the MX family of products was clearly defined, and I was very pleased to see the product integration and future-direction features being implemented first-hand.

Another interesting session at DevCon



By Simon Horwith

was Spectra's birds-of-a-feather discussion. The future of the Spectra source code seems to be up in the air. As of the end of 2003 at latest, it may become open source or it may become a product that is open source for a cost. Nobody seems quite sure what the future holds for Spectra and there certainly were many CMS solution

providers in attendance, eager to fill the void that this may create – most notably PaperThin Software (CommonSpot Content Management Server).

Thomas Harris, one of our CFDJ List members, said he was very impressed with all of the information regarding the use of CFCs to tie together ColdFusion, ASP.NET, and Flash (via remoting). He was a little disappointed that seating was so limited for the hands-on sessions, particularly the CFC hands-on (for which I was the slated presenter, and I just want to take a second to apologize for not being at my own presentation due to a family emergency; I will, however, be a presenter at CF-Europe in London in May). As for the hands-on sessions being full – as a Certified Instructor I can tell you that most instructors wouldn't feel comfortable with more students than the rooms were equipped to hold.

Tyson Kelly, another List member, was happy about the large number of Flash presentations. I agree with Tyson – this year's DevCon offered a lot in terms of Flash hands-on and general sessions. I hope many of you took advantage of these workshops.

Candace Cottrell, a frequent list contributor, enjoyed the new products preview, with the CFC information and Flash for CF Developers session, and the amount of information she was able to receive about Dreamweaver dos and don'ts. Without a doubt – there was plenty of opportunity to improve Dreamweaver skills at DevCon (something all of us needed!!).

Glenn Cross notes that for him, the best part of DevCon was the peer interaction. I couldn't agree more. I look forward to DevCon every year, most of all, because it gives me the opportunity to meet with all of you – the **CFDJ** readers and list members. It's great to spend all year exchanging e-mail and then, for four days, meet face to face and talk (and drink) with the other members of the CF Community.

CFDJ List member Max Hamby summed it up. He felt that the amount of knowledge he walked away with far exceeded his expectations. He especially liked the CFC sessions, the SQL Server 2000, and the XML presentations.

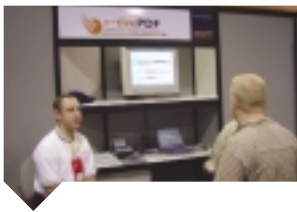
Finally, I took advantage of the opportunity at DevCon to speak with Sarge (of Macromedia Tech Support fame) about the changing role of tech support at Macromedia. He shed light on many topics that I think the community should be made aware of, so look for an article from me on this in the near future.

In the meantime, on the following pages is a look at what DevCon exhibitors offered attendees this year.

About the Author

Simon Horwith, a senior developer and Macromedia-certified ColdFusion instructor at Fig Leaf Software in Washington, DC, has been using ColdFusion since version 1.5. He is a contributing author to Professional ColdFusion 5.0 (WROX) as well as technical editor of The ColdFusion 5.0 Certification Study Guide (Syngress).

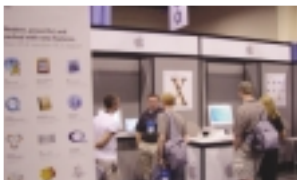
shorwith@figleaf.com



activePDF

www.activepdf.com

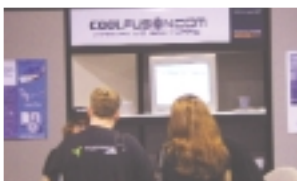
activePDF is a leading provider of server-side PDF development and conversion tools for Windows applications. Users can dynamically convert over 280 document types to PDF, including HTML, with no individual user licensing fees. Whether you need to create or convert to PDF, stamp, stitch, merge, paint, or form fill PDFs, activePDF has the answer.



Apple

www.apple.com

Apple ignited the personal computer revolution in the 1970s with the Apple II and reinvented the personal computer in the 1980s with the Macintosh. Apple is committed to bringing the best personal computing experience to students, educators, creative professionals, and consumers around the world through its innovative hardware, software, and Internet offerings.



Coolfusion.com

www.coolfusion.com

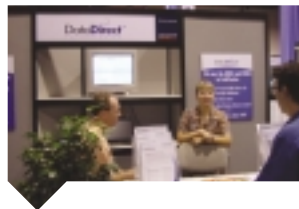
Coolfusion.com is a leader in application server integration. Their award-winning InFusion Mail Server (iMS) is the world's most configurable e-mail server. iMS utilizes ColdFusion templates for all processing intelligence, offering: dynamic e-mail processing, rapid application development, open integration, and an unlimited featureset.



CyberSage Software

www.cybersage.com

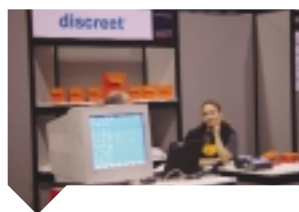
CyberSage Software specializes in building professional Java and XML-based software products for developers and graphic artists who use SOAP, Macromedia Flash, Java, and XML. CyberSage products solve real-world problems with such current releases as Firefly for Flash and FlashDoc, as well as other innovative products now in development.



DataDirect Technologies

www.datadirect-technologies.com

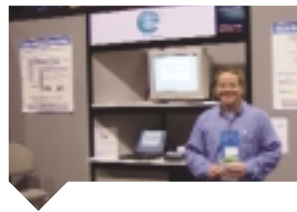
DataDirect Technologies is the leading provider of components for connecting software to data, providing standards-based technology that ensures consistent behavior and performance across diverse environments such as the Internet, J2EE, Microsoft .NET, and client/server. With the most comprehensive support for ODBC, JDBC, ADO, and XML, DataDirect Technologies makes it easy to connect software to data.



Discreet

www.discreet.com

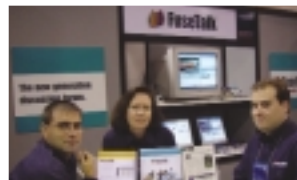
Plasma is the flagship 3D software for Web design from Discreet. Uniting proven 3D technology and unparalleled integration with Macromedia Flash and Director, plasma allows you to model, animate, and export directly to Macromedia Flash or Director. Plasma provides full control over animated output with modeling flexibility, as well as multiple camera and rendering options.



Ektron

www.ektron.com

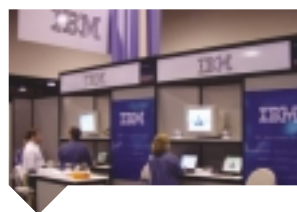
Ektron is a leader in browser-based Web content authoring and management solutions. Ektron software is affordable, is easy to deploy, and enables business users to easily update and maintain Web content – without knowing HTML or XML – while Web developers maintain control of the site. More than 500,000 end users in thousands of organizations worldwide use its award-winning products.



FuseTalk

www.fusetalk.com

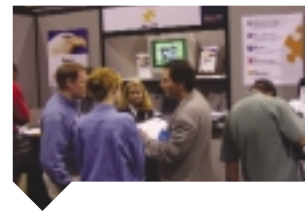
FuseTalk is the world's leading discussion forum solution for ColdFusion, delivering a robust environment for building online communities. The new FuseTalk Community Edition for ColdFusion MX provides two-tier administration and exciting new collaboration modules, including streaming n-way video, audio, chat, and whiteboarding.



IBM WebSphere

www.ibm.com/websphere

IBM WebSphere is the market-leading Internet infrastructure software (or middleware) for creating, running, and integrating e-business applications across a variety of computing platforms. Built on open standards such as J2EE, XML, and the new Web services standards, WebSphere server software and development tools are used by tens of thousands of customers.



Integration New Media

www.integrationnewmedia.com

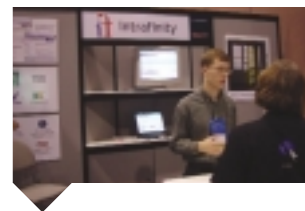
Integration New Media (INM) specializes in the development of cross-platform software components for rich media applications delivered locally and on the Internet. INM offers a range of products, solutions, services, and superior customer support. INM products include the V12 Database Engine, PDF Xtra, SecureNet Xtra, and GoldenGate Database Connector.



Intel

www.intel.com

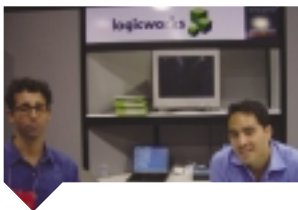
Intel is working closely with Macromedia to shape the next wave of Internet application experiences for developers, businesses, and end users. Through technology sharing and optimizations, Intel and Macromedia have improved the speed, reliability, and functionality of a full range of Macromedia products.



Intrafinity

www.intrafinity.com

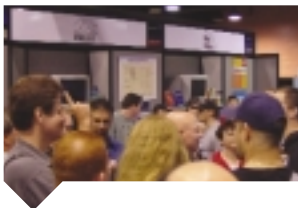
Intrafinity is the leading provider of knowledge management products and services based in ColdFusion. Their products include the Intrafinity Click Portal Server, the corporate portal that provides relevant information, personalization, and Web service delivery; and Intrafinity eWorkspace Server, the server that provides a document management system for employees to effectively collaborate on projects and initiatives.



Logicworks

www.logicworks.net

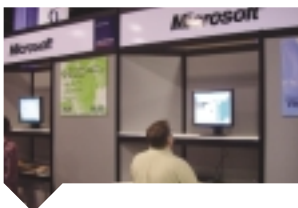
Founded in 1993, Logicworks is a solid, growing leader in managed dedicated server hosting, the hassle-free, low-cost alternative to colocation or running servers in-house. With its personalized 24x7x365 customer service, Logicworks customizes solutions to your exact needs with its excellent technical expertise and world-class facilities.



Macromedia Press

www.peachpit.com

Macromedia Press is the official publisher for Macromedia, offering expert training through top-notch books for both designers and developers. Choose from step-by-step project-based tutorials in the *Training from the Source* series, comprehensive reference guides in the *Macromedia Demystified* series, start-to-finish development projects in the *Reality* series, Certification Study Guides, developer Construction Kits, or task-based Visual QuickStart Guides.

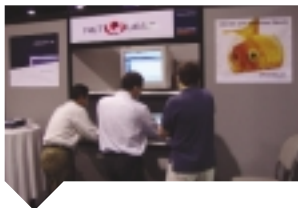


Microsoft

www.microsoft.com

Microsoft .NET is software that connects people, information, systems, and devices. A key component of .NET is a new Web development plat-

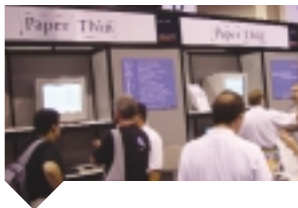
form called ASP.NET. With ASP.NET, developers can build high-performance Web and mobile Web applications, faster than ever before, using tools they're already familiar with. Several tools, including Macromedia Dreamweaver MX, Visual Studio .NET, and ASP.NET Web Matrix, now offer support for designing and developing ASP.NET applications.



NetQuest

www.nqcontent.com

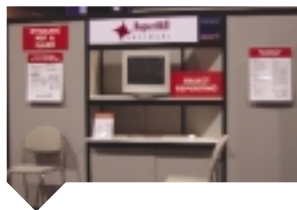
NetQuest offers NQcontent, an award-winning, ColdFusion-based Web content management solution. With a proven track record as the development platform of a wide spectrum of Web projects and intranets, NQcontent demonstrates its ability to cope with the most demanding customization needs. NQcontent is the favorite choice of Web developers looking for an affordable, powerful, yet easy-to-learn Web content management solution.



PaperThin

www.paperthin.com

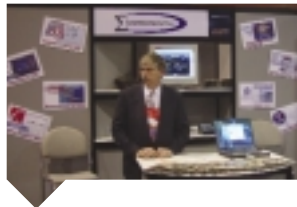
PaperThin offers CommonSpot Content Server, the premier Web publishing and content management solution. Based in Macromedia ColdFusion, the CommonSpot Server enables organizations to disseminate the creation, management, and delivery of content – all within a controlled, out-of-the-box framework. The open and extensible CommonSpot architecture allows for the easy customization of features as well as the seamless integration of ColdFusion applications.



ReportMill

www.reportmill.com

ReportMill is a Java developer tool for dynamically generating Web pages and reports from Java applications in Macromedia Flash and PDF. ReportMill combines an easy-to-use page layout application for template design and a powerful pure Java developer framework to generate content on the fly for rich Internet applications.



SIGMAinteractive

www.sigmatech.com

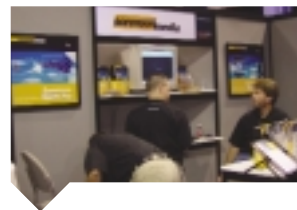
SIGMAinteractive is a premier user of Macromedia products developing high-level, custom, asynchronous learner-led Web-based and CD-ROM – based training. The company incorporates Virtual Reality, simulations, and 3-D graphics in their development. SIGMAinteractive has developed a tool that allows the customer to stay actively involved in the development process, and eliminates errors.



Site Executive

www.systemsalliance.com

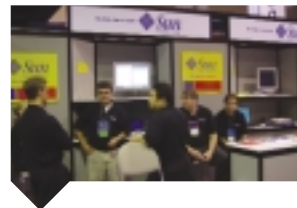
Site Executive, a product of Systems Alliance, is a dynamic content management system designed for mid- to enterprise-level organizations that allows content creators to post to the Web themselves – immediately. Site Executive automatically generates static pages with metadata options for each page, allows custom module creation, and provides outstanding extensibility.



Sorenson Media

www.sorenson.com

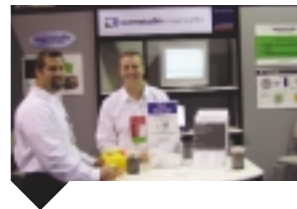
Sorenson Media provides simple and affordable solutions that enable people everywhere to easily compress and deliver quality video over the Internet. With its innovative tools and services that automate video compression and streaming, Sorenson Media is in a unique position to drive mass-market adoption of rich-media delivery over the Internet.



Sun Microsystems

www.sun.com

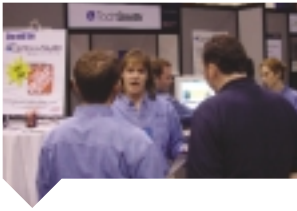
Since its inception in 1982, the Sun Microsystems singular vision – “The Network Is The Computer” – has propelled the company to its position as a leading provider of industrial-strength hardware, software, and services that “make the Net work.”



Teamstudio

www.teamstudio.com

Founded in 1995, Teamstudio develops and markets award-winning, agile software tools that enhance developer productivity and improve application quality. The company's Teamstudio Screensurfer is a Web-to-host integration tool and IDE that allows Web developers to effortlessly integrate legacy systems into existing Web applications, enabling organizations to leverage their IT investments.



TechSmith

www.techsmith.com

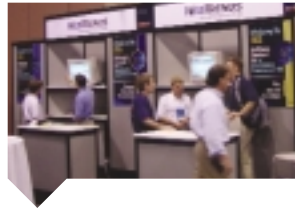
TechSmith is the maker of Camtasia Studio, which makes any software look simple by capturing all on-screen activity within a video that can be enhanced and delivered in all popular multimedia formats (Macromedia Flash, AVI, RM, MOV, WMV). Camtasia Studio combines the power of five world-class applications for the real-time creation of powerful support, marketing, and training material.



TrueSpectra

www.truespectra.com

TrueSpectra Image Server eliminates manual resizing, cropping, and storing of images by expanding Web server capabilities, enabling images to be modified on the fly using a single URL query string. Image modifications can include text overlays, visual watermarks, drop shadows, and borders – as well as zoom and pan functionality – using either DHTML or Macromedia Flash MX interfaces.



WebTrends

www.netiq.com

WebTrends, the Web analytics from NetIQ, offers comprehensive solutions to help businesses report, analyze, and act upon data generated by Web site visitors. The award-winning WebTrends software and eService solutions provide insight into user behavior and preferences that translate into higher returns on marketing and Web investments, as well as improved visitor-to-customer conversion rates.



Webvertising

www.webvertising.com

Founded in 1994, Webvertising provides Web application and rich Internet application development services for clients in a variety of industries. The company has created innovative products such as its OneScreen hotel reservation system for iHotelier, first used by the Broadmoor.

“Macromedia did a good job familiarizing attendees with their many new products – all of which integrate very well with each other, and result in an increase in efficiency and creativity in finding solutions to business requirements”

PaperThin

www.paperthin.com

All About Arrays

They're not too geeky

With all of the excitement over the new features of ColdFusion MX, it's easy to skip over some of the basics of ColdFusion.

In training developers, I find that the subject of arrays causes a good deal of confusion – so much so that many good coders omit arrays from their code altogether. One friend of mine simply refuses to use them. “They’re too geeky,” he complains. “Besides, in five years of ColdFusion development, I’ve always found a way around them.”

I think that my friend is representative of many ColdFusion developers who come to the language without the formal computer science background that would have exposed them to working with arrays. While it’s certainly possible to eschew arrays, doing so deprives the coder of an excellent tool for handling a variety of situations. It’s like a carpenter who purposefully excludes a particularly helpful tool from his toolbox. The irony of the situation is that arrays – a member of the group of so-called “complex data types” – can actually make our code easier to work with and more adaptable to inevitable changes. This month, let’s do some work with arrays.

All variables can be thought of as name/value pairs. In this code

```
<cfset myName="Hal Helms">
```



By Hal Helms

the name of the variable is “myName”; the value of the variable is “Hal Helms”.

Variable values have a property known as a data type. Some languages (Java, for instance) require that you specify the data type a variable will hold before you use it. Such languages are

called “strongly typed.” ColdFusion is an example of a “weakly typed” language, one that does not require data type declarations. Behind the scenes, though, ColdFusion keeps track of the data type the variable holds. In the above example, the value of “myName” is of data type, “string”.

Data types can be simple or complex. Simple data types include strings, numbers and Boolean values. Complex data types include result sets (from queries), structures, and arrays. Arrays are just ordered collections of variables. By “ordered”, I mean that the order in which you place values in an array is significant and you can depend on ColdFusion to keep track of this. Structures, on the other hand, are unordered.

Let’s take the case where we have a set of numbers to keep track of – perhaps the numbers of days in the months

of the year. We could, of course, keep track of them individually:

```
<cfset january = 31>
<cfset february = 28>
...
<cfset december = 31>
```

That would give us 12 distinct variables to create and keep track of – not too bad, but as the number of such related variables increases, so does the complexity of managing them. And if you needed to pass these related variables into a function, as an example, the code quickly gets unwieldy. A better plan is to create an array.

It’s helpful to think of a spreadsheet as an analogy for an array.

Days	31	28	31	30	31	30	31	31	30	31	30	31
------	----	----	----	----	----	----	----	----	----	----	----	----

While spreadsheets use letters as column names, arrays use numbers:

	1	2	3	4	5	6	7	8	9	10	11	12
Days	31	28	31	30	31	30	31	31	30	31	30	31

Note:

The spreadsheet analogy, while a good one, is not perfect. The representation of this array as a single row with multiple columns could also be viewed as a series of rows with a single column and, in fact, many authors illustrate arrays in just this way. Both are equally valid as the dimensions in an array are virtual. However, I find that representing them as I have is easier for most people to understand.

Here's the code for creating a `daysInMonth` array:

```
<cfscript>
    daysInMonth = ArrayNew(1);
    daysInMonth[1] = 31;
    daysInMonth[2] = 28;
    daysInMonth[3] = 31;
    daysInMonth[4] = 30;
    daysInMonth[5] = 31;
    daysInMonth[6] = 30;
    daysInMonth[7] = 31;
    daysInMonth[8] = 31;
    daysInMonth[9] = 30;
    daysInMonth[10] = 31;
    daysInMonth[11] = 30;
    daysInMonth[12] = 31;
</cfscript>
```

The variable, `daysInMonth`, represents the entire array. To get information on a specific month, you provide the index for a specific month: `daysInMonth[6]`. Indices are placed in square brackets.

One of the nice features of arrays is that you can loop over the array easily:

```
<cfoutput>
```

```
<cfloop from=1
to="#ArrayLen(daysInMonth)#" index="i">
    #daysInMonth[i]#<br>
</cfloop>
</cfoutput>
```

Running this code produces this:



The function, `ArrayLen()`, returns the number of elements in the array – a useful feature when you don't know the number of array elements.

ColdFusion comes with several very useful array functions. While it's good to know that all of them exist, you're likely to use the following in routine code:

- `ArrayNew(dimension_number)`, as you've already seen, is used to create an array
- `ArrayLen(array)` returns the number of elements in the specified array
- `ArrayAppend(array, value)` places the value specified at the end of the referenced array

- `ArrayPrepend(array, value)` places the value specified at the beginning of the array, sliding the existing elements to the left
- `ArrayAverage(array)` returns the average of all elements in the referenced array

The snippet:

```
There are #ArrayAvg(daysInMonth)# days in
an average month
```

when run, produces:

There are 30.4166666667 days in an average month.

`ArrayClear(array)` removes not only the values of any elements in an array, but the elements themselves, something that we can verify by using the `ArrayLen()` function:

```
<cfset ArrayClear(daysInMonth)>
<cfoutput>
    There are #ArrayLen(daysInMonth)# ele-
ments in the array.
</cfoutput>
```

Subscribe Now and **SAVE!**



ONLY **\$89⁹⁹** for 12 monthly issues

Special Limited Time Offer

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

**COLD FUSION
Developer's Journal:**

- New Features of ColdFusion MX
- Tips, Techniques, and Tricks in Server-Side Programming
- DB Programming Techniques in CF
- High-Quality Technical Features
- <BF on CF> Column by Ben Forta
- Interviews with CF Makers and Shakers
- Display Ads of the Best Add-On Products
- CF User Group Information
- Product Reviews of the Best CF Add-Ons
- Q&A with ColdFusion Experts
- Full Working Source Code Samples



Running the above snippet produces this:

There are 0 elements in the array.

`ArrayDeleteAt(array,position)` removes the element at the position specified, sliding to the left any elements after the position indicated.

```
<cfset ArrayDeleteAt(daysInMonth,2)>
<cfoutput>
    There are #ArrayLen(daysInMonth)# ele-
ments in the array.
</cfoutput>
```

The above snippet produces this:

There are 11 elements in the array.

`ArrayInsertAt(array,position,value)` inserts a new element into the array, sliding to the right any elements after the position specified.

```
<cfset ArrayInsertAt(daysInMonth,12,35)>
<cfoutput>
    There are now #ArrayLen(daysInMonth)#
elements and the 12th month now has
#daysInMonth[12]# days in the month.
</cfoutput>
```

The above snippet produces this:

There are now 13 elements and the 12th month now has 35 days in the month.

`ArrayIsEmpty(array)` returns a Boolean value (true or false) depending on whether the array has any elements in it.

```
<cfset ArrayClear(daysInMonth)>
<cfoutput>
    There are now #ArrayLen(daysInMonth)#
elements in the array.
</cfoutput>
```

The above snippet produces this:

There are now 0 elements in the array.

`ArrayMax(array)` returns the largest element in the array.

```
<cfoutput>
    The greatest number of days in any
month is #ArrayMax(daysInMonth)#.
</cfoutput>
```

The above snippet produces this:

The greatest number of days in any month is 31.

`ArrayMin(array)` is the opposite of `ArrayMax()`.

```
<cfoutput>
    The fewest number of days in any month
is #ArrayMin(daysInMonth)#.
</cfoutput>
```

The above snippet produces this:

The fewest number of days in any month is 28.

`ArraySort(array, sort_type [, sort_order])` sorts the specified array by the `sort_type` and in the `sort_order` specified.

```
<cfset
ArraySort(daysInMonth,'numeric','asc')>
<cfoutput>
    <cfloop from="1"
to="#ArrayLen(daysInMonth)#" index="i">
        #daysInMonth[i]#<br>
    </cfloop>
</cfoutput>
```

The above snippet produces this:



`ArraySum(array)` adds all the elements in the array.

```
<cfoutput>
    There are a total of
#ArraySum(daysInMonth)# days in the year.
</cfoutput>
```

The above snippet produces this:

There are a total of 365 days in the year.

`ArrayToList(array [, delimiter])` returns a list composed of the elements in the

array, useful for times that you want to use list functions on your data. Note that the default delimiter is a comma.

```
<cfset dayList = ArrayToList(daysInMonth)>
<cfoutput>
    The ordered list of days in each month
is #dayList#.
</cfoutput>
```

The above snippet produces this:

The ordered list of days in each month is 31,28,31,30,31,30,31,31,30,31,30,31.


The value of an element in an array does not need to be a numeric value, as this code illustrates:

```
<cfset daysInWeek = ArrayNew(1)>
<cfset daysInWeek[1] = "Sunday">
<cfset daysInWeek[2] = "Monday">
<cfset daysInWeek[3] = "Tuesday">
<cfset daysInWeek[4] = "Wednesday">
<cfset daysInWeek[5] = "Thursday">
<cfset daysInWeek[6] = "Friday">
<cfset daysInWeek[7] = "Saturday">

<cfset allDays = ArrayToList(daysInWeek)>
<cfoutput>
    The days in the week are #allDays#.
</cfoutput>
```

The above snippet produces this:

The days in the week are Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday.

Further, the value of an element in an array is not limited to simple data types. One such construct is an array of arrays, in which each element in an array is itself an array. In ColdFusion, these are called multidimensional arrays, but that discussion must wait for next month. In the meantime, if you're interested in learning more about arrays, be sure to check out my monthly lesson and test available at www.halhelms.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

International Web Services Edge 2003

CONNECTING THE
ENTERPRISE WITH
WEB SERVICES,
JAVA, XML, AND .NET

March 18-20, 2003
Boston, MA

- ▶ Focus on Web Services
- ▶ Focus on XML
- ▶ Focus on JAVA
- ▶ Focus on .NET
- ▶ Focus on strategies for your enterprise from security to interoperability and more.

web services **EDGE**
conference & expo

The Largest
Web Services,
Java, XML, and .NET
Conference and Expo!

For more information visit
www.sys-con.com

Over 200 participating companies will display and demonstrate over 500 developer products and solutions.

Over 3,000 Systems Integrators, System Architects, Developers and Project Managers will attend the conference expo.

Over 100 of the latest sessions on training, certifications, seminars, case-studies, and panel discussions promise to deliver REAL Web Services Benefits, the industry pulse and proven strategies.



Hynes Convention Center
Boston, MA

London
June 3-5

Berlin
June 24-26

Hong Kong
Coming soon...



Boston call for papers opens
October 15, 2002.

Submit your papers online at:
www.sys-con.com/webservices2003east

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

JAVA DEVELOPER'S JOURNAL
WebSphere DEVELOPER'S JOURNAL
WebLogic DEVELOPER'S JOURNAL

WebServices JOURNAL

wireless JOURNAL

LINUX BUSINESS WEEK

XML JOURNAL

GF Advisor

.NET DEVELOPER'S JOURNAL

COLDFUSION DEVELOPER'S JOURNAL

PowerBuilder DEVELOPER'S JOURNAL



BlueDragon: Another Option for CF Developers

Imagine going to buy a new car. You are down to specifying the options for the vehicle and the salesman says, "You can have it in any color as long as it's black." If black is your favorite color that's great; but if it's not, then maybe they have just lost a customer. These days it's all about giving people choices.

Until recently, if you were developing ColdFusion applications you didn't have much choice. Macromedia was the only vendor providing a ColdFusion application server. Now, BlueDragon from New Atlanta has arrived to change all that. [Note: BlueDragon was originally a product called tagServlet developed by n-ary Limited. So it's not entirely new.] I have taken some time to check out BlueDragon, and here is what I have seen.



Reviewed By
Phil Cruz

What Is BlueDragon?

In a nutshell, BlueDragon is a family of Java-based products that implement the CFML (5.0) language. (There are a few CF5 tags that are not supported in BlueDragon, and support for CFMX language features is under development.) This allows you to develop and deploy CFML applications on an application server other than that provided by Macromedia. In addition, you have the ability to extend your applications with Java servlets, JavaServer Pages (JSP), or other J2EE technologies.

The product family consists of three products: BlueDragon Server, BlueDragon Server JX, and BlueDragon for J2EE. The products are priced competitively to Macromedia ColdFusion Server products, and range in cost from \$249/server-\$2499/

cpu. BlueDragon is free for development (like ColdFusion MX).

Check the New Atlanta site for details, but basically the server product is an application server to be used in conjunction with your Web server (it supports IIS, Apache, and Netscape/iPlanet) to serve CFML pages. Similar to CFMX,

it has a built-in Web server and can be used in standalone mode. The Server JX product adds the ability to serve JSP in addition to CFML. (While CFMX can execute JSPs as well, this is only available in the Enterprise Edition.)

The BlueDragon for J2EE product is quite different from the server products. Rather than being a complete application server, it is a portable CFML runtime module that allows for packaging your application into a Web application archive or WAR file and deploying onto any J2EE server. One interesting thing to note about the WAR file packaging is that BlueDragon/J2EE allows you to deploy your files as compiled binaries – without source – thereby allowing you to protect your intellectual property and source code. For a CF developer looking to migrate more seamlessly to J2EE and/or protect his or her source code, BlueDragon provides an answer.

Installation and Administration

Since I wanted to try the BlueDragon/J2EE product, I needed to obtain a J2EE compatible server. (Again, the two more basic versions are themselves complete application servers, like CFMX.) I chose to download Tomcat, which is readily available and free for deployment. Installing Tomcat went without a hitch; I installed it on default port 8080 so as not to conflict with IIS or CFMX running on the same box. Installing BlueDragon/J2EE was straightforward as well.

Essentially, BlueDragon/J2EE ships as a set of files that you drop into the Web application directory of the J2EE server. The template files include an index.cfm that displays a page so that you can verify your installation. You access the BlueDragon Administration screen by browsing to the /bluedragon/admin.cfm relative to your application URL, for example <http://localhost:8080/myapp/bluedragon/admin.cfm>.

The BlueDragon Administrator is laid out similarly to the CF Administrator and should feel familiar to any CF developer (see Figure 1). As you would expect, there are functions to set data sources, mappings, application/session variable timeouts, scheduled tasks, and mail server settings.

The BlueDragon Administrator differs from the CF Administrator in that it does not have functions for Verity (not supported in BlueDragon), debugging settings, log files, tag restrictions, and a few other miscellaneous settings. It does include a Deploy WAR File function that ColdFusion Server does not support. As for database support, the popular databases are supported and BlueDragon ships with New Atlanta's JTurbo, which is

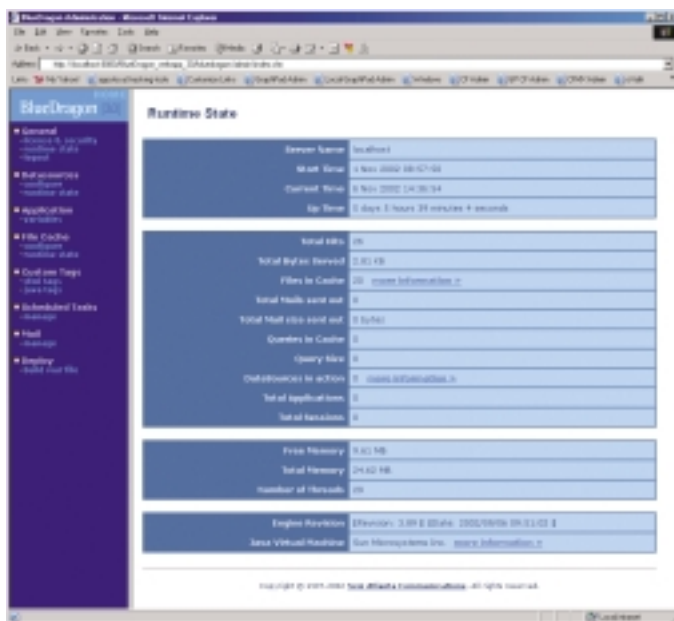


Figure 1: Screen shot of BlueDragon Administrator

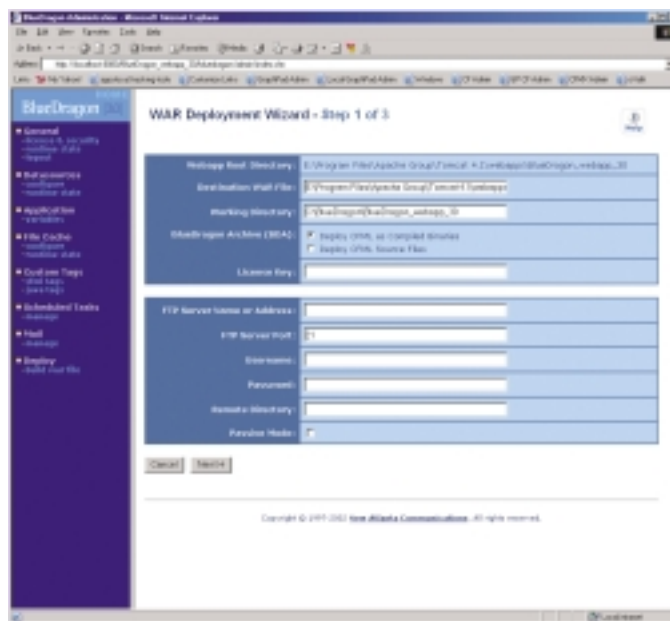


Figure 2: Screen shot of BlueDragon Administrator WAR deployment wizard

a Type 4 JDBC driver for connecting to Microsoft SQL server. I had no problems creating a data source to a Microsoft SQL server database.

Running Some Code

For my initial test, I tried running the employee directory application for the fictitious New Line Software company that ships as one of the example applications with ColdFusion 5. I selected this application because most people would have access to the code and it does all the basic interactions with a database: add, edit, delete, and search records. I copied the code from the CF 5 examples into the Web application directory with the other BlueDragon/J2EE files, created the data source, and launched the application – it just worked.

There's really not much to report. I tested all the functionality of the application and it worked flawlessly. Anyone browsing the application would have no indication that BlueDragon, not a ColdFusion Server, was serving pages.

Creating the WAR file for deployment was straightforward. From the BlueDragon Administrator, click "build war file" to launch the WAR deployment wizard (see Figure 2). You must provide a destination for the WAR file, select if you want to deploy source files or compiled binaries, and provide a license key. The following screen will ask which database drivers and optional tags you want to

include in your WAR file. Press finish and your WAR file is created.

Upon inspecting the contents of the WAR file (which is basically a zip archive), I confirmed the source code was not included in the package. In place of the source was a compiled binary file, webapp.bda. As for deployment, WAR deployment is handled slightly differently depending on the particular J2EE server. Tomcat supports "auto-deployment" in which you simply copy the WAR file to the \webapps directory of Tomcat and restart. I was able to auto-deploy the WAR file for my sample application with no problem.

Granted, the example application is very simple under the hood. I was eager to try BlueDragon with a "real" application. I tried to run a bug-tracking application that is built with Fusebox 3, has user authentication/authorization, extensive use of custom tags, and user-defined functions.

Before running the application, I did read the product documentation, which includes a CFML Compatibility Guide that discloses certain CFML tags that may be unsupported in BlueDragon or are supported, but with differences. I found the documentation to be clear, informative, and well written. Upon trying to run the bug-tracking application I ran into some issues that were both documented and undocumented.

As for documented issues, the application utilizes CFASSOCIATE and CFOBJECT with COM objects that are not sup-

ported in BlueDragon. I also found a custom tag that still uses the deprecated parameterExists() function. In CF5, parameterExists() is deprecated but still valid syntax; it is not valid syntax in BlueDragon. None of these are showstoppers and the code could be modified to remove dependence on them. One suggestion for New Atlanta is to provide a "code compatibility analyzer" utility that you could run on your source code to flag such issues, similar to what CFMX has.

As for undocumented issues, I ran into a bug involving structure deserialization using WDDX. Also, I ran into an issue related to <cfmodule> syntax. In CF5/CFMX, you can call a tag like <cfmodule template="mytag.cfm"> </cfmodule> and because it has an end tag it will be executed twice. You can also code it like <cfmodule template="mytag.cfm"/> (like XHTML syntax); that implies the end tag and will also execute the tag twice. Using the latter form in BlueDragon, the code was executing only once. I raised both of these issues on the BlueDragon mailing list and New Atlanta was quick to confirm the bugs as well as indicate that they would be resolved in an upcoming service pack.

So, while I did run into some issues, none proved to be insurmountable and I'm quite pleased at the prospect of having more deployment options for my application. I have not done any performance/load testing. From what I hear from

product review

New Atlanta, BlueDragon generally outperforms ColdFusion Server 5 and is on par with CFMX. As usual, do your own testing and your mileage may vary.

CFML – The New Standard?

I think we can agree that competition is a “good thing.” Competition can put pressure on vendors to provide quality service/products at competitive prices. However, now that BlueDragon has entered the game, what happens if New Atlanta extends the CFML language with their own features? In fact, they have already indicated their intent to do this by implementing a <CFFORWARD> tag that would allow you to forward the request to a JSP, similar to how the <jsp:forward> tag does.

The potential is there for a “fracturing” of the CFML “standard” and having code that will run on BlueDragon, but have issues on ColdFusion Server or vice versa. Having endured the browser-compatibility wars, I for one would not look forward to having to write conditional code all throughout my applications to deal with “CF server compatibility” issues.

It is possible, though perhaps unlikely, that Macromedia will implement a standards process similar to the Java Community Process that Sun has, and allow other vendors to provide input to the future of CFML. Lacking that, New Atlanta has said that there is one standard for CFML and that is what is provided by Macromedia in terms of published documentation, and what is implemented in the actual server products.

Summary

BlueDragon is not a “ColdFusion killer” and it’s not meant to be. If anything it’s an affirmation and indication of maturity of the ColdFusion market. It addresses some issues that Macromedia’s ColdFusion does not. BlueDragon is not 100% compatible with ColdFusion (and doesn’t claim to be) or bug free (no product is), but I found it to deliver as advertised. It’s another choice available to us as developers, and one worth checking out.



About the Author

Phil Cruz is a certified ColdFusion developer. He has over 10 years of experience in the computing industry and specializes in the design and development of Web-based applications.

philcruz@yahoo.com

Vitals

BlueDragon • New Atlanta Communications, LLC
100 Prospect Place • Alpharetta, GA 30005
Phone: 678 256-3011 • Fax: 678 256-3012
E-mail: info@newatlanta.com general sales@newatlanta.com sales
Test Environment: Windows 2000 Server



Editorial —continued from page 5

to mention, because it’s viewed by many in our always-interesting industry as a touchy subject, and one worth covering.

We’ve commissioned a product review of New Atlanta’s BlueDragon, a Java-based server that also compiles CFML pages. The current release, 3.0, supports most of the standards of the CFML 5.0 language, with an MX-compatible release under construction. Phil Cruz’s in-depth review hits upon some of the high and low points of the server, and it’s worth a read.

We’ve had several e-mails in the past few months from people asking us to give the product a good look, and even more e-mails from people wondering where it fits into the ColdFusion market. I think that the product is good for the ColdFusion market because it shows the embrace of CFML as a standard language and, on the server side, the ColdFusion world as a maturing market.

Till next month – happy coding!



Q&A —continued from page 7

the user input “Tom, Dick, Harry”, I’m trying to get the query to execute something like “SELECT * FROM TABLE WHERE Name LIKE ‘%Tom%’ OR Name LIKE ‘%Dick%’ OR Name LIKE ‘%Harry%’”.

How can I create the query to do what I want?

A: The question is really, “How do I dynamically build queries?” and it’s an issue that CF developers need to address in almost every application. The bottom line is this: you can use CF tags nested inside your CFQUERY tag to dynamically generate the necessary SQL. For your scenario, there are several ways to solve the problem, but here’s my suggestion (see Listing 3).

Listing 3

```
<CFQUERY NAME="qSearch" DATASOURCE="YourDNS">
SELECT * FROM TABLE
WHERE Name = 'PlaceholderText'
<CFLOOP LIST="#Form.SearchText#" INDEX="SearchItem">
    OR Name LIKE '%Trim(SearchItem)%'
</CFLOOP>
</CFQUERY>
```

First, you’ll need to create what I call a “placeholder” in your SQL. This is a valid SQL statement that you put immediately after the WHERE clause, but it has no real impact on your query; it’s just there to make the rest of your query easier to generate. In this case, I’ve used the statement “WHERE Name = ‘PlaceholderText’”. The likelihood of having a name equal to “PlaceholderText” is pretty slim.

Next, you need to dynamically build your OR statements by looping over the list that the user submitted. For example, if your form field is called “SearchText” and they submit “Tom, Dick, Harry”, you would use CFLOOP to loop over the values in Form.SearchText. Inside your loop, simply create the OR condition for your query using the value of the list index as your search criteria. Be sure to use the Trim() function to eliminate any spaces that the user may have placed after the commas. The end result will be a SQL query that contains an OR clause for each term that the user supplies.



Info

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



SOLD
SEPARATELY
ONLY
\$71⁹⁹

HURRY!

LIMITED TIME
OFFER WHILE
SUPPLIES
LAST!



SOLD
SEPARATELY
ONLY
\$89⁹⁹

**\$198
VALUE**
FROM

Ultimate

**COLDFUSION
CDs**

ONE LOW PRICE!

The most complete library of exclusive
ColdFusion articles on CD!

CDs are edited by *ColdFusion Developer's Journal* Editor-in-Chief
Robert Diamond and organized into 44 chapters containing
more than 650 exclusive ColdFusion articles.

Applications	Database	Interviews	Source Code
CF & Java	E-Commerce	News	CFSpectra
CF Basics	Editorial	Object-Oriented CF	Upgrading CF
CF Server	Enterprise CF	Programming Tips	User-Defined Functions
CFDJ List	Error Handling	Q&A	Verity
CFMX	Flash	Reviews	WDDX
CF Tips & Techniques	Forms	Scalability	Web Services/XML
Custom Tags	Fusebox	Security	Wireless
	Hosting		

2 CD OFFER

"CFA: THE COMPLETE WORKS"
PLUS

"CFDJ: RESOURCE CD"

ONLY **\$99⁹⁹!**



Order Online at

WWW.JDJSTORE.COM

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Sold Exclusively through...



ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES ARE TRADE NAMES, SERVICE MARKS, OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.

Creating and Accessing Collection Files

ColdFusion components make it simple

In computer terms, objects are abstracts of real-world entities that are all around us in everyday life. Examples of such objects could be people, organizations, cars, animals, or even less tangible entities such as math or time.

Of these examples, only the first four can be considered collections. One or more persons is a collection of people; one or more organizations is a collection of organizations, and so on. However, math and time is not a collection of math and time is not a collection of time. They are just simple objects with methods and properties. Simply defined, a collection is a set of related objects.

Several languages already use built-in objects that mimic real-world entities. Many of you may already be familiar with some form of Visual Basic or JavaScript. In Visual Basic, a group of related objects is called a collection. Although the term collection is foreign to the JavaScript language, the language does have objects that act like collections. For instance, the forms object is a collection. You can loop through the collection and access the individual objects using array syntax such as `document.forms[0]`.

Collections come with many advantages. First, they are objects, so any and all advantages of objects, such as encapsulation and reusability, are inherent to the collection. Second, you can access a group of objects easily by looping through the collection and executing the same



By Steve Parks

code against each object in the collection. Finally, you can simplify a lot of the external data manipulation by using collections. For instance, instead of returning a query containing users (which might change in six months) you could use a Users collection. The User object that is stored in the Users collection would

encapsulate all the code that goes into retrieving the user data. That way, when the database changes, you can change your User object while your collection and the code accessing the collection remain the same.

What Makes a Collection?

There are three hard and fast rules that an object (or in ColdFusion, a component) has to adhere to in order to be a collection:

1. Every object in the collection can be accessed individually. Each object in the collection is unique in instance, if not data. If there are five objects in the collection, you should be able to call properties and methods from each object.
2. Every object in the collection is the same type of object and has the same

properties and methods. This means that you wouldn't have a collection called People with two types of objects: People and Computers. You would, in fact, have two collections, the People collection and the Computers collection.

3. A collection usually has at least four methods: Add, Remove, Count, and Item.

Most collections can be completely accessed through the use of just four methods:

- **Add()**: Adds a new object to the collection
- **Remove()**: Removes an object from the collection
- **Item()**: Retrieves an object from the collection (for use in the application)
- **Count()**: Returns the number of objects in the collection (for looping purposes)

Creating the Collection Files

In order to create a collection we need to create two files, both of which are CFCs. The first CFC we'll create will be named `ActiveUsers.cfc`; the second will be named `ActiveUser.cfc`. The `ActiveUser.cfc` file will be the actual object that is contained in our collection. The collection filename is usually the plural term for the name of our object, hence, `ActiveUser` object and `ActiveUsers` collection. The `ActiveUser.cfc` file is shown in Listing 1.

The `ActiveUsers` collection will have four methods: `Add(strFirstName, strLastName)`, `Remove(index)`, `Item(index)`, and `Count()`. Our `ActiveUser` object will contain five

methods: GetName(), SetName(strFirstName, strLastName), SetLoginTime(), GetLoginTime(), and GetMinutesActive().

Our ActiveUsers collection will be used to display only users who are currently active on our Web site, similar to a "who's on" interface. Anytime you access a variable in a shared scope, such as the Application scope, you should lock it using CFLOCK.

The first step is to set up a data structure to store our data in the ActiveUsers collection. We need only one private (or member) variable called arItems, which is an array. You can add that code just inside of your CFCOMPONENT tag as shown below.

```
<CFCOMPONENT>
    <CFSET arItems = ArrayNew(1)>
</CFCOMPONENT>
```

Next we're going to add four methods to the component. The first method is the Add method. The Add method will take two arguments, FirstName and LastName, both string literals. The Add function simply adds an element to the private variable arItems using the intrinsic function, ArrayAppend(). The tricky part is that our Add method needs to add a value that represents an instance of an object. Since we can't use the CFOBJECT tag inside the ArrayAppend() function, we will use the CreateObject() function, shown below. After we add that element to the array, we also need to get the length of the array so we know the index of our recently added user. Again, locking is important here, because we want to add an element to the array and then get the length of the array before anyone else adds another element to it. This is sometimes called a race or race to commit, in database terminology. If we don't lock the actions, we will be racing against other requests to add elements to the array, potentially returning the wrong values to the calling page.

```
<CFSET ArrayAppend(arItems, CreateObject("component",
    "ActiveUser"))>
<CFSET idx = ArrayLen(arItems)>
```

The next two lines of code in our function will be calling methods from the prebuilt ActiveUser object. These methods set three internal variables in the ActiveUser object: the user's first name, the user's last name, and the date the user logged in.

```
<CFSET arItems[idx].SetName(Arguments.FirstName,
    Arguments.LastName)>
<CFSET arItems[idx].SetLoginTime(Now())>
<CFRETURN arItem[idx]>
```

The last thing we do is return the current element of the array for the programmer to use in the calling page, if necessary. That was the hard part. The next three functions are short and sweet, so I'm going to spend more time describing why you need to do this, rather than stepping through the code line by line. Listing 2 shows the code for the entire collection object.

With the Remove method, we require one argument that is an integer representing the index of the internal array,

Pacific Online

www.paconline.net

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.ACTIVEPDF.COM	949.582.9002	4
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.DYNAMICS	25
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX.HOST	3
COLDFUSION DEVELOPER'S JOURNAL	WWW.COLDFUSIONDEVELOPERSJOURNAL.COM	888.303.5282	37
COOLFUSION	WWW.COOLFUSION.COM	631.737.4668	15
CTIA WIRELESS 2003	WWW.CTIASHOW.COM		51
EDGE WEB HOSTING	WWW.EDGEWEBHOSTING.NET/CFDJ		19
EKTRON	WWW.EKTRON.COM/FREE		13
FUSETALK	WWW.FUSETALK.COM/NEW	866.477.7542	6
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	27
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	52
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	43,49
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CFMXLIGHT	877.460.8679	21
MACROMEDIA	WWW.MACROMEDIA.COM/GO/USERGROUPS	877.460.8679	23
NETQUEST	WWW.NOCONTENT.COM		11
NEW ATLANTA COMMUNICATIONS			2
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	45
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	35
SYS-CON SUBSCRIPTIONS	WWW.SYS-CON.COM/SUBOFFER.CFM	888.303.5282	30-31
WEB SERVICES EDGE 2003	WWW.SYS-CON.COM	201.802.3069	39

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 agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in ColdFusion Developer's Journal. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

arItems. We need to check to make sure the value passed is less than or equal to the length of the array so there aren't any errors produced, and then we simply remove that element using the `ArrayDeleteAt()` function. Remember, you'll need to lock this action to prevent the wrong index of the array from being deleted. In a situation where the method is called twice, there will be a race to commit the deletion. In the event the deletion was not locked, the array would be resized after the first deletion and then the wrong index could potentially be deleted during the second deletion.

The `Item` method also requires an index value, but the result is different. Instead of removing an item (or doing any type of action), we're just going to return the instance of the `ActiveUser` object that was stored at that index. This allows us to access each individual object from external ColdFusion templates easily, as you'll see later.

The `Count` method accepts no arguments and returns an integer describing the number of objects in our collection. We use the `ArrayLen()` function to determine the return value.

That's it for our collection. All of the internal code is written for our `ActiveUsers` collection to work properly. It's relatively easy to generate code for collections once you understand the concepts. Now we're going to look at how to use collections in ColdFusion templates.

Making the Collection Persist

In order for our "who's on" application to persist, we need to set up the `ActiveUsers` collection in the `Application` scope. Open the `Application.cfm` file in Listing 3 and add the following code:

```
<CFIF NOT IsDefined("Application.
objActiveUsers")>
<CFOBJECT NAME="Application.
objActiveUsers" COMPONENT="ActiveUsers">
</CFIF>
```

This code basically says to check whether the `Application.objActiveUsers` collection is defined at the beginning of every request. If it isn't defined, it will create the collection using the `CFOBJECT` tag.

Accessing the Collection

In the `index.cfm` file (see Listing 4) we're going to output the current number of users online and then loop through the `ActiveUsers` collection and output data pertaining to the users that are online. You can do all this using the code below:

```
<CFSET iCount =
Application.objActiveUsers.Count()>
```

As you'll see, we've set the `iCount` variable to the value of the `ActiveUsers` collection's `Count` method. This lets us know how many active users are online right now. Now we can loop through the collection using a simple `FROM/TO` (or `FOR`) loop where the `TO` attribute is equal to the number of users online, or `iCount`.

We can access the individual objects in the collection using the `Item(index)` property such as `Application.objActiveUsers.Item(i)`. In the code below, we use that notation to call different methods of the object, such as `GetName()`, `GetLoginTime()`, and `GetMinutesActive()`.

```
<CFLOOP FROM="1" TO="#iCount#" INDEX="i">
  <CFSET stName = Application.objActive
    Users.Item(i).GetName()>
  #i# - #stName["FirstName"]#
  #stName["LastName"]# -
  #DateFormat(Application.objActive
    Users.Item
      (i).GetLoginTime(), 'm/d/yy')#
  #TimeFormat(Application.objActive
    Users.Item
      (i).GetLoginTime(), 'h:mm tt')# -
  #Application.objActiveUsers.Item(i).Get
    MinutesActive()#<BR>
</CFLOOP>
```

Running `index.cfm` at this point in time would result in the printing of "Number of Users: 0", since we haven't added any users to the collection. In order to add users to the collection we have to call the `Add()` method of the `ActiveUsers` collection. In this example, we'll make a ColdFusion page called `AddUser.cfm` (see Listing 5) that requires two URL variables called `FirstName` and `LastName`. The following code is all we'll use for our `AddUser.cfm` file:

```
<CFSET objActiveUser = Application.objActive
Users.Add(Trim(URL.FirstName),
```

```
Trim(URL.LastName))>
<CFLOCATION URL="index.cfm">
```


If we access the `AddUser.cfm` page passing "Steve" and "Parks" through the query string such as

```
AddUser.cfm?FirstName=Steve&LastName=Parks
```

our page will then add the user to the collection and refresh us to the `index.cfm` page. You should now see that there is one user named Steve Parks in the collection. If your browser caches the `index.cfm` page, you may have to manually refresh the page. In the `index.cfm` file we called the `GetName()`, `GetLoginTime()`, and `GetMinutesActive()` methods from the `ActiveUser` object in the collection. You'll now see that data outputted through the collection. Try it again, adding another user using your name. You should now see that you are in the collection as well.

The last file that needs to be created is the `RemoveUser.cfm` file, which will remove users from the collection. Instead of passing the `FirstName` or `LastName` of the user, you'll need to pass a valid index. In this example, we'll just remove the first index, but in theory you would remove users based on an algorithm possibly determined by a method called `GetMinutesIdle()`. To remove the first user from the application, call the file `RemoveUser.cfm`, shown in Listing 6, passing a value of 1 in the attribute `index`, as shown below:

```
RemoveUser.cfm?Index=1
```

We've now created our `ActiveUsers` collection in a simplified form. You can add properties and methods to the `ActiveUser` object to make the system more robust. I recommend adding locking and validation to the `Application` scope and methods, respectively. 

About the Author

Steve Parks, CEO of Adept Developer Consulting, Inc. (www.adeptdeveloper.com), is a Macromedia Certified Advanced ColdFusion Developer. Steve has seven years of experience developing Web applications and relational databases.

steve@adeptdeveloper.com

ATTN: Developers

**STEP UP
to the mike
and be...**

Go to
<http://developer.sys-con.com>

HEARD!

**Calling Sleek
Geeks Everywhere!**

Make sure you have your finger on
the pulse of i-Technology...bookmark
<http://developer.sys-con.com> today.

i-Technology

News

i-Technology

Views

i-Technology

Comment

i-Technology

Debate



Listing 1: ActiveUser.cfc

```
<CFCOMPONENT DISPLAYNAME="ActiveUser" HINT="This object holds data
specific to a user. Each ActiveUser object will be part of the
ActiveUsers Collection">
    <CFSET stName = StructNew(>
    <CFSET dtLogin = ''>
    <CFFUNCTION NAME="GetName" ACCESS="public" RETURNTYPE="STRING">
        <CFRETURN stName>
    </CFFUNCTION>
    <CFFUNCTION NAME="SetName" ACCESS="public" RETURNTYPE="VOID">
        <CFARGUMENT NAME="FirstName" TYPE="string" REQUIRED="Yes">
        <CFARGUMENT NAME="LastName" TYPE="string" REQUIRED="Yes">
        <CFSET stName["FirstName"] = Arguments.FirstName>
        <CFSET stName["LastName"] = Arguments.LastName>
    </CFFUNCTION>
    <CFFUNCTION NAME="SetLoginTime" ACCESS="public">>
        <CFARGUMENT NAME="dt" TYPE="date" REQUIRED="Yes">
    </CFFUNCTION>
    <CFSET dtLogin = dt>
</CFFUNCTION>

<CFFUNCTION NAME="GetLoginTime" ACCESS="public" RETURNTYPE="DATE">
    <CFRETURN dtLogin>
</CFFUNCTION>
<CFFUNCTION NAME="GetMinutesActive" ACCESS="public"
RETURNTYPE="numeric">
    <CFSET var iMinutes = 0>
    <CFIF DateDiff('n', dtLogin, Now()) GT 0>
        <CFSET iMinutes = DateDiff('n', dtLogin, Now())>
    <CFELSE>
        <CFSET iMinutes = 1>
    </CFIF>
    <CFRETURN iMinutes>
</CFFUNCTION>
</CFCOMPONENT>
```

Listing 2: ActiveUsers.cfc

```
<CFCOMPONENT DISPLAYNAME="ActiveUsers" HINT="This is the collection
object. It is essentially a set of ActiveUser objects">
    <CFSET arItem = ArrayNew(1)>
    [Add a returnType="ActiveUser" here. -te]
    <CFFUNCTION NAME="Add" ACCESS="public" RETURNTYPE="ActiveUser">
        <CFARGUMENT NAME="FirstName" TYPE="string" REQUIRED="yes">
        <CFARGUMENT NAME="LastName" TYPE="string" REQUIRED="yes">
    </CFFUNCTION>
    <CFSET ArrayAppend(arItem, CreateObject("component",
"ActiveUser"))>
    <CFSET var idx = ArrayLen(arItem)>
    <CFSET arItem[idx].SetName(Arguments.FirstName,
Arguments.LastName)>
    <CFSET arItem[idx].SetLoginTime(Now())>
    <CFRETURN arItem[idx]>
</CFFUNCTION>
    <CFFUNCTION NAME="Count" ACCESS="public" RETURNTYPE="numeric">
        <CFRETURN ArrayLen(arItem)>
    </CFFUNCTION>
```

```
<CFFUNCTION NAME="Item" ACCESS="public" RETURNTYPE="ActiveUser">
    <CFARGUMENT NAME="Index" TYPE="numeric" REQUIRED="Yes">
    <CFRETURN arItem[Index]>
</CFFUNCTION>
    <CFFUNCTION NAME="Remove" ACCESS="public" RETURNTYPE="VOID">
        <CFARGUMENT NAME="Index" TYPE="numeric" REQUIRED="yes">
    </CFFUNCTION>
    <CFLOCK NAME="LockDeletion" TYPE="EXCLUSIVE" TIMEOUT="15">
        <CFIF Index LTE ArrayLen(arItem)>
            <CFSET ArrayDeleteAt(arItem, Index)>
        </CFIF>
    </CFLOCK>
    <CFRETURN "True">
</CFFUNCTION>
</CFCOMPONENT>
```

Listing 3: Application.cfm

```
<CFAPPLICATION NAME="AppCollections" SESSIONMANAGEMENT="Yes">
    <CFIF NOT IsDefined("Application.objActiveUsers")>
        <CFOBJECT NAME="Application.objActiveUsers"
COMPONENT="ActiveUsers">
    </CFIF>
```

Listing 4: Index.cfm

```
<CFOUTPUT>
<CFSET iCount = Application.objActiveUsers.Count()>

Number of Users: #iCount#
<BR>
<BR>
    <CFLOOP FROM="1" TO="#iCount#" INDEX="i">
        <CFSET stName = Application.objActiveUsers.Item(i).GetName()>
        #i# - #stName["FirstName"]# #stName["LastName"]# -
#DateFormat(Application.objActiveUsers.Item(i).GetLoginTime(),
'm/d/yy')#
#TimeFormat(Application.objActiveUsers.Item(i).GetLoginTime(),
'h:mm tt')# -
#Application.objActiveUsers.Item(i).GetMinutesActive()#<BR>
    </CFLOOP>
</CFOUTPUT>
```

Listing 5: AddUser.cfm

```
<CFSET objActiveUser =
Application.objActiveUsers.Add(Trim(URL.FirstName),
Trim(URL.LastName))>
    <CFLOCATION URL="index.cfm">
```

Listing 6: RemoveUser.cfm

```
<CFSET Application.objActiveUsers.Remove(Int(Trim(URL.Index)))>
    <CFLOCATION URL="index.cfm">
```


THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

Go to www.SYS-CON.com

The most innovative products, new releases, interviews, industry developments, and plenty of solid *i*-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at www.sys-con.com.

JAVA Industry > Newsletter

WebServices Industry > Newsletter

XML JOURNAL Industry > Newsletter

wireless Industry > Newsletter

WebLogic Industry > Newsletter

WebSphere Industry > Newsletter

ColdFusion Industry > Newsletter

.NET Journal Industry > Newsletter

SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS! CHOOSE ONE OR TRY THEM ALL!

FREE

E-Newsletters

SIGN UP TODAY!

Exclusively from the World's Leading *i*-Technology Publisher

SHOP ONLINE AT **JDJSTORE.COM** FOR BEST PRICES OR CALL IN YOUR ORDER AT **1 888-303-JAVA**

BUY THOUSANDS OF PRODUCTS AT GUARANTEED LOWEST PRICES!

GUARANTEED BEST PRICES

FOR ALL YOUR SOFTWARE AND WIRELESS NEEDS

MACROMEDIA® ColdFusion MX Server Pro
Multiplatform Upgrade

Rapidly build and deploy dynamic sites and rich Internet applications with ColdFusion MX. CFMX is a powerful environment for rapidly building and deploying dynamic sites and rich Internet applications. Its easy server scripting environment includes tag-based language, new server-side ActionScript, and complete integration with Dreamweaver MX (sold separately).

Macromedia® ColdFusion MX Server Pro\$514⁰⁰

MACROMEDIA® JRun 4 Win/Linux/Unix
Full 1 CPU

Macromedia JRun 4 is the fast, affordable, and reliable solution for delivering Java applications. New features such as drag-and-drop deploy, XDoclet integration, and optimized Macromedia Flash connectivity speed the development and deployment of your next-generation Internet applications. Currently used in production at over 10,000 companies worldwide, JRun has confirmed reliability for powering J2EE applications.

Macromedia® JRun 4 Win/Linux/Unix.....\$849⁰⁰

MACROMEDIA® ColdFusion Server 5 Enterprise
Multiplatform Upgrade

Rapidly build and deploy dynamic sites and rich Internet applications with ColdFusion MX. CFMX is a powerful environment for rapidly building and deploying dynamic sites and rich Internet applications. Its easy server scripting environment includes tag-based language, new server-side ActionScript, and complete integration with Dreamweaver MX (sold separately).

Macromedia® ColdFusion MX Server Enterprise\$2319⁰⁰

WWW.JDJSTORE.COM

Macromedia Contribute Makes Web Site Updates As Easy As Word Processing

(San Francisco) – Macromedia, Inc., has announced the release of Macromedia Contribute, a groundbreaking new desktop application that enables anyone to easily update, add, and publish Web content to existing Web sites without requiring technical skills beyond basic word processing. The product brings ease and simplicity to the Internet by putting the power of Web publishing into the hands of nontechnical users, while also providing sufficient control for the professionals who manage Web sites. Macromedia Contribute works with any HTML Web site, including those coded by hand or created with tools like Macromedia Dreamweaver MX or Microsoft FrontPage.

"Macromedia Contribute represents a new product vision for Macromedia that reaches a whole new class of users with a beautiful solution that also frees up our current designers and developers to stay ahead of the technology curve," says Rob Burgess, chairman and CEO, Macromedia. "Contribute levels the playing field for Web content contribution, and makes waiting for an overburdened Webmaster to make site updates a thing of the past."

"Macromedia Contribute allows all of our content owners, regardless of their work discipline, to have full control over their Web content," says Christopher L. Rogers, lead programmer/analyst, training systems and support, The Boeing Company. "With an intuitive user interface, content release management tools, powerful administration controls, and one-button publishing, Macromedia

Contribute enables our team members to easily and efficiently become part of the information release process."

Macromedia Contribute allows users to update Web content in three simple steps. Users browse to the Web page they want to update, edit the page, and publish their updated

page to the live site. Macromedia Contribute, available for Windows in December, features an introductory price of \$99. It will be available for Mac OS X users in 2003. To download a technology preview, go to www.macromedia.com/go/contribute.

Userplane Releases Macromedia Flash-Based Instant Messenger (Los Angeles) – Userplane, a Macromedia partner, has announced the release of a Flash client instant messenger for corporate intranet use, online learning, online dating Web sites, customer service, and direct online sales.

"Macromedia Flash is the perfect platform for instant messaging," says Kevin Prentiss, a partner at Userplane. "It allows users to send interactive animations between clients, and send audio and video along with standard instant message text services. Teamed with Flash Communication Server as our back end, we now can offer cost-effective instant message solutions to any Web site." www.totekasche.com

NetIQ Brings WebTrends Web Analytics to Macromedia Dreamweaver MX Developers (San Jose, CA) – NetIQ Corporation, a provider of systems management, security management, and Web analytics solutions, announced the availability of its free WebTrends Developer Kit for Macromedia DreamweaverMX. The kit allows developers to easily build more effective Web sites by integrating Web analysis into the development cycle.

Developers can now automatically tag Web pages for analysis from within Dreamweaver MX, thus minimizing hand-coding time and errors.

The WebTrends Developer Kit is available free of charge at www.netiq.com/products/wrc/developer.asp.

Discovering CFCs, by Hal Helms, Now Available

Discovering CFCs, by **ColdFusion Developer's Journal's** very own Hal Helms, and Ben Edwards, explores the possibilities and pitfalls of developing ColdFusion applications using ColdFusion components. CFCs heavily borrow OO concepts, and the book clearly explains why and how to make use of polymorphism, inheritance, and encapsulation to create powerful, robust applications. There are full examples given in the book, and an accompanying workbook of more problems and exercises is available from techspedition.com.



Montara and New Atlanta Partnership Creates Platform for Easy Web Content Management

(Atlanta, GA) – A new partnership between two Atlanta-based companies, Montara Software, Inc., and New Atlanta Communications, will soon produce an innovative new Web content management system that will allow nontechnical people unprecedented access to Web publishing.

Thanks to the agreement announced between Montara, maker ofAlchemy EX, and New Atlanta, maker of BlueDragon 3.0, the Alchemy EX content management system – designed for small-to medium-sized businesses and easy access by J2EE and CFML (ColdFusion Markup Language) developers – will be released in beta form in Q1.

Alchemy EX provides the infrastructure for developers to create custom components for a variety of distribution methods, including Web services. BlueDragon 3.0 facilitates deployment of CFML Web applications onto

standard J2EE platforms as part of Alchemy EX. The foundation of the two provides a flexible framework that not only scales, but can be easily manipulated afterward by nontechnical employees who want to update their content themselves, quickly and easily.

The partnering of New Atlanta and Montara products means that Web content can be managed without a full-time programmer at the helm and that developers can build programs without the limits of platform or media distribution type. This is a strategic solution, especially for small- to medium-sized businesses without an IT department. In addition, the Alchemy EX system can also be implemented as a supplied online application, costing much less than a current conventional Web content system.

www.montarasoftware.com
www.newatlanta.com

 **New Atlanta**



CTIA WIRELESS 2003

www.ctiashow.com

Intermedia

www.intermedia.net