WEB-PAGE FORM NIGHTMARES Pg. 18

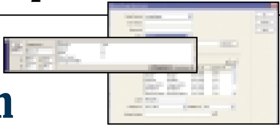
U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

November 2002 Volume: 4 Issue: 11

ColdFusion Developer's Journal
CFDJ
READERS' CHOICE AWARD
ANNOUNCING THE 2002 CFDJ RCA WINNERS
See pg.34



RETAILERS PLEASE DISPLAY
UNTIL JANUARY 31, 2003

$8.99US $9.99CAN

12>

0 71486 02689 1

SYS-CON MEDIA

COLDFUSION Developer's Journal

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 Stilley jennifer@sys-con.com

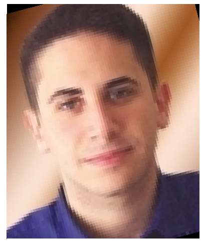
production
production consultant
Jim Morgan jim@sys-con.com
art director
Alex Botero alex@sys-con.com
associate art directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com

contributors to this issue
Charles Arehart, Olivier Bridgeman, Ben Forta, Hal Helms, Simon Horwith, David Levesque, Carey Lilly, Tom Nunamaker, Tom Peer, Deanna Schneider, David Shadovitz, 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

Constantly Improving

Welcome to a visually new and improved ***ColdFusion Developer's Journal***. You'll notice subtle changes in certain aspects of the design and layout of the magazine. We've also made a few behind-the-scenes changes to the roster of in-house sous-chefs, those who help cook the magazine each month and deliver an even better ***CFDJ***.



By Robert Diamond

We're working on giving the magazine a somewhat cleaner look, and I trust you'll agree that what you hold here in your hand this month is a step in the right direction. If we've succeeded, you should find that you've been served up the same great meat – or tofu if you're a vegetarian – only on an even nicer set of china. (Don't you just love analogies?)

Here at ***CFDJ***, the design of the magazine isn't the only thing we're working to improve. A significant project has also begun that involves the Web site. I'll talk more about that in an upcoming issue, but in the meantime, we're hard at work planning the magazine's 2003 "editorial calendar."

For the benefit of those of you unfamiliar with the term *editorial calendar*, this is the calendar of topics we plan to cover each month in the course of the coming year. It's not something that we stick to 100% by any means – that would be nigh-on impossible in an industry like ours. But it does help provide a framework for our writers, for the marketing plans of our advertisers, and for us, to help define the magazine's vision as a whole.

That vision is ever-evolving, with one constant goal: that of improvement. That goal starts with ***CFDJ*** as a magazine, here to serve you, the reader. If we're doing our job right, it also extends to helping you – through the "tech meat" we bring you each month – to improve and expand your skillset as a developer. It's a win-win approach and not a bad mantra to live by, especially in the current economic climate.

We are currently analyzing reader feedback and working with our editorial board to plan another year's worth of quality content. There are several themes resounding through the next few months, and I'd like to give you a sneak preview. The first is an increased focus on making your code better, both for beginners and advanced developers. This issue becomes even more relevant nowadays with ColdFusion MX, because many of the tricks and techniques we've all been using up through CF 5 to tweak the performance of CF are no longer needed.

For starters, MX brings us a whole new way of caching, and there are further areas to explore on how to do things better. This goes hand-in-hand with more general MX coverage, because MX, on its own, brings a specific range of topics like UDFs and CFCs – that almost every developer I've spoken to is finding new reasons to love.

Another hot topic will be Flash, which has been getting hotter throughout the early stages of the MX cycle, with real smoke starting to come from the fire with its remoting capabilities. XML, Web services, and Java will continue to be roads that many of us will be going down in the year ahead, and we're planning content to help guide you too, and to keep you from getting lost.

These are just some of the many topics we'll be covering over the next several months in

—continued on page 44

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

SYS-CON MEDIA

president & ceo
Fuat A. Kircaali fuat@sys-con.com

business development
vp, business development
Grisha Davida grisha@sys-con.com
COO/CFO
Mark Harabedian mark@sys-con.com

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

sys-con events
vp, events
Grisha Davida grisha@sys-con.com
conference manager
Michael Lynch mike@sys-con.com
sales executives, exhibits
Michael Pesick michael@sys-con.com
Richard Anderson richarda@sys-con.com

customer relations
customer service representative
Margie Downs margie@sys-con.com
manager, customer relations
Rachel McGouran rachel@sys-con.com

sys-con.com
vp, information systems
Robert Diamond robert@sys-con.com
web designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com

online editor
Lin Goetz lin@sys-con.com

accounting
assistant controller
Judith Calnan judith@sys-con.com
accounts receivable
Kerri Von Achen kerri@sys-con.com
accounts payable
Joan LaRose joan@sys-con.com
accounting clerk
Betty White betty@sys-con.com

subscriptions
Subscribe@sys-con.com
1 888 303-5282
For subscriptions and requests for bulk orders,
please send your letters to Subscription Department
Cover Price: $8.99/issue
Domestic: $89.99/yr (12 issues)
Canada/Mexico: $99.99/yr
all other countries $129.99/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. $15 all others

Tales from the List

String manipulation brainteaser

This installment of **Tales from the List** focuses on a couple of postings by long-time list member and frequent provider of expert advice, I-Lin Kuo. Someone posted to the list stating that they had a variable containing a list, and they wanted to replace the last comma with the word “and”. Therefore, “Red, Green” would become “Red and Green” or if there are more than two records, then append the “and” onto the last portion instead of a comma: “Red, Blue, and Green”. I-Lin responded with an interesting and easy solution to implement: the “double reverse”.

By Simon Horwith

Each item in the list is separated from the other by its delimiter: a comma in this list. The challenge here is that we want to specifically manipulate the delimiter between the last and next-to-last list items, to change it from a comma to an “and”. Unfortunately, processing the delimiters between list items isn’t trivial, and there’s no automatic way to do this in CFML. The trick I-Lin suggested is to reverse the entire string that we’re treating as a list, so that the last comma (if any) becomes first.

Now we can simply treat that first comma as a string that can be replaced with the word “and”. Since we’ll be reversing it again to “put the string back to normal”, we’d want to use “dna” instead of “and”. The code looks like this:

```
<CFSET ColorList = "Blue,Red,Green" >
<CFSET ColorList = Reverse(ColorList)>
<CFSET ColorList = Replace(ColorList,","," dna ")>
<CFSET ColorList = Reverse(ColorList)>
<CFSET ColorList = Replace(.ColorList, ",", ", ",
"ALL")>
<CFOUTPUT>#colorlist#</CFOUTPUT>
```

The list in the first line might come from a ValueList or other list-manipulating function. Also notice that the two “replace” lines have extra spaces between their arguments to help them stand out here in print.

List members were all very impressed with this solution, which I-Lin then took a step further by suggesting yet another very cool use for this trick. As per I-Lin: “Often you want to truncate a long block of text, but if you just use Left (myText, 80), you may wind up truncating in the middle of a word.” She supplied the following variation on the double reverse for retrieving the first 80 characters *without* breaking a word in two:

```
<!--- assume Len(myText) > 80 --->
<cfset myText =
reverse(Left(myText,80))>
<cfset SpacePosition = Find(" ",myText)>
<cfset myText = Right(myText,80-SpacePosition)>
<cfset myText = reverse(myText)>
```

Very nice indeed. I-Lin then posed a little brainteaser: “How can you accomplish the same thing without using a double reverse or a cfloop?”

Timothy Heald replied:

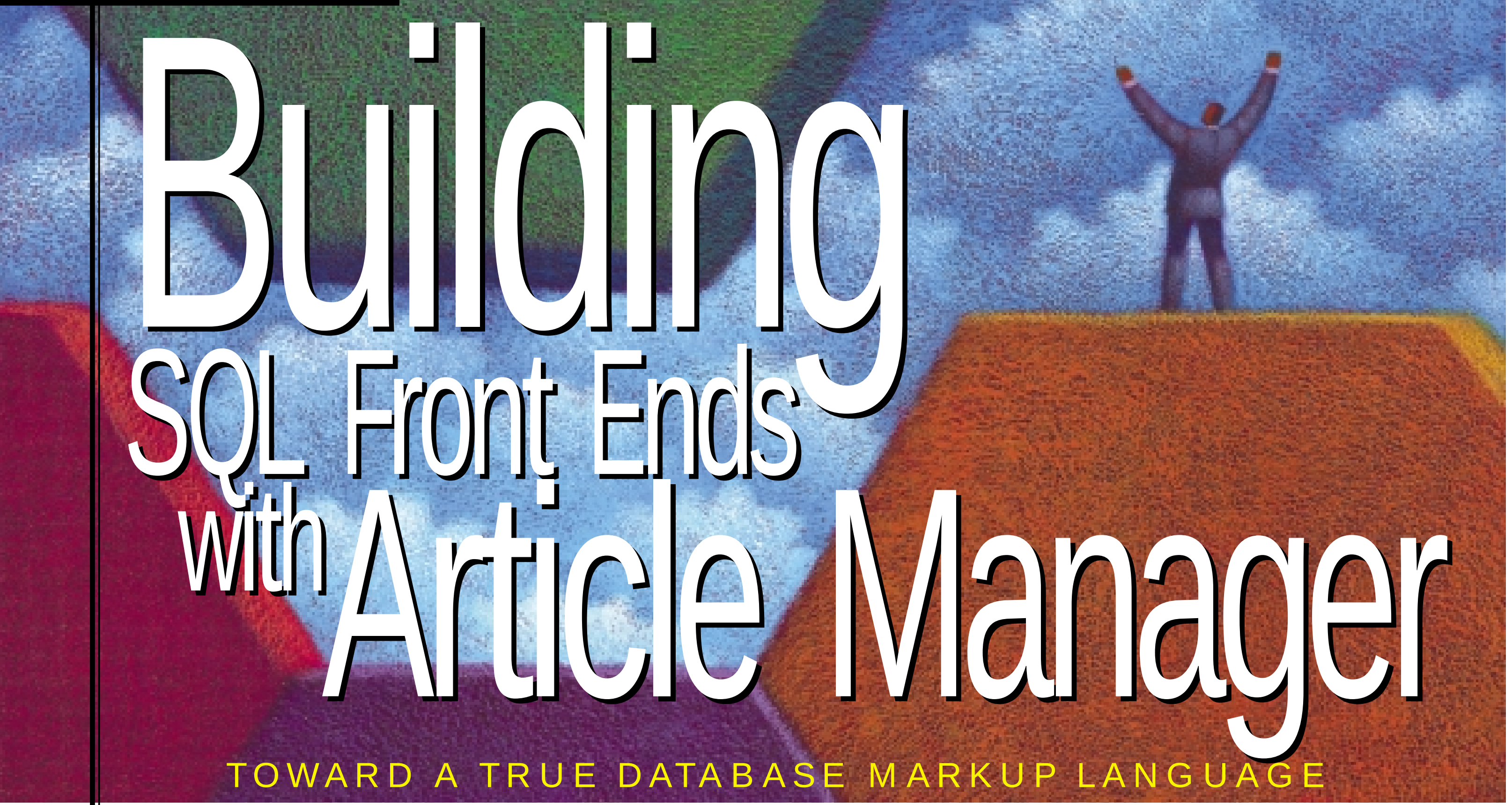
```
<cfset myText = left(myText,80)>
<cfset count =  len(myText) -
(len(listLast(myText, " ")) + 1)>
<cfset myText = left(myText,count)>
```

—continued on page 44

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



Building SQL Front Ends with Article Manager

TOWARD A TRUE DATABASE MARKUP LANGUAGE

Given the number of applications available for developing database front ends, it might seem strange that the most popular method of doing it is using HTML and an application server. It's become apparent, however, that the Web offers something in terms of simplicity and familiarity that users value more than the added functionality that dedicated packages can bring.

No doubt their bosses also value the cost savings, and it can be convenient for developers not to have to worry about installing dedicated packages or custom-built front ends.

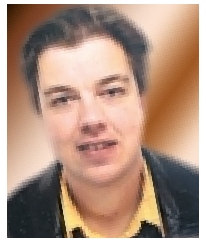
The problem, though, with using ColdFusion and HTML for building front ends is similar to the problem of using Visual Basic or raw Java to build dynamic Web sites – you're using a language primarily designed for something else to do a job that would be better served by a dedicated toolset. When building front ends, you can easily find yourself duplicating code – or at best, writing similar functions over and over again, and having to test and debug those functions each time.

This is where dedicated packages have advantages over Web-based solutions. First, they don't require you to write SQL insert, or update or delete code; second, perhaps more important, they reuse the same forms for adding, editing, and searching data. Packages like Access and FileMaker let you design a form once, then use it for each of these separate functions without any extra effort.

Even if you stick to CFUPDATE and CFINSERT, you still end up duplicating code to create the forms for different functions. Although you can use Studio/Dreamweaver "wizards" to create the code in the first place, it still needs testing, debugging, and support in the long term.

To solve this problem I started developing Article Manager (AM) – a set of custom tags to simplify the process of building front ends, and thus avoid the need for duplicating code. It's so named because it started life as a content management tool for online magazines, but it's evolved into a general-purpose front-end tool without any specific implementations in mind.

Although it's still a set of ColdFusion custom tags, the long-term goal of the project is to create a pure tag-based markup language for database front ends. In its latest version, v.3, AM is nearly that, and in its next version it will be. It's intended solely for the dull administration side of Web sites – the side that dare I say, is being a little neglected with the excitement of Flash MX. Behind every dazzling Web site there needs to be a solid, easy-to-use management system, and AM is designed to build it.



By Tom Peer

Building Article Manager Applications

AM applications are built from two main components – list pages (see Figure 1) and record detail pages (see Figure 2). We'll call detail pages record pages, because as we'll see they do more than simply display record details.

Creating a list page couldn't be much simpler. You need to specify, at minimum, a table and the fields to show. The only code you'd need to make a list page like the one shown is:

```
<cf_recordlist table="ARTICLES" fields="HEADLINE,PUBDATE,AUTHOR">
```

By default, 50 records at a time are shown, with next and previous links added when needed. You can override this parameter globally or on a per-list basis. Dates are automatically detected and formatted using LSDateFormat() and a mask that must be specified in application.dateformat even if you're using the default locale.

The default list page looks like the one shown in Figure 1. Specifying an attribute of TYPE="GRID" uses the CFGRID tag to avoid the need for a separate record page.

In the AM example application, the Categories table uses this feature. Usually, though, you'll want to have separate record pages, because as well as allowing you to edit and add records, they also allow for searching,which will be covered later.

Using TYPE="Query" returns a query for a customized list. Because this is such a common requirement, the "more, next" part of <cf_recordlist> is available in its own custom tag.

"TABLE" and "FIELDS" are the only required attributes for a record list, but usually you'll want to specify others. "SO" can be used to specify a sort order. "SHOW_ID" lists the record ids in the list; "VIEW" specifies a view to query rather than a table, although TABLE must be specified as well for advanced searches and deletions. An example is shown in Listing 1.

If your database doesn't support views, you can specify "joined" tables using SQL (e.g., ARTICLES LEFT JOIN CATEGORIES ON ARTICLES.CATEGORIES_ID = CATEGORIES. CATEGORIES_ID).

"CRITERIA" specifies search criteria; "SELECT" and "GROUP" can also be used to pass custom SQL. It can get quite complex, and begs the question why not just have a Query attribute? This is because AM makes extensive use of joins for its searches. Search results and criteria are stored within the database, and the SQL can become quite complex.

Record Pages

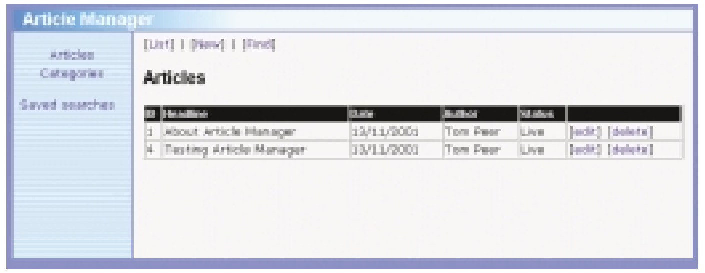
A basic record page looks like Figure 2, with each field listed one after the other in a two-column table. The tag that actually displays the fields is <CF_RECORDFIELD>, which works similarly to <CFINPUT> and <CFSELECT>, combining and extending the two tags.

The type of field is specified in the "TYPE" attribute, and the various options are listed in Table 1. The ones that you may not be expecting are "Combo" – a text field with a list underneath it that allows you to select an option or type in a new value, and "Swapbox" – two multiple select lists that allow you to move values between them.

A basic record field tag requires only the field name:

```
<CF_RECORDFIELD FIELD="HEADLINE">
```

Usually you'll want to add a "LABEL", which, by default, is the field name with underscores changed to spaces. Any attributes available to CFINPUT or CFSELECT are also available, and the default "message" is more helpful. For text it says "please enter a value for field xxx"; for dates and numbers, the message is more specific, indicating the format required.

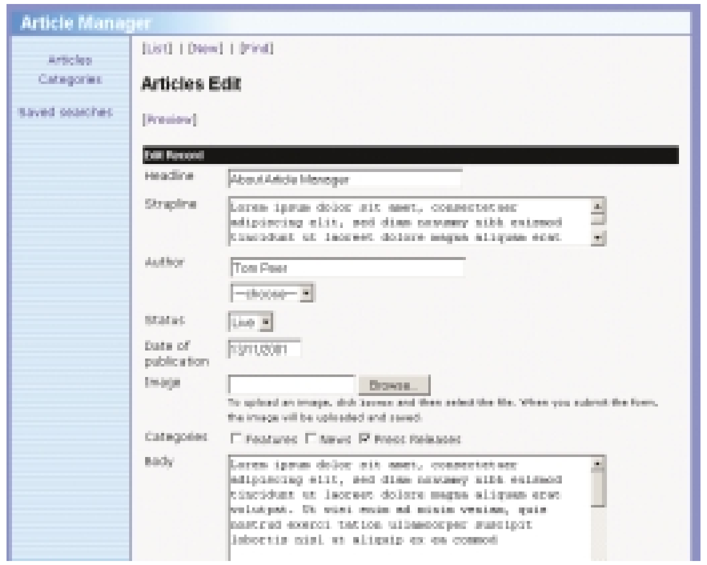


[List] | [New] | [Find]

Articles

ID	Headline	Date	Author	Status	
1	About Article Manager	13/11/2001	Tom Peer	Live	[edit] [delete]
4	Testing Article Manager	13/11/2001	Tom Peer	Live	[edit] [delete]

Figure 1: A basic Article Manager list page.



[List] | [New] | [Find]

Articles Edit

[Preview]

Edit Record

Headline: About Article Manager

Strapline: Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat

Author: Tom Peer
—choose—

Status: Live

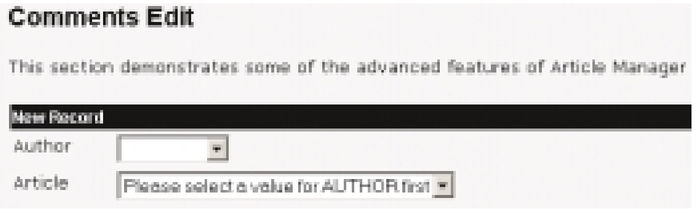
Date of publication: 13/11/2001

Image:
To upload an image, click browse and then select the file. When you submit the form, the image will be uploaded and saved.

Categories: □ Features □ News ☑ Press Releases

Body: Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo

Figure 2: A typical record page. The same page is used to add, edit, and search records.



Comments Edit

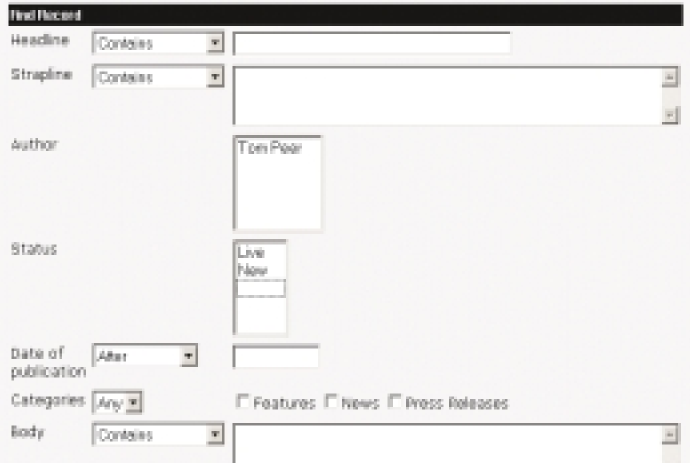
This section demonstrates some of the advanced features of Article Manager

New Record

Author:

Article: Please select a value for AUTHOR first

Figure 3: In the example application, the comments record page has a pair of related fields. The author field "controls" the headline field.



Find Record

Headline: Contains

Strapline: Contains

Author: Tom Peer

Status: Live
New

Date of publication: After

Categories: Any □ Features □ News □ Press Releases

Body: Contains

Figure 4: Article manager in Find mode.

The QUERY attribute is also available for radio and checkbox fields, together with DISPLAY and VALUE. A REQUIRED="No" attribute applied to a list will insert a blank option when a query is specified.

Using the CONTROLS_FIELD and RELATED_FIELD attributes (see Figure 3), you can link two select boxes so that the "slave" shows only values from a query where a RELATED_KEY field matches the selected value in the "master" list (this will make better sense if you view the example application).

There are also field types that allow easy storage of images and files. One difference between AM and some systems is that images and files aren't stored within the database; they're saved to the file system and the filename is recorded in the database. The directory to use is configured in the application scope.

Article Manager Field Types

TEXT	Simple text field [default- can be omitted].
RADIO	The same as <cfinput> with type="radio" but allows use of the query and list attributes as per <cfselect>
CHECKBOX	Similar to radio, extends <cfinput>. Also allows for many to many joins using the JOIN_TABLE attribute.
BOOLEAN	A single checkbox for Boolean (yes/no) field types [takes away the worry of the field not being defined when you use HTML checkboxes
INTEGER	Simple text field using CFINPUT with validate="integer"
NUMERIC	Simple text field using CFINPUT with validate="float"
DATE	A text field formatted using application.dateformat and with validate="Date" for US locale and validate="eurodate" for all other locales.
TEXTAREA	An HTML textarea.
IMAGE	File upload facility that stores filename in database and image in file system
FILE	As per image except without the image preview.
LIST	Almost identical to CFSELECT. Puts in a blank line when required="no" and also allows one list to control another (selecting an option in a master list will narrow the options in a slave list).
COMBO	Creates a select list of the existing distinct values in a field and displays them below a text field. Selecting a value pastes it into the field. [An explicit list can also be specified]
SWAPBOX	The brightest star in the AM firmament – use identical syntax to <CFSELECT> to create two multiple select boxes containing choices and selected options with buttons to move options between the two.

Table 1

AM has also been designed to allow easy adding of your own field types, perhaps using a Java applet like Webedit or a DHTML calendar control. Simply customize a dsp_fieldtype.cfm file and an act_parse_fieldtype.cfm file using the templates and "fieldtype" can be used as a value for the type attribute in CF_RECORDFIELD.

One of AM's strengths is the JOIN_TABLE attribute that you can use with checkbox and swapbox fields to store the values in a separate table – a "join" table – for building many-to-many relationships. The join table should contain just two fields – the record id and the id of the related value. In the AM example application, these are ARTICLES_ID and CATEGORIES_ID. A full discussion of this technique for creating many-to-many joins is beyond the scope of this article, but to my mind they're the litmus test of good database design. If you're unfamiliar with the concept, getting to know it would be time well spent.

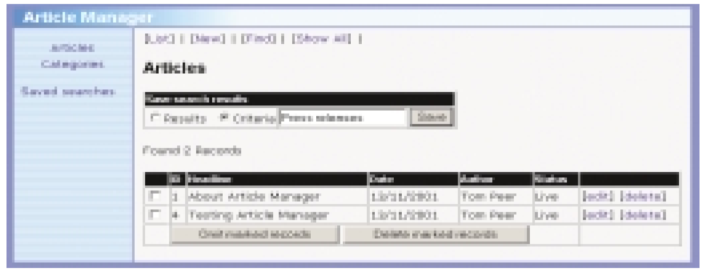
Two other useful attributes of CF_RECORDFIELD are DEFAULT – the value to use a new record (a value of "today" for a date field will enter the current date), and SAMEASLAST (specify true or false) – which will override the default by remembering the value from the last record edited and using that. It's useful when multiple records have to be entered at the same time.

Like dedicated packages, AM requires the fields to be specified only once. A record page has three "modes" – "new", "edit", and "find". "New" and "edit" show a blank form and the data for a specified record, respectively. AM takes care of all inserts and updates – there's no need for any SQL. This is all done by the <CF_RECORD> tag that goes above the <CF_RECORDFIELD> tags. The full code for the record page shown is in Listing 2. As shown in this listing, you have to add <cfform> tags and the <cf_am_submit> tag for the submit button, but this will change with the next release, leaving AM as almost a pure mark-up language.

When in "find" mode, the two-column table becomes three columns, and "qualifiers" are added for each type of field. All list boxes turn into multiple selects, and multiple fields have a qualifier of "all" or "any". It's a handy tool for editors and admin staff to quickly find records even if it won't replace hand-cranked SQL for report writing.

"Find" mode (see Figure 4) works differently than the other actions ("add", "edit") in that it posts the data to the list page for the table in question, and CF_RECORDLIST performs the find. Once in "find" mode, selecting "list" for the table in question will show only the "found set". The criteria of the search can be saved immediately after the search, and the results at any time hence (see Figure 5).

This is important because you can "omit" records from the found set, perform another search within the found set, or add the results to the found set – performing AND or OR searches in a slightly roundabout way. These saved "found" sets can be used for exports or reports (both easily done by using CF_RECORDLIST



Articles

Press releases

Found 2 Records

	ID	Headline	Date	Author	Status	
□	1	About Article Manager	12/11/2001	Tom Peer	Live	[edit] [delete]
□	4	Testing Article Manager	12/11/2001	Tom Peer	Live	[edit] [delete]

Figure 5: After a search, the results or the criteria can be saved for reuse.

with a type="query" that you can use as you wish). Currently you can't save multiple criteria sets – although I hope to change this soon.

To clear a found set, the user simply clicks "Show all", and the record list reverts to showing all records. This is equivalent to an MS Access "clear filter" instruction. Searches are associated with a user using the value of SESSION.CFID – not a particularly sophisticated or robust solution, but one that is easily replaced with your own chosen mechanism (e.g., a user id [table of USERS?] or perhaps an authenticated user name).

Implementing Article Manager

AM currently works with Access, SQL Server, and MySQL (an improvement on earlier versions which used sub selects and only ran on Microsoft databases). The SQL is very standard, and there's no reason Openbase or Interbase shouldn't work. I just haven't gotten around to installing or testing them.

For AM to work, there are some restrictions on the design of your database and some tables that need to be added to your solution. These are for storing the search results and criteria. In addition, all tables have to have a numeric key set to increment and named TABLE_ID. Ideally this should be the primary key, but it doesn't have to be if you already have one. Every table also has to have a field called "TEMP_ID" of at least 35 characters, which is used when records are inserted to store a UUID and then used to return the new ID. This provides a cross-product solution to the problem of returning new IDs without relying on the Max(ID) method.

The biggest obstacle to implementing the system isn't usually these details, but the uniform look the system applies to every page. Without any of the bells and whistles of dedicated packages or any of the hand-finishing that often gets applied to Web-based solutions, AM applications can seem a little bare.

The first comments people usually make when they start using AM are along the lines of "I'd make that more obvious," or "it wasn't clear what had happened." The user experience, though, isn't something I'm in any hurry to change. I've found that you can waste lifetimes making things "user-friendly" when all they really needed was a few hours' familiarity.

Any process-specific additions take away from the simplicity that is key to successful programming. If I can keep a system strictly to standard AM lists and record pages, I know that it's going to be easy to support and maintain in the long term. Every tweak and modification to satisfy a first-look knee-jerk reaction is going to add manifold to the complexity of the project.

A True Markup Language?

A goal I set myself when I'm building content-management systems is that any bit of information on a Web site should be changeable with a single edit using a consistent interface. That's what I'm aiming for with Article Manager. A single change to a field tag should be enough to take care of all inserts, updates, and searches. The dream's still a long way off, but using AM it's certainly closer than hand cranking SQL and HTML forms.

In its next version, AM will become a genuine markup language, using XML in place of custom tags. While this is primarily to allow the introduction of new features (such as sub forms for related data), it will also pave the way for a Flash MX interpreter or even dedicated interpreters using systems like Delphi.

With the arrival of Flash MX and the growth in mobile browsers, the need to separate data and presentation has never been more apparent, and a database markup language would provide a level of abstraction that's missing from many of the early examples of Flash MX interfaces. HTML and ColdFusion have shown just how powerful simple, tag-based languages can be. It's vital that we don't lose that simplicity with the move to rich client interfaces, and with a database markup language, there's no reason why we should.

Download Now

Download Article Manager, the example application, and documentation at www.articlemanager.com

About the Author

Tom Peer has been in electronic publishing of one sort or another for eight years, including a stint as manager of New Scientist Online *(www.newscientist.com). He specializes in taking printed publications online and has recently completed the online edition of* The World Handbook of Stock Exchanges *(www.exchange-handbook.com).*

tom@tompeerconsulting.com

Listing 1: Example record list page

```
<cf_recordlist table="ARTICLES" view="ARTICLES_VIEW"
   relatedtables="ARTICLES_CATEGORIES_JOIN"
   show_id="1"
   so="PUBDATE DESC"
   fields="HEADLINE,PUBDATE,AUTHOR,STATUS"
   labels="Headline,Date,Author,Status">
```

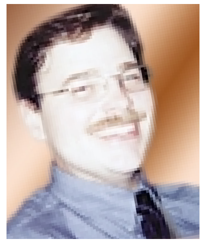
Listing 2: Example record page

```
<cf_am_menu section="ARTICLES">
<cf_record table="ARTICLES">
<cfform action="#request.form_action#" name="main" enctype="multi-
part/form-data">
<cf_recordfield field="HEADLINE" label="Headline" >
<cf_recordfield field="STRAPLINE" label="Strapline" type="TEXTAREA"
   rows="3">
<cf_recordfield field="AUTHOR" label="Author" type="COMBO"
   sameaslast>
<cf_recordfield field="ARTICLES_STATUS_ID" label="Status"
   type="LIST" required="YES" query="application.ARTICLES_STATUS"
   value="ARTICLES_STATUS_id" display="STATUS">
<cf_recordfield field="PUBDATE" label="Date of publication"
   sameaslast type="DATE" default="today" >
<cf_recordfield field="IMAGE" label="Image" type="IMAGE">
<cf_recordfield field="CATEGORIES" label="Categories" type="check-
box" sameaslast join_table="ARTICLES_CATEGORIES_JOIN" query="appli-
cation.categories"
   value="categories_id" display="category">
<cf_recordfield field="Body" type="TEXTAREA" rows="8">
<cf_am_submit>
</cfform>
```

Studio MX by Macromedia

Take the dread out of building Web pages

Dreamweaver has been upgraded to include many features familiar to users of CF Studio, and includes other crucial Web-design applications, specifically Fireworks, Flash, and Freehand.



Reviewed by Carey Lilly

Okay, let's get one thing straight: I don't like WYSIWYG editors for coding. *Do not.*

When I said that to a Macromedia rep on the phone recently, I could feel the wince all the way across the country. "But," I said, after a few heartbeats, "I may change my mind on that."

Just so you don't think I was being unfair, I wasn't making a blanket statement. I first used visual Web-design tools with the HTML add-on for Word (it was horrible). I played with others whose names I have unrepentantly forgotten. I first played with Dreamweaver when UltraDev came out. I felt I had no reason to want a Web-design–friendly environment since I spend most of my time writing code. I thought that UltraDev was nice enough, but I just couldn't write CF apps using drop-down menus and property windows (and seeing a little yellow tab in place of lines and lines of nice neat code also kind of bugged me).

When it came to quickly sketching out a design, laying out tables, and so on, Dreamweaver was nice, but I write CF code by hand, and visual editing seems to just slow me down. I also found that I relied a great deal on Studio's "tag hints" feature, which was missing in UltraDev. And finally – and you may laugh at me for this – I just couldn't stand all those windows floating all over the place! The short of it was, after playing with UltraDev, I dropped back to good old CF Studio. That was until UPS dropped a package off one fine summer day....

Note to readers

Coming as I did from CF Studio 4.5 to Dreamweaver MX, I may point out features that were available in previous versions of CF Studio/Dreamweaver. My apologies if anything I mention is not a surprise to you. – Carey

What Is Studio MX?

Since Macromedia acquired Allaire, Dreamweaver has been upgraded to include many features familiar to users of CF Studio. Studio MX is a Web development package, not a new version of CF Studio. The package also includes other crucial Web-design applications, specifically Fireworks, Flash, and Freehand. This review will focus mostly on Dreamweaver, but I'd like to mention the other products briefly, as you may use them more than you think:

- ***Fireworks:*** As a graphics editor, Fireworks does not compare to a high-end product like Photoshop. Of course, it doesn't have the price tag, either. Fireworks is well-suited as a Web-design tool. If you're not familiar with this product, it allows you to combine graphic and text elements into a whole (a Web site front page, for instance). Certain images can be designated as hyperlinks, and the image can be "sliced" into smaller elements. The completed image can then be exported as an HTML page, with all images properly set into tables.

  Fireworks basically does the graphics-based stuff that HTML editors don't handle (or at least the stuff that you would have to spend hours fiddling with manually in most editors). I've used an older version of Fireworks for a year or two. This new one didn't seem to add anything earth-shattering, except that it now supports the Macromedia Extension Manager, allowing you to more easily use material from the Macromedia Exchange. It can open, edit, and export Photoshop files, too.
- ***Flash:*** I spent some time trying to wrap my brain around Flash. This is definitely something that will take more than a few weeks to master, so I'm not going to try to write up anything exhaustive about this app. Essentially, Flash MX allows you to create a richer end-user experience. Since I've seen everything from shopping cart applications to games written in Flash, I'll let you decide whether it's something you want to learn or not.
- ***Freehand:*** Okay, I've loved Freehand since version 3 or something. I used it in my desktop publishing days on an old Mac, and it was a critical tool. Once I made the jump into Web applications, I lost track of it. Finding version 10 on my MX CD was like finding an old friend in an unexpected place. Freehand does all of the other little graphic tricks that Fireworks doesn't handle (creating vector-based graphics, for instance). I suppose Freehand could deserve its own review article. Suffice it to say that Freehand can work with Flash and Fireworks, giving you a pretty complete set of tools for Web design.

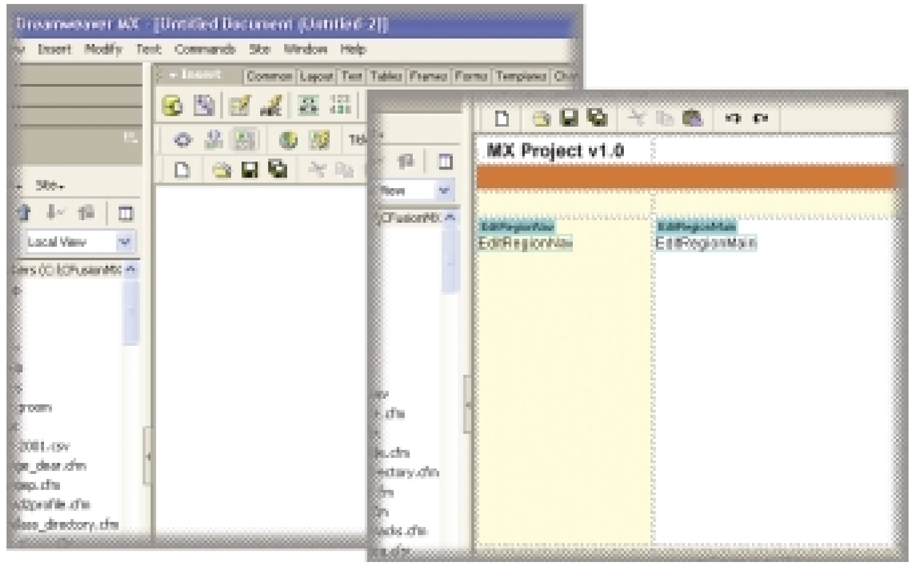


Figure 1: Visual Web design: from zero to Web page in about 6 clicks (okay, maybe 10)

(Drumroll please...) Dreamweaver

Dreamweaver MX is actually two products that are now fused into one: Allaire's Homesite/Studio product, used primarily by coders and Web applications developers, and Dreamweaver UltraDev, geared more toward Web designers needing visual tools. Dreamweaver itself was primarily a visual Web design application until the UltraDev version came along in 2000 to add tools for ASP, JSP, and CF. I've been using Homesite/CF Studio since before it was an Allaire product, and I have always appreciated that it's CF-centric.

Now I have to tell you, I clicked on the Dreamweaver icon with a bad attitude. Not Macromedia's fault, mind you. When I first tried it, I just didn't need Dreamweaver; a colleague was trying to get me to switch, and I was too busy. On top of that, I'd had the dubious experience of using visual Web tools in the past, and most of them were laughable if not downright travesties.

My attitude changed (just a tad) when Dreamweaver asked me if I'd prefer the MX workspace (with Homesite/coder style) or the DW4 workspace. Now, I've said that one thing that bugged me about Dreamweaver was all those windows all over the place. All those layered windows and floating/docking toolbars just made me uncomfortable. Maybe it doesn't bother you, but I can't concentrate in that environment.

It was a relief to see that Dreamweaver MX includes a familiar interface. The Homesite/coder-style interface feels a lot like CF Studio; it's closed in and divided into nice, neat little boxes, just as I like. If you like it the old way, choose the Dreamweaver 4 style.

I started by working on a little personal project that's been on my mind. First I needed a page (one with simple navigation), then I needed a fairly straightforward form, and then I needed to insert the form into a data source. Finally, I needed to display the contents of the database. This is all bread-and-butter CF code–not complicated. I thought it was a good test for an unfamiliar product.

As I was already in the design view, I decided to first create the tables that would form the structure of the page. Click the "insert table" icon, so many rows, so many columns, width, border. Resize the cells a little... voilá! Okay, so now I have the basic structure of my page, with a title row, a nav column, a big cell for content, and a footer bar. Add some styles (I really like creating CSS styles in DW. Once you know where stuff is, it's really simple). Add some nav hyperlinks, create an editable region for content, and then save the whole thing as a template (see Figure 1). Easy.

If you're not familiar with how Dreamweaver works, then you should know that templates will probably save your life when you have a deadline of one hour ago and not one page to show for it. Templates allow you to save blocks of common code (like the basic structure of a Web site). For each new page you create, all you have to do is add the specific content for that page (you mark the content area as an "editable region"), and then save it. With Dreamweaver, you can slap a site together in almost no time at all, and it'll look pretty good, too.

Creating my data input form in visual mode was just as easy. I created another table, inserted a form tag, added form labels, and then started dropping in form elements. This will be nothing new to you CF Studio users. Dreamweaver has a similar "form" toolbar. You click an icon and it drops the form element at the cursor. The "properties" window (yes, the very one that bugged me when I first used DW) allows you to modify the parameters of any element. Maybe once I had mellowed to DW, I just found the properties window to be no different than coding it in text. Or maybe using WYSIWYG to create forms can be fun! Either way, I had the basic stuff done quickly using the visual design tools.

Here's another neat toy I found: Dreamweaver MX includes an "Application" toolbar with a "record insertion form." Essentially, you supply it with a data source and table, it asks which fields you would like to populate, and what their labels should be, then codes the form and the insert query into your page (see Figure 2). This is one of those "pick and shovel" tasks that I almost always dread when building an application for the Web. Okay, I'll admit that at this point, my attitude had gone 180 degrees.

The code Dreamweaver builds is nice, but I wanted to tweak it a little. This is when I switched into code view. Dreamweaver has three views: design view, code-and-design (a split window), and code view. Now here's what really bugged me about UltraDev: despite all the CF tools, UltraDev lacked one feature that made me stick with Homesite/CF Studio for years... tag hints.

Very simply, you type in a tag – say, CFOUTPUT – and a list of available parameters pops up. I feel that it improved my productivity, but UltraDev didn't have it. With Dreamweaver MX, you get that, plus the added bonus of hints for function parameters. That was a pleasant surprise. Dreamweaver's hints also include event handlers like onClick for HTML elements.

Now I started hand-coding the results page. I was only coding this for a short while when I thought: If Dreamweaver will automate an insertion, why wouldn't it automate a simple SELECT query? I hunted around, and sure enough, not only will it build the search query, it will build the table and supply the code to "show next *n* records" to navigate the results! Another "pick and shovel" task had just been reduced to a few clicks. Once I figured out how to use these automated features, I almost told my fiancée to cancel her plans, 'cause I was going to marry Dreamweaver! (I am kidding!)

After playing with all my new toys, I decided to give MX a spin on some of my older Web apps. I'm sure you have one or two. You know: the big, nasty, ugly legacy stuff that you maintain mostly because it still works, but that you'd love to just tear down and rewrite some year. I was disappointed that Dreamweaver started to really slow down when I started opening, searching, and saving big files (and not really that big: some were only 500 lines).

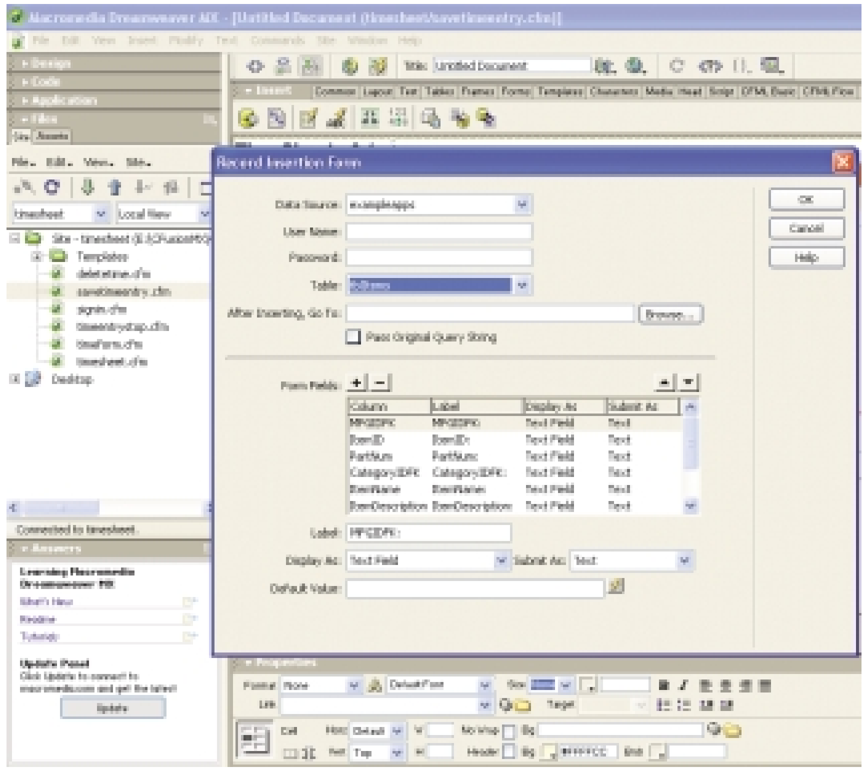
The more files you open, the slower it responds. This got tedious since some of this app's pages run to 1000 or 1500 lines. These days, my apps tend to use lots of smaller files, but if your .cfm files tend to be big, you'll probably want to break them up before you edit them in MX.

Conclusion

Increasingly, Web sites are required to be much more than static design. Even the added dynamic elements provided by ColdFusion are not enough for today's evolving, content-rich sites. Whether you're a Web designer or an application developer, you've probably needed to cross the line between the two. In my experience, my ColdFusion application is as likely to have Javascript and DHTML elements as it is to have HTML forms. XML, JSP, PHP, and other technologies are more commonly found working together in today's professional Web sites.

Dreamweaver is meant to be used for all of these purposes. An application with separate teams for design, user interface, rich content, and programming can all work in the same environment, and – where needed – cross the line into other territory without needing to learn new tools.

Studio users take note: there is a learning curve involved in moving over to Dreamweaver. It's a lot *like* CF Studio, but it is still quite a different application. Don't worry. The confusion will dissipate.



Record Insertion Form

Data Source: exampleapps

User Name:

Password:

Table: tblItems

After Inserting, Go To: Browse...

□ Pass Original Query String

Form Fields: + −

Column	Label	Display As	Submit As
MFGIDFK	MFGIDFK:	Text Field	Text
ItemID	ItemID:	Text Field	Text
PartNum	PartNum:	Text Field	Text
CategoryIDFK	CategoryIDFK:	Text Field	Text
ItemName	ItemName:	Text Field	Text
ItemDescription	ItemDescription:	Text Field	Text

Label: MFGIDFK:

Display As: Text Field Submit As: Text

Default Value:

Figure 2: Coding data-entry forms just got a little easier. Automatically create a form for fields in your table and the insert query for the new record, all with a few clicks.

Installation

I ran Studio MX on a Windows XP Pro system. I used a machine with an 800MHz processor, but scads of RAM (512 MB, to be exact). Installation was nice and quick. The CD includes an "install everything" function, or you can install one product at a time. I chose the "everything" mode and it all went on without a hitch. I used the developer's version of ColdFusion MX that was included, which runs on its own Web server.

Pricing

Dreamweaver Studio MX is $799 for the full commercial box or download version. Upgrade prices vary based on which prior versions you have. Dreamweaver, Fireworks, Flash (version 2 or later), and Freehand (version 7 or later) users can upgrade to Studio MX for $599 if you have one of those applications.

Dreamweaver UltraDev, CF Studio, CF UltraDev Studio, and JRun Studio users can upgrade for $399. The $399 upgrade price also applies if you have any two of Dreamweaver, Fireworks, Flash (version 2 or later) and Freehand (version 7 or later). An educational version is available for $199.

About the Author

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

carey@worldcontact.com

Vitals

Studio MX
Macromedia, Inc.
Address: 600 Townsend Street
San Francisco, CA 94103
Phone: 415 252-2000
Fax: 415 626-0554
Web: www.macromedia.com
Test Environment: Windows XP Pro system, ColdFusion MX Developer's version

Data Entry ReFORMed

Web-page form nightmares are a thing of the past

HTML-based data entry is a pain, both for developers and for end users. There's no way to sugarcoat it, HTML forms as used today are pathetic. Sure, there are new form specifications on the way, but they are not yet in use, browsers don't support them, and they are still unproven entities. Luckily, there's an option available right now; it's called Macromedia Flash MX.

By Ben Forta

The Forms Quagmire

Are HTML forms really that bad? Consider the following, first, from a developer's perspective:

- HTML form controls are limited; you can't extend HTML with your own controls (simple things like date choosers or calendars).
- HTML form controls provide almost no automatic validation (sure, you can validate text post-input, but you can't prevent the user from typing invalid characters in the first place).
- There is no simple event model in form controls; basic form control that should be simple (things like graying out controls or making them available based on other selections) is anything but.
- And one of the most annoying of all, because browsers cannot make round-trips to servers within a page, it is impossible to do things like creating side-by-side list boxes whereby options in one change based on selections in another. (We've all tried this one at some point.)

To make up for these limitations, developers create multipart forms. These are not a feature of HTML, they are a hack born of necessity. Consider the following, this time from an end user's perspective:

- Server-side form field validation is time-consuming and extremely annoying. Forcing a user to hit "Back" to make corrections is not user friendly.
- Some browsers may even lose all entered data, requiring the user to reenter it all.
- Users will often submit forms multiple times, or hit Back and Refresh repeatedly, creating multiple records or executing multiple requests (which they'll then e-mail you about later).

And this is just the start of it. As a developer you've probably wasted countless hours having to accommodate for glaring inadequacies in HTML as it exists today.

Which brings us to Flash MX.

Why Flash-Based Forms?

I am not going to sell you on the virtues of Flash – that's another column, and one probably better suited to be written by the Flash experts among us. For now, suffice it to say that Flash is readily available, can be very lightweight, is extremely portable, and (this is the most important part) can be used for more than animation, movies, and annoying popup ads.

Not that I'm saying Flash should not be used for those things, it should be (well, maybe not for those popup ads that invade your screen, but for everything else). But Flash can also be used for building entire application clients or parts thereof, parts like forms.

So why don't you see more Flash-based forms out there? There are a couple of reasons for this. First, until recently, Flash was rather inaccessible to developers (those of us who actually create forms); the tool was geared far more toward designers. And second, creating forms meant starting from scratch – actually creating form controls.

Flash MX addresses both of these problems. It is far more developer friendly than any previous version of Flash, and it also comes with all sorts of form controls that you can literally just drag and drop into place.

That alone makes Flash MX an ideal solution for building forms, but it gets better. Flash MX and ColdFusion MX can talk to each other with minimal work on your part, which makes creating forms that interact with back-end processing very easy.

Still not convinced? I'll show you. Let's walk through an example together, in which we'll build a Flash-based form used to select an employee. For this example, we'll create the much-needed side-by-side list boxes. The left list box will display departments, and when a department is selected, the right list box will automatically update with the employees in that department allowing easy selection.

Note:

The examples here use the same databases included with ColdFusion MX, so you'll need those installed to follow along. And of course you'll need a copy of Flash MX too. If you don't have a copy, you can download an evaluation version from www.macromedia.com/downloads.

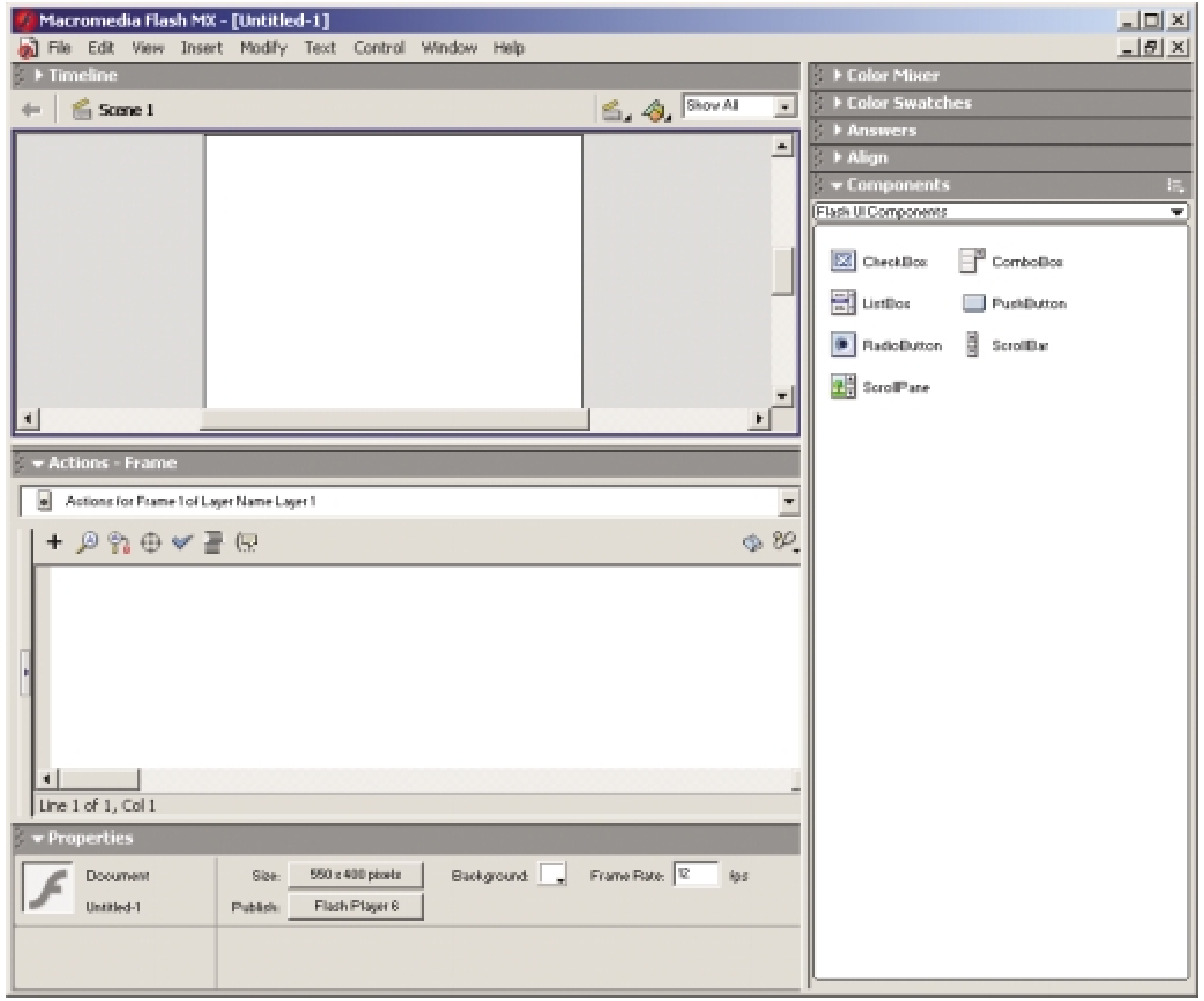


Figure 1: If you are not creating animation, simplify the Flash environment

The Back End

ColdFusion MX and Flash MX interact in several ways, but one of the simplest is via ColdFusion Components (explained in detail in ***CFDJ*** Volume 4, issues 6, 7, and 10). So we'll start with a simple CFC which contains two methods: GetDepartment returns a list of all departments, and GetEmployees returns all employees (either in a specific department or in all departments. Here is employees.cfc:

```
<!--- employees.cfc --->
<CFCOMPONENT>

<!---
GetDepartments method
Get all departments.
--->
<CFFUNCTION NAME="GetDepartments"
            ACCESS="remote"
            RETURNTYPE="query"
            OUTPUT="false">

   <!--- Get departments --->
   <CFQUERY NAME="Departments"
            DATASOURCE="exampleapps">
   SELECT *
   FROM tblDepartments
   ORDER BY DepartmentName
   </CFQUERY>

   <!--- Return query --->
   <CFRETURN Departments>

</CFFUNCTION>

<!---
GetEmployees method
Get all employees if no department
specified, otherwise gets employees
in specified department.
--->
<CFFUNCTION NAME="GetEmployees"
            ACCESS="remote"
            RETURNTYPE="query"
            OUTPUT="false">
   <!--- Optional department ID --->
   <CFARGUMENT NAME="DeptID"
               TYPE="uuid"
               REQUIRED="no">
```

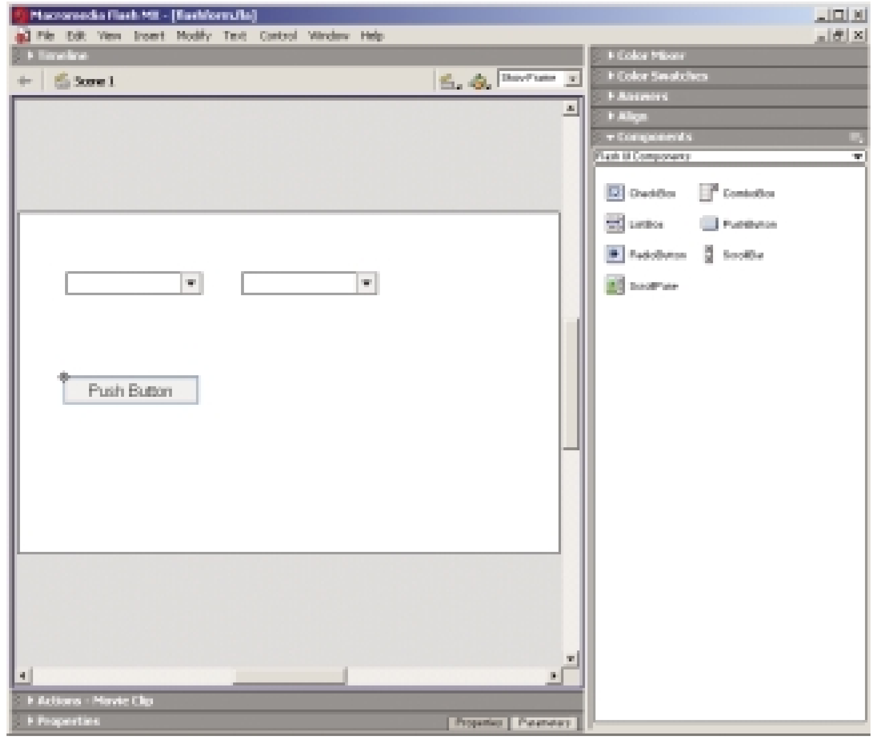
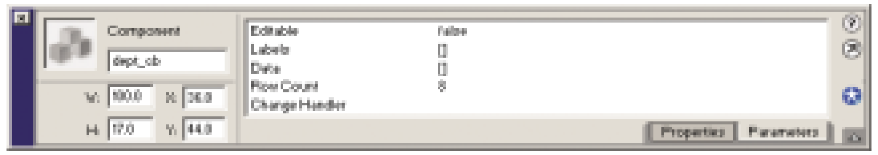


Figure 2: To use a component, simply drag it onto the stage



Component
dept_cb
W: 100.0 X: 36.0
H: 17.0 Y: 44.0

Editable false
Labels []
Data []
Row Count 8
Change Handler

Figure 3: Use the Properties window to set component properties

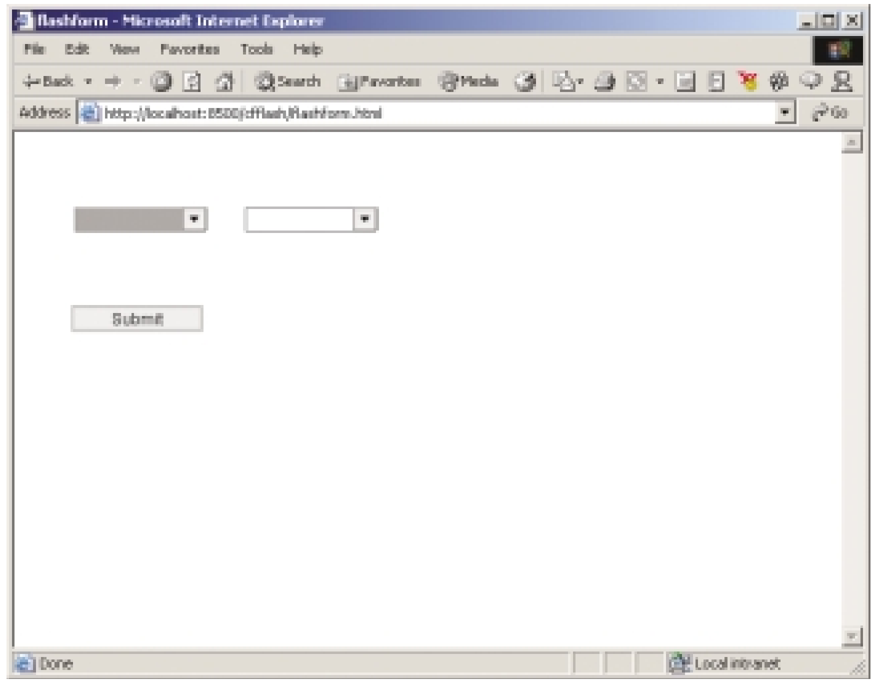

flashform - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://localhost:8500/cfflash/flashform.html
Go
Submit
Done
Local intranet

Figure 4: The ComboBox minus any data

```
   <!--- Get employees --->
   <CFQUERY NAME="Employees"
            DATASOURCE="exampleapps">
   SELECT *
   FROM tblEmployees
   <CFIF IsDefined("DeptID")>
      WHERE DeptIDFK = '#DeptID#'
   </CFIF>
   ORDER BY LastName, FirstName
   </CFQUERY>

   <!--- Return query --->
   <CFRETURN Employees>

</CFFUNCTION>

</CFCOMPONENT>
```

Create a simple ColdFusion script to test the component. The following code invokes GetDepartments and lists the returned department names:

```
<CFINVOKE COMPONENT="employees"
         METHOD="GetDepartments"
         RETURNVARIABLE="depts">

<UL>
<CFOUTPUT QUERY="depts">
   <LI>#DepartmentName#</LI>
</CFOUTPUT>
</UL>
```

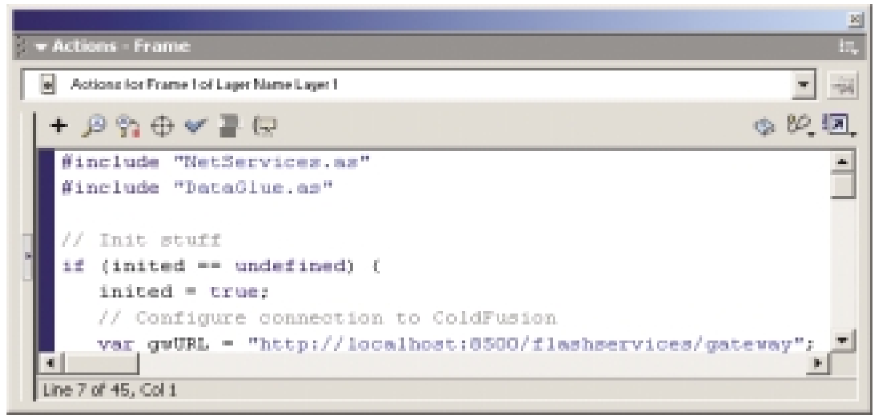
The following snippet tests the GetEmployees method, again displaying returned data in a list:

```
<CFINVOKE COMPONENT="employees"
         METHOD="GetEmployees"
         RETURNVARIABLE="emps">
<UL>
<CFOUTPUT QUERY="emps">
   <LI>#Lastname#, #FirstName#</LI>
</CFOUTPUT>
</UL>
```

Assuming that the code worked correctly, you now have a CFC capable of accessing and returning data – a component that may be used by ColdFusion (as in this test code) and also by client-side Flash.

Creating the Flash Movie

Now fire up Flash MX. We're going to keep the Flash really simple – we'll ignore frames, layers, animation, tweening, etc., so feel free to close the



```
#include "NetServices.as"
#include "DataGlue.as"

// Init stuff
if (inited == undefined) {
  inited = true;
  // Configure connection to ColdFusion
  var gwURL = "http://localhost:8500/flashservices/gateway";
```

Figure 5: ActionScript code is entered into the Actions window

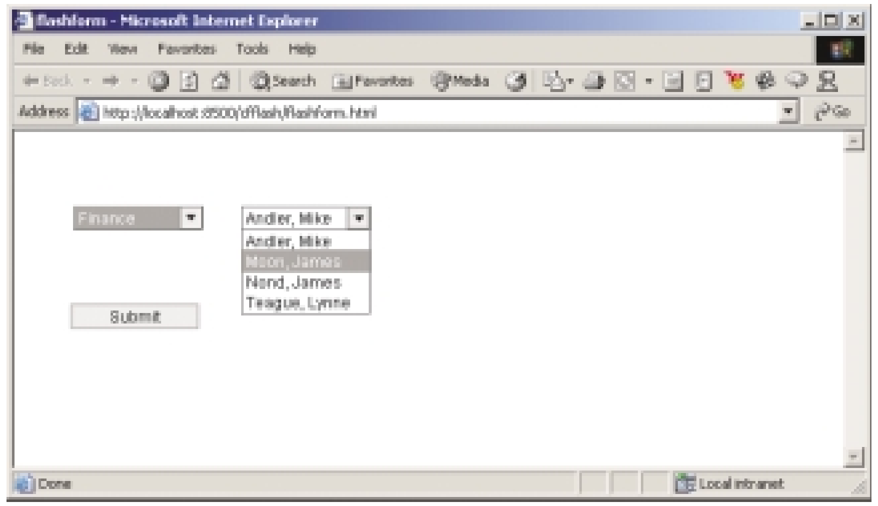

flashform - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8500/cflash/flashform.html
Go
Finance
Andler, Mike
Andler, Mike
Moon, James
Nond, James
Teague, Lynne
Submit
Done
Local intranet

Figure 6: The finished form

TimeLine window (as I routinely do). The only windows you'll need open are the Actions window, the Properties window, and the Components window (as seen in Figure 1).

Next, save the new (empty) Flash movie somewhere beneath your Web root (so that you can get to it).

The form we'll create has two drop-down list boxes (actually called ComboBoxes), so drag two ComboBoxes from the Components window (make sure the drop-down list at the top of the window reads "Flash UI Components") onto the stage. We'll also need a Submit button so drag a PushButton component onto the stage as well. You now have three ready-to-use components on your stage (as seen in Figure 2).

The next step is to name the instances of the components we just placed on the stage (so as to be able to programmatically script them). Make sure the Properties window is open, then click on each of the three components to access its properties:

- Click on the left ComboBox and specify dept_cb as the instance name (as seen in Figure 3).
- Click on the right ComboBox and specify dept_emp as the instance name.
- Click on the button and specify submit_btn as the instance name and Submit as the label.

Save the movie and then execute it. Either select Test Movie from the Control menu to run it right within Flash, or select Publish from the File menu to publish the movie so that it may be loaded in a Web browser. You now have a working (albeit rather useless) Flash-based form (as shown in Figure 4).

ColdFusion-Powered Flash

Now we need to get Flash to interact with the CFC. Open the Actions window (if it is not already open) and make sure the *Actions for Frame 1 of Layer Name Layer 1* option is selected from the drop-down list above it, and enter the following ActionScript code (as seen in Figure 5):

```
#include "NetServices.as"
#include "DataGlue.as"

// Init stuff
if (inited == undefined) {
   inited = true;
   // Configure connection to ColdFusion
   var gwURL =
"http://localhost:8500/flashservices/gate-
way";
   NetServices.setDefaultGatewayURL(gwURL);
   gw =
NetServices.createGatewayConnection();
   employeeService =
gw.getService("cfflash.employees", this);
   // Invoke the GetDepartments method
   employeeService.GetDepartments();
}
```

The first two lines of code are include statements (kind of like <CFINCLUDE> in CFML). The first includes support for Flash Remoting (the ability to interact with back-end systems like ColdFusion) and the second includes DataGlue, a series of functions that dramatically simplifies binding returned data to UI controls (as you will soon see).

The next block of code is initialization code that needs to be run only once, as it checks to see if it, itself, has been run already (checking for a variable that is defined within the if statement).

A variable is then declared to store the URL to the Flash Gateway on the ColdFusion MX server. The snippet shown here accesses a local ColdFusion MX using the integrated HTTP server, so the URL is localhost:8500. You'll need to change this to point to your own ColdFusion server as needed.

The next few lines of code create a gateway connection, and then a local reference to a remote CFC. The path here is specified as cfflash.employees which points to the employees.cfc file in the cfflash directory. You'll need to change that path as needed so that it points to your own employees.cfc file.

The last line of code actually calls the GetDepartments method. Flash knows that employeeService has been bound to a CFC on the other end of a connection, and so a call to employeeService.GetDepartments() is actually a call to the GetDepartments method in the service bound to employeeService. As soon as this last line of code is executed, Flash will have a query containing the list of departments.

If you were to run the Flash movie now, it would look no different than it did before. Data is being retrieved but nothing is being done with it (think of it as a <CFQUERY> without a <CFOUTPUT>). The next step is to tell Flash how to populate the appropriate ComboBox with the retrieved data.

When GetDepartments() is executed, results are returned. Flash automatically looks for a function named GetDepartments_Result(), and will execute it if present. So we'll need to create that function which will loop through the returned data, processing one row at a time and inserting them into the ComboBox. Fortunately you don't have to do that manually; that's where DataGlue comes into play. Here is the GetDepartments_Result() function:

```
// Process retrieved departments
function GetDepartments_Result(result) {
   // Populate dept_cb with results
 DataGlue.bindFormatStrings(dept_cb,result,
 "#DepartmentName#",
 "#DepartmentID#");
}
```

As you can see, GetDepartments_Result() is really simple. It calls DataGlue.bindFormatStrings() to bind the data in result to ComboBox dept_cb using #DepartmentName# as the label (display text) and #DepartmentID# as the value.

While we're at it, add a similar function to process results returned by GetEmployees(), as we'll be needing that next:

```
// Process retrieved employees
function GetEmployees_Result(result) {
   // Populate emp_cb with results
   DataGlue.bindFormatStrings(emp_cb,
   result,
   "#lastname#, #firstname#",
   "#employeeid#");
}
```

If you save and execute the Flash movie, you'll now see that the left ComboBox is being populated by data returned by ColdFusion. So far, so good. Now for the left ComboBox. What we need is a way to force the left ComboBox to refresh each time a selection is made in the right ComboBox. First create a new function named employeesRefresh(). Each time this function is called, it invokes the GetEmployees method in the employees.cfc (which in turn triggers the GetEmployees_Result() function just created). In order to get employees for the currently selected department, the getSelectedItem() ComboBox method is used to get the current department ID, which is passed as a parameter to GetEmployees(). Here is the employeesRefresh() function:

```
// dept_cb change handler
function employeesRefresh() {
   // Invoke GetEmployees method
employeeService.GetEmployees(dept_cb.getSel
ectedItem().data);
}
```

To actually get this employeesRefresh() function to be executed when a department selection is made, all we need to do is add the following code into the initialization code (you can put it right after the inited=true;):

```
   // Set change handler for first combobox
dept_cb.setChangeHandler("employeesRefresh"
);
```

This code sets employeesRefresh as the change handler for dept_cb (the left ComboBox), so that any time a change is made in dept_cb, employeesRefresh() will be executed and the employee list will be refreshed which in turn will force emp_cb to be updated.

Try out your new form. You should see both ComboBoxes populated (as seen in Figure 6) and whenever a department is selected, the employees in that department will be listed automatically. This process makes roundtrips to ColdFusion as needed, but it happens so quickly and efficiently (and without needing a page refresh) that you'd never even know.

There is one last step needed. To actually submit the form, you need form processing code, something that will be executed when the submit button is clicked. Here is the code:

```
// Process submit button pressed
submit_btn.onrelease = function () {
   EmployeeID =
emp_cb.getSelectedItem().data;
   getURL("action.cfm", "", "POST");
}
```

To post a form variable, you first need to create it, so the first line of code creates a variable named EmployeeID, and populates it with whatever the ID of the selected employee is (again using the getSelectedItem() method). Then getURL is used to actually submit the form fields; here the action page is named action.cfm but you can name yours whatever you'd like. For a simple test you can create an action.cfm that contains the following:

```
<CFOUTPUT>
EmployeeID: #EmployeeID#
</CFOUTPUT>
```

And there you have it. That's all the code you need to create side-by-side ComboBoxes that do exactly what is needed without requiring a single screen refresh. Just to be on the safe side, Listing 1 shows the complete ActionScript code listing.

Summary

Creating Flash-based forms is not complicated once you understand the basic steps. Using the form components that are provided removes the need for UI work (where most of us CFers would have gotten stuck) and requires minimal ActionScript. The best part of it is that the form we just created is highly usable and is only 11K in size (yes, believe it or not, Flash need not be big and bloated).

Give it a try (and be sure to go to the Developers Exchange at www.macromedia.com to download additional form controls, including a complete calendar, a tree control, and even a data grid). You'll find that Web-page form nightmares are a thing of the past.

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

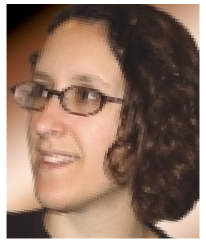
Listing 1

```
#include "NetServices.as"
#include "DataGlue.as"

// Init stuff
if (inited == undefined) {
   inited = true;
   // Set change handler for first combobox
   dept_cb.setChangeHandler("employeesRefresh");
   // Configure connection to ColdFusion
   var gwURL = "http://localhost:8500/flashservices/gateway";
   NetServices.setDefaultGatewayURL(gwURL);
   gw = NetServices.createGatewayConnection();
   employeeService = gw.getService("cfflash.employees", this);
   // Invoke the GetDepartments method
   employeeService.GetDepartments();
}

// dept_cb change handler
function employeesRefresh() {
   // Invoke GetEmployees method
   employeeService.GetEmployees(dept_cb.getSelectedItem().data);
}

// Process retrieved departments
function GetDepartments_Result(result) {
   // Populate dept_cb with results
   DataGlue.bindFormatStrings(dept_cb,result,"#DepartmentName#",
                                             "#DepartmentID#");
}

// Process retrieved employees
function GetEmployees_Result(result) {
   // Populate emp_cb with results
   DataGlue.bindFormatStrings(emp_cb,result,"#lastname#,
                        #firstname#","#employeeid#");
}

// Process submit button pressed
submit_btn.onrelease = function () {
   EmployeeID = emp_cb.getSelectedItem().data;
   getURL("action.cfm", "", "POST");
}
```

Welcome to the Wonderful World of Java: A Training Review

Do we all need to run out and learn Java?

By the end of the course, I had a good grasp of the various parts of the Java language, a better understanding of object-oriented programming, and a great appreciation for how powerful Java can be.



By Deanna Schneider

J2EE, Java, EJBs, JavaBeans, JSP… . The list of Java-related technologies can be mind-boggling. What's a ColdFusion developer to do? We know Java is important. Macromedia has built ColdFusion MX on a J2EE (Java 2 Enterprise Edition) platform, but, does that mean we all need to run out and learn Java? How about JSP? JRun? How will we use it in our day-to-day work? Will we use it and, if so, what flavor? Does a ColdFusion developer have the skills to learn this material?

These were the questions I was hoping to get answered as I headed for a series of two Macromedia-approved courses: "Fast Track to JSP" and "Java for Web Developers." Class offerings were difficult to come by, and my first scheduled class, in Washington, D.C., was canceled due to low enrollment. So, I was rerouted to the Fig Leaf training facilities at the Piedmont Center in Atlanta, Georgia.

These classes were sparsely attended, too, with only four students in each class, but this allowed for plenty of one-on-one time with the instructor, Alex Hearnz. Alex is an outside consultant who regularly teaches Java-related courses for Macromedia-certified companies as well as for other J2EE products, such as IBM's WebSphere.

The format of both classes would be familiar to anyone who's taken the "Fast Track to ColdFusion" class. The courses were divided into lectures, walkthroughs, and lab time; a student workbook filled with examples and instructions was provided. For an IDE, we used Macromedia's JRun Studio – familiar territory for a ColdFusion Studio user. (Although with the introduction of Dreamweaver MX and JRun 4.0, this will probably change.) And, although JRun 4.0 had already been released, we were still using the previous version.

"The Fast Track to JSP" class started the week. It was broken down into nine units:

1. Introduction
2. Setting up the Environment
3. Introducing JSPs
4. Reusing Code
5. Dynamically Generating Content
6. Building a Drill-Down Interface
7. Inserting Data
8. Application Partitioning Using JavaBeans
9. Session Management

Brushing Up on the Basics

For the Java novice, simply getting acquainted with all of the acronyms can be a challenge, and much of the first day was spent acquainting students with the terminology and the development environment. We also covered basic Web development architecture topics, such as the benefits of using dynamic versus static pages.

Some of the topics covered on the first day were things that I would have expected anyone coming to the class to already understand, based on the published prerequisites. Others, such as the process of establishing a database connection in the JRun server administrator, were specific and detailed enough to really catch my attention.

Once we got past the really basic stuff, we started focusing on the syntax of JSPs. JSP stands for JavaServer Pages, which are designed to make working with other Java objects easier. ColdFusion developers will find the syntax familiar. There is a series of tags that can be used to perform common functions. For example the following snippet of code works much like the familiar <cflocation> tag:

```
<jsp:forward page="nextpage.jsp"/>
```

Notice, however, that unlike ColdFusion, single line tags must still be closed in JSP, by using a trailing "/" such as you would do in XML. And JSP, along with all Java technologies, is case-sensitive. More than once we students cried, "Ugh! Stupid capital letter!"

Throughout the two days, we used our newfound skills to build a fictional online extreme sports company. Many of the common tasks Web developers must perform were covered: building a form and action page, querying a database and returning the results of that query back to the browser, and handling simple session state–management. But it was a cursory look at each of the topics.

Most experienced CF developers would be left with questions. Some of mine dealt with the same topics that are covered in the CF forums and discussion groups on a regular basis. For example, when outputting the results of a query, how do you group to a certain column? Unfortunately, the instructor was unable to provide an answer to this and several other questions.

At the end of the JSP class, I felt somewhat disappointed. I wanted to be able to replicate CF in JSP. I was hoping it might be that easy. But, I learned that JSP by itself is limited. To really harness the power of Java, learning how to write actual Java code is necessary. Luckily I was staying for the second class, "Java for Web Developers."

Exploring the Power of Java

For me, this was clearly the better of the two classes. The instructor seemed much more confident in his knowledge of pure Java. He seemed truly excited to seek answers to challenging questions. Also, we'd already gone through most of the stuff that would be redundant to experienced CF developers. So, we were ready to dive right in.

These three days were divided into 12 units:

1. Java Overview and Positioning
2. JavaServer Pages
3. Java Language Basics
4. Object-Oriented Programming
5. Object-Oriented Programming in Java
6. Developing and Using JavaBeans Components
7. Inheritance
8. Handling Exceptions
9. Working with Interfaces
10. Accessing the Database Using JDBC
11. Advanced Data Manipulation
12. Application Partitioning

After spending a little bit of time convincing us of the merits of Java and setting up a few necessary system variables, we got to the good stuff – exploring the almost infinite possibilities of Java. Like most Java classes, we began by using the command-line Java compiler utility to compile and run a simple "Hello World"–type Java class. But, we quickly moved on to using JRun Studio for all our development and compilation. JRun studio makes the process of compiling quick and painless.

One exercise was building an online loan service for the "Acme Credit Company." Once more, we covered many processes with which CF developers would be familiar. None is as easy as what we do in CF. For example, Listing 1 shows a simple CF example of querying a database and outputting that query. Listing 2 shows the corresponding code necessary to do the same thing in Java. At first glance, this looks awfully scary, and you might wonder why we'd want to bother with so much code. But, as we spent more time learning about theory, particularly object-oriented programming theory, it became clear that Java could accomplish some amazing things, such as true three-tiered applications.

Simply put, a three-tiered application is one in which database logic is separate from business logic, which is also separate from presentation logic. The beauty of a three-tiered application is that you can change the presentation layer for different environments without affecting the business or database layer. It's in this area that Java really becomes powerful.

Of course, creating a three-tiered application requires more thought and planning than "whipping out" a CF application in a day or two. Consequently, we spent time in the class learning a little about UML, or the Universal Modeling Language, and building UML diagrams of our Java components. This process is a little bit like creating an ER (Entity Relationship) diagram for a database, but because Java is object-oriented, you create UML objects that can represent classes or JavaBeans.

We also spent considerable time in this class in the online java docs, available at Sun's site at http://java.sun.com/j2se/1.4/docs/api/index.html, or on your local hard drive after downloading the Java Development Kit. If you learn Java, you will become very familiar with these pages. In three days I developed something of a love-hate relationship with them. They're wonderful because you know that everything you need to know is there. They're terrible because somehow you have to remember which package or class has the information you want.

> "Creating a three-tiered application requires more thought and planning than 'whipping out' a CF application in a day or two"

Overall, a Good Experience

By the end of the course, I had a good grasp of the various parts of the Java language, a better understanding of object-oriented programming, and a great appreciation for how powerful Java can be. Did I run right back to the office on Monday and starting writing Java code or JavaServer Pages? Well, no. (We don't have a J2EE app server, and we haven't upgraded to CFMX yet.) Will I use it when given the chance? You bet.

All in all, I would recommend both classes to others interested in developing a better understanding of Java and JSP. While the JSP class was a bit disappointing, I still learned some valuable information, and enjoyed the fact that I could be introduced to JSP in a familiar environment. The "Java for Web Developers" course completely met my expectations with one caveat – the course doesn't take the full three days, so if you're flying in like I did, plan accordingly.

One final note, the instructor mentioned that the future of both courses is in jeopardy, due to low interest. So, if you want to take them, call your Macromedia-certified training facility today and request they be put on the schedule.

About the Author

Deanna Schneider has worked as an interactive media developer for the University of Wisconsin Cooperative Extension since October 1999. Although she has no formal education in computer sciences, she has a successful career as UWEX-CES' lead CF developer. She has written approximately 100 CF applications to meet the needs of faculty and academic staff throughout Wisconsin.

deanna.schneider@ces.uwex.edu

Listing 1: Query a Database in CF

```
<cfquery name="getloan" datasource="mydatasource">
SELECT * FROM loan
</cfquery>
<cfoutput query="getloan">
<!---Output data here--->
</cfoutput>
```

Listing 2: Query a Database in Java

```
<%--Start your page by importing the various classes you'll need
later--%>
<%@ page import="javax.naming.*,javax.sql.*, java.sql.*"%>
<%
//Initialize all the variables you'll need
   String sql = "SELECT * FROM loan";
   Connection connection=null;
   Statement stmt=null;
   ResultSet rs=null;
   try {
//Create a database connection
         InitialContext ctx = new InitialContext();
         DataSource ds = (DataSource)
ctx.lookup("java:comp/env/jdbc/jloan");
         connection=ds.getConnection();
         stmt=connection.createStatement( );
         rs=stmt.executeQuery(sql);
         while (rs.next()) {
%>
<%--Output your data here --%>
<%
//Close your catch block and close the database connection
         }
   } catch (Exception e) {
         out.println(e.getMessage());
   } finally {
         rs.close();
         stmt.close();
         connection.close();
   }
%>
```

ODBC Escape Sequences

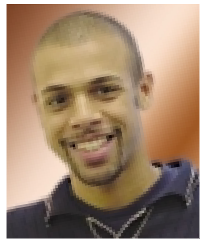
Enabling DBMS compatibility

Within the field of computer science, competing products often implement standards differently. Database management systems (DBMSs), with their different syntaxes for SQL queries, are no exception. How can we write queries within our applications that will be compatible with several DBMSs, especially when we need to use functionality that different DBMSs have implemented differently?

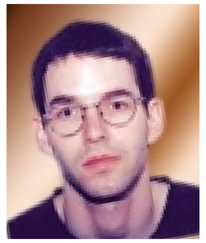
Recently, Technomedia Training, Inc., was awarded a contract to deliver our core product, a Web-based human resources management software application, to a new client. The client would be hosting the application, and the IT department's infrastructure and expertise were strictly Oracle-based. However, our application was designed to work with SQL Server 2000. Our challenge was to transform our Web application to make it compatible with an Oracle8*i* database.

The first step was to convert the database and all its objects from SQL Server 2000 to Oracle8*i*. (We employed some interesting conversion techniques, but they're a whole subject themselves and we won't cover them in this article.) Next, we discovered that many of the queries within the application didn't work anymore with the Oracle8*i* database because of differences in implementation of the SQL standard. For example, the following outer join query in Transact-SQL works with SQL Server 2000 but is incompatible with Oracle8*i*:

```
SELECT    EMP.ID, EMP.NAME, DEP.NAME
FROM      EMPLOYEES EMP
          LEFT OUTER JOIN DEPART-
          MENTS DEP ON
          DEP.DEPART-
          MENT_ID = EMP.ID
```



By Olivier Bridgeman



and David Levesque

In Oracle8*i*, the syntax is:

```
SELECT    EMP.ID,EMP.NAME.
          DEP.NAME
FROM      EMPLOYEES EMP, DEPART-
          MENTS DEP
WHERE     EMP.DEPARTMENT_ID =
          DEP.ID(+)
```

We looked at several alternatives for resolving this problem, and concluded that ODBC escape sequences are the best way to make our application's queries compatible with several DBMSs.

Technological Architecture

Our core product is a Web application that uses a standard, three-tier architecture. A ColdFusion Server works with an IIS Web server to run the application code. The ColdFusion Server uses an ODBC driver to connect to the database. Users' Web browsers call up pages from the ColdFusion/IIS servers using the HTTP protocol, and the application works with a SQL Server 2000 database – we need to make it compatible with Oracle8*i* (see Figure1).

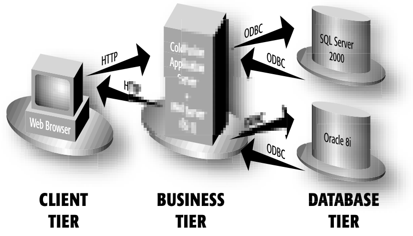


Figure 1: Three-tier architecture

Dealing with DBMS-Specific Queries

To deal with syntax differences among DBMSs, we considered three possible solutions. We could:

1. Change the application's queries to remove the DBMS-specific elements.
2. Put conditional statements around all queries with DBMS-specific terms and have both syntaxes present in the code. Then, depending on which DBMS the application is connected to, the appropriate syntax would be used.
3. Use ODBC escape sequences and let the ODBC driver map to the DBMS-specific syntax.

We'll examine these three possibilities next.

Simple SQL

The first option for dealing with DBMS-specific queries is to remove all terms that are DBMS-specific and to use only SQL syntax that is common to all DBMSs. For example, the SQL Server 2000 syntax for concatenating a first and a last name looks like this:

```
<cfquery name = "qsEmployees"
datasource="Sigal">
SELECT first_name + '  ' +  last_name AS
name
FROM employees
</cfquery>
```

Instead, the concatenation could be done outside of the query like this:

```
<cfquery name = "qsEmployees"
datasource="Sigal">
SELECT first_name, last_name, NULL AS name
FROM employees
</cfquery>
<cfloop query = "qsEmployees">
   <cfset tmp = QuerySetCell (qsEmployees,
"name",
   qsEmployees.first_name & "  " &

   qsEmployees.last_name,
   qsEmployees.CurrentRow)
</cfloop>
```

The advantage of this solution is that the query will work with all types of DBMSs and with all types of drivers, including non-ODBC drivers. However, we discarded this solution because of its disadvantages. If we limit our queries to a lowest-common-denominator SQL, we can no longer use some functionality that is essential to our application. There's no way to execute outer joins, and most date and time functions are excluded. All queries have to be relatively simple, which implies having to restructure certain queries or even sub-dividing them to make them more compatible. In addition, a lot of the processing is transferred from the database to the application code. This incurs a performance penalty, since the database server is optimized for many of the data operations, while the application server is not.

Different Versions of Incompatible Queries

The second option for making DBMS-specific queries compatible with several DBMSs is to create different versions of the query, each version containing DBMS-specific syntax. Using this method, our concatenation example would look like this:

```
<cfquery name = "qsEmployees"
datasource="Sigal">
<cfif application. DBType eq "ORA">
   SELECT  first_name || ' ' || last_name
AS name
   FROM    employees
<cfelse>
   SELECT  first_name + '  ' + last_name AS
name
   FROM    employees
</cfif>
</cfquery>
```

The advantage of this solution is that it's relatively easy to implement. A new, DBMS-specific version of each incompatible query is written and the application executes the appropriate version for the database it's connected to. However, doubling queries makes the application code bigger. Modifying and maintaining the code becomes more complex because changes to a query need to be repeated for each DBMS-specific version. Moreover, this solution doesn't scale with other DBMSs. If one day the need arose to make the application compatible with a DBMS other than Oracle or SQL Server, the whole process would need to be repeated. The code would become even more complex and difficult to modify and to maintain.

ODBC Escape Sequences

Using ODBC escape sequences for the DBMS-specific queries is the third option for making queries compatible with different DBMSs. ODBC escape sequences are a way of coding database functionalities that have different syntaxes in different DBMSs. As we have seen, the syntax for concatenations in Oracle8*i* is different from that used in SQL Server 2000, but we can use an escape sequence that represents the concatenation functionality provided by the DBMSs instead of the DBMS syntax. The concatenation example now looks like this:

```
<cfquery name = "qsEmployees"
datasource="Sigal">
SELECT    {fn concat (first_name, {fn con-
cat                         (' ',
last_name)})} AS name
FROM      employees
</cfquery>
```

ODBC escape sequences allow queries to be compatible with several DBMSs while giving access to the functionalities that have been implemented differently by different DBMSs. It's important to note that an escape sequence will only work with a DBMS if the DBMS supports the escape sequence functionality. Escape sequences are not implementations of database functionalities. Rather, they are an alternative way of writing a functionality, an alternative syntax that is translated by the ODBC driver into the DBMS syntax. It's possible to have one version of the queries using this technique. This solution also scales much better than the previous one because the queries are not only compatible with Oracle8*i* and SQL Server 2000, but also with any other DBMS that has an ODBC driver, in the measure that the DBMS supports the functionality represented by the escape sequences. The learning process is simplified because we don't need to learn the DBMS syntax of different DBMSs once we have learned the ODBC escape sequences. However, using ODBC escape sequences can make the code more complex because of the somewhat ornery syntax. You'll need to plan for a learning period for everyone to become comfortable with the new syntax and when to use it. Despite these mild difficulties, ODBC escape sequences are, in our opinion, the best way to deal with the different ways different DBMSs implement certain functionalities.

Using ODBC Escape Sequences

Let's look at how ODBC escape sequences are used. The functionalities that are implemented differently by different DBMSs are coded using a special syntax, called "escape sequences". Escape sequences have the following structure:

- They are placed within curly braces so the ODBC driver recognizes them.
- A code that represents the escape sequence type follows the opening curly brace.
- The escape sequence content is written.

The ODBC driver reads the escape sequence and translates it into the DBMS-specific syntax before sending the query to the database. ODBC escape sequences can be used for functions, outer joins, LIKE escape characters, GUIDs, intervals, procedure calls, and date, time, and timestamp literals. For example, an escape sequence function looks like this:

```
SELECT    {fn MonthName (date_membership)}
AS        month_membership
FROM      employees
```

An outer join escape sequence has this syntax:

```
SELECT    emp.id, ent.name
FROM      {oj employees emp
          LEFT OUTER JOIN entities ent ON
          emp.entity_id = ent.id}
```

A date literal has this syntax:

```
SELECT    COUNT(id) AS num
FROM      employees
WHERE     date_membership > {d '2003-01-
01'}
```

Challenges

When we started implementing and testing ODBC escape sequences in our application's queries, we discovered that not all ODBC drivers fully implement the ODBC standard. Some of the functions in the ODBC escape sequence specification were not implemented by our Oracle ODBC driver. For example, we weren't able to use the TimeStampDifference() function with the Oracle ODBC driver. In such cases, we used one of the other two alternatives, conditional statements around different versions of the query, or simple SQL with more processing outside the query.

Another concern was that by using ODBC escape sequences, we were limiting our database connectivity to ODBC drivers. Our company also develops Java objects to extend the functionality of ColdFusion, and we questioned whether the ODBC escape sequences were a good long-term solution, as Java objects use the JDBC protocol. Research revealed that certain JDBC drivers, known as ODBC bridges, use the ODBC protocol. Also, the JDBC protocol specification includes an escape syntax that is very similar to the ODBC escape sequences. Testing with a pure JDBC driver showed that it was compliant with several escape sequences. We strongly recommend testing with the JDBC driver you're planning to use.

Conclusion

Although the process of converting an application's queries to be compatible with a new DBMS can be lengthy, using ODBC escape sequences ensures that most of the work will be done only once. It also preserves the application's performance by allowing access to the database functionality, which has different syntaxes in different DBMSs.

References

- *ODBC escape sequences:* http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcescape_sequences_in_odbc.asp
- *JDBC API:* http://java.sun.com/products/jdbc/download.html
- *ColdFusion JRun JDBC driversdocumentation:* www.macromedia.com/v1/documents/jr31/jdbcref.pdf

About the Authors

Olivier Bridgeman is a Web programmer for Technomedia Training, Inc., maker of the industry-leading HR Web solution SIGAL. He has been developing Web applications for three years, the past two years using ColdFusion.

David Levesque is a Web programmer for Technomedia Training, Inc., and has been working with ColdFusion for a year. He also worked as an Oracle developer for two years.

olivier.bridgeman@technomedia.ca
david.levesque@technomedia.ca

Inc. 500
2002
WINNER
Inc. 500
2001
WINNER
Inc. 500
1999
WINNER
Deloitte & Touche Technology Fast 500
2002
WINNER
Deloitte & Touche Technology Fast 50
2002
WINNER
Deloitte & Touche Technology Fast 50
2001
WINNER

And the Winners Are...

Announcing the 2002 ***CFDJ*** Readers' Choice Awards Results

ColdFusion Developer's Journal Readers' Choice Awards program identifies and recognizes excellence in the solutions and services provided by the top ColdFusion vendors in the market. Each year ***CFDJ*** conducts a poll to select the best ColdFusion products in 12 different award categories. More than 5,000 ***CFDJ*** readers cast their votes to select the best ColdFusion products of the year from a record 158 nominations. The polls opened on August 26, 2002 and ran through October 1, 2002. Here are the winners and top three finalists in each of this year's categories.

Best Content Management Tool

Winner...
CommonSpot 3.0
CommonSpot Content Server 3.0, the flagship offering of PaperThin, Inc., is the leading ColdFusion content management solution. Organizations of all sizes worldwide depend on CommonSpot's award-winning technology to manage their web sites efficiently and effectively. As a full-featured, completely browser-based, out-of- the box Web publishing and dynamic content management solution, CommonSpot Content Server lets content producers easily participate in the efficient creation, management and delivery of Web content, without the help of a Webmaster and without having to know HTML.
Company: PaperThin
Web: www.paperthin.com

Winner...
NQcontent V2
NQcontent is the Web Content Management software of choice for ColdFusion developers. Its open and extensible architecture makes it possible for developers to customize its out-of-the-box functionality on a per project basis and implement custom data and content driven applications in record time.
Company: Netquest
Web: www.nqcontent.com

🟥 Winner 🟪 First Runner Up
🟩 Second Runner Up 🟧 Third Runner Up

First Runner Up...
ActivEdit
Company: CFDev
Web: www.cfdev.com/activedit2/demo

Second Runner Up...
RemoteSite Oasis 3.5
Company: RemoteSite Technologies
Web: www.remotesite.com

Best Book

Winner...
Advanced ColdFusion MX Application Development
The third edition of Ben Forta's Advanced CF book has won in this category for the last two years.
Company: Ben Forta
Web: www.forta.com/books/0321127102/

First Runner Up...
ColdFusion MX Web Application Construction
Company: Ben Forta
Web: www.forta.com

Second Runner Up...
Programming ColdFusion
Company: O'Reilly & Associates
Web: www.oreilly.com/catalog/coldfusion/

Third Runner Up...
Mastering ColdFusion
Company: Sybex
Web: www.sybex.com

Best Web Service

Winner...
House of Fusion Mailing Lists
The popular House of Fusion mailing lists include CF-Talk, CF-Community, JRun-Talk, and many others.
Company: House of Fusion
Web: www.houseoffusion.com/cf_lists

First Runner Up...
ActivEdit
Company: CFDev
Web: www.cfdev.com/xml/soap/google/

Second Runner Up...
Fonix SpeakThis
Company: Fonix
Web: www.speakthis.com

Third Runner Up...
BillingOrchard
Company: BillingOrchard
Web: www.billingorchard.com

Best Consulting Service

Winner...
TeraTech Consulting
TeraTech has been offering custom programming services since 1989. Their staff of analysts and programmers is experienced at understanding client needs and managing projects.
Company: TeraTech
Web: www.teratech.com/services/cfm

First Runner Up...
Fig Leaf Software Consulting Services
Company: Fig Leaf Software
Web: www.figleaf.com

Second Runner Up...
RemoteSite Oasis 3.5
Company: RemoteSite Technologies
Web: www.remotesite.com

Third Runner Up...
Biznet Web Design
Company: Biznet (Ireland)
Web: www.biznet-solutions.ie

Best Design Service

Winner...
Fig Leaf Software Creative Services
Fig Leaf Software's Creative Services Division creates intuitive and elegant designs based on research into specific users and audiences.
Company: Fig Leaf Software
Web: www.figleaf.com

First Runner Up...
X-Axi Media Creative Services
Company: X-Axi Media
Web: www.x-axi.com

Second Runner Up...
Biznet Web Design
Company: Biznet (Ireland)
Web: www.biznet-solutions.ie

Third Runner Up...
Oleani Technologies
Company: Oleani Technologies
Web: www.oleanistudios.com

Best e-business Software

Winner...
NQcontent V2
NQcontent is the Web content management software of choice for ColdFusion developers, making it possible to create dynamic, scalable e-business without limitations.More than Web content management, NQcontent makes it possible to deliver rich Web data management interfaces in minutes, giving total control over all aspects of e-business sites.
Company: Netquest
Web: www.nqcontent.com

First Runner Up...
inFusion Mail Server
Company: On-Line Data Solutions
Web: www.coolfusion.com

Second Runner Up...
A QuickEStore
Company: A QuickEStore
Web: www.quickestore.com

Third Runner Up...
CFWebstore
Company: Dogpatch Software
Web: www.cfwebstore.com

Best Education and Training

Winner...
CFUG-O-RAMA
Fig Leaf Software, a premier Macromedia partner, provides high-caliber instruction at their training centers in Washington DC, Atlanta, Chicago, Baltimore, Northern Virginia, or on-site.
Company: Fig Leaf Software
Web: www.figleaf.com

First Runner Up...
TeraTech Training
Company: TeraTech
Web: www.cfxgraphicsserver.com

Second Runner Up...
CF_Buster
Company: Centrasoft Corporation
Web: www.centrasoft.com

Third Runner Up...
House of Fusion Mailing Lists
Company: House of Fusion
Web: www.houseoffusion.com/cf_lists

Best Web Application

Winner...
FuseTalk
A fully customizable and scalable discussion forums solution, FuseTalk accommodates any number of users in an interactive community that increases Web traffic, saves costs and wins increased site loyalty. With full administration capabilities, robust tools, rich features and an easily customizable UI, FuseTalk is the choice of over 600 companies worldwide. For Ignite Sports, building strong communities of fans builds strong team support and that translates into greater ticket sales for their clients.
Company: e-Zone Media
Web: www.e-zonemedia.com

Winner...
NQcontent V2
NQcontent is ideal for ColdFusion developers who wish to rapidly build dynamic scalable Web sites for their e-business applications. Developers can use NQcontent without limitation in their favorite development environment like Macromedia Dreamweaver MX or Flash MX to create rich content and custom data driven applications.
Company: Netquest
Web: www.nqcontent.com

First Runner Up...
Fig Leaf Software–EarthTrends Environment, a Searchable Database
Company: Fig Leaf Software
Web: http://earthtrends.wri.org

Second Runner Up...
CF Controlpanel V3.0
Company: CFM-Resources.com
Web: www.cfm-resources.com

Best Web Development Tool

Winner...
Macromedia Dreamweaver MX
Macromedia Dreamweaver MX combines functionality of what had been three separate products – Dreamweaver, Dreamweaver UltraDev and HomeSite – to provide programmers, developers, and designers with one environment to create and manage any professional Web site in a cost-effective way. Dreamweaver MX spans HTML, XHTML, XML, Web Services, ColdFusion, ASP.NET, ASP, JSP, or PHP.
Company: Macromedia
Web: www.macromedia.com

First Runner Up...
NQcontent V2
Company: Netquest
Web: www.nqcontent.com

Second Runner Up...
CommonSpot 3.0
Company: PaperThin
Web: www.paperthin.com

Third Runner Up...
Macromedia Flash MX
Company: Macromedia
Web: www.macromedia.com/software/flash

Best Web Hosting

Winner...
CFXHosting
One of the first companies to focus on ColdFusion hosting, CFXHosting provides customers with resources and knowledge. Also awarded the 2000 CFDJ Readers' Choice Award for excellence in ColdFusion Web hosting.
Company: CFXHosting
Web: www.cfxhosting.com

First Runner Up...
CFM-Resources.com
Company: CFM-Resources.com
Web: www.cfm-resources.com

Second Runner Up...
Edge Web Hosting-Dedicated/Shared CF
Company: Edgewebhosting.net
Web: www.edgewebhosting.net

Third Runner Up...
ColdFusion Hosting
Company: Intermedia.net
Web: www.intermedia.net

Best Web Site

Winner...
Macromedia Designer & Developer Center
A resource center targeted toward Macromedia developers and designers.
Company: Macromedia
Web: www.macromedia.com/desdev

First Runner Up...
CFM-Resources.com
Company: CFM-Resources.com
Web: www.cfm-resources.com

Second Runner Up...
Ben Forta's ColdFusion Site
Company: Ben Forta
Web: www.forta.com/books/0321125169

Third Runner Up...
CFUG-O-RAMA
Company: Fig Leaf Software
Web: www.cfugorama.com

Most Innovative CF Application

Winner...
NQcontent V2
NQcontent is the Web content management software of choice for ColdFusion developers. Developers who know ColdFusion can unleash their talent with NQcontent, experience new levels of productivity, amaze customers, and deliver more in less time.
Company: Netquest
Web: www.nqcontent.com

First Runner Up...
inFusion Mail Server
Company: On-Line Data Solutions
Web: www.coolfusion.com

Second Runner Up...
CommonSpot 3.0
Company: PaperThin
Web: www.paperthin.com

Third Runner Up...
"DaVinci"
Company: Fig Leaf Software
Web: www.davinci.figleaf.com

Working with Use Cases

Learn to speak the same language

Designing Web applications is very different from designing Web sites. Web site design typically consists of finding ways to organize information so that it is clear (hopefully) to the site visitor.

By Hal Helms

An excellent book that deals with Web site design is Steve Krug's *Don't Make Me Think: A Common Sense Approach to Web Usability* (Que; 2000). Krug explains and illustrates why so many Web sites fail in their mission. Displaying information on the Web, Krug explains, is different from showing that same information in the offline world, and requires a different approach. Krug does a wonderful job of offering practical advice on making Web sites better.

Web applications, though, are not so much about the static display of information as about capturing business processes in software. While the Web offers challenges different from traditional client-server programming, Web applications have much more in common with traditional desktop applications than they do with Web sites.

That point is often overlooked by Web developers, who may begin a project by seeking to ascertain the main sections of a Web application such as "Customer Service", "Store", and so on. A better method, used by many practitioners of object-oriented design, involves finding the roles played by individuals interacting with the application (or system), and determining what each actor needs to accomplish. For example, an Inventory Clerk – an actor in an online inventory system – may wish to add the initial inventory. This desired action is called a *use case.*

The process of discovering use cases has a singular benefit: it allows developers and clients to communicate in strictly non-technical terms. Too often, we developers have assumed that the ideas and terms we use to think about an application are likewise understandable and helpful to clients. Such is rarely the case. More often, when a specialist tries to communicate with a non-specialist, a great deal of confusion occurs as the nonspecialist struggles to understand the terms used by the specialist. If you've ever had the dizzying experience of speaking with a doctor who insisted on speaking in "medicalese," you'll recognize the problem.

The problem isn't simply one of the doctor (in this case) relying on jargon. The doctor uses jargon because it allows precision and clarity in very few words – certainly a desirable thing. When the doctor tells you that you have a subdermal hematoma, for example, he or she is using a term with a very precise definition – provided that you have an understanding of the specialist's vocabulary. But unless you've been exposed to some medical education, you won't understand that what the doctor calls a subdermal hematoma, you call a "bruise."

In the case of developers and clients, we run into the same problem. We developers, by nature and training, analyze a problem; we determine the ingredients that make up the problem and the relationship between them, and our language reflects this analytical bent. But in communicating with clients, we need to find a new vocabulary that includes the client in the process. Without this, we have no hope of finding out what the client really wants.

Use cases provide a simple, clear way of communicating with clients. When we talk in terms of use cases, we speak in terms that are clear to the client. In fact, we deliberately speak in terms that make sense to the client: "Inventory Clerk adds initial inventory." Use cases have proven so valuable that a formal visual "language" has been developed to assist developers in speaking a common language – yet one that is understandable to clients.

This is called a use-case diagram, and it's part of the Unified Modeling Language, a visual language that uses different diagram types to focus on different aspects of an application. While some diagrams – class diagrams and state diagrams, for example – provide much more precision in analysis, use-case diagrams focus on communication between client and architect.

Let's look at a use case that deals with doctors and patients (Figure 1):

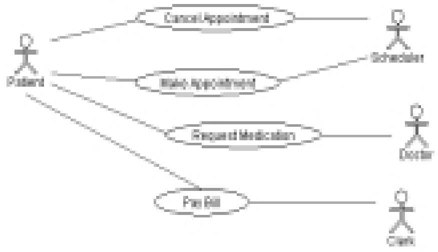


Figure 1: A simple UML use-case diagram

The actors involved are Patient, Scheduler, Doctor, and Clerk. What sort of things does a Patient want to do? The Patient wants to make or cancel an appointment. Both of these use cases involve another actor, the Scheduler. The Patient wants to request medication – a use case that involves the Doctor actor – and so on.

Though very simple, use-case diagrams offer a powerful means of communicating. Here are some tips for employing use cases:

- ***Use cases should begin with a strong verb:*** In the use-case diagram above, the verbs are *cancel, make, request,* and *pay*. These are better than more generic and less descriptive verbs such as *do, perform*, and their like.
- ***Use terms that make sense to your client:*** Since the purpose of use cases is communication, make sure that the words you use are meaningful to your client. This will help you understand the domain the application must operate in.
- ***If there are timing considerations, stack use cases:*** To use an example from a different domain, a Bank Customer might have the following use cases: Open Account, Deposit Funds, Withdraw Funds, Close Account. Since each succeeding use case requires that the previous one has been done first, stack the use cases in the order in which they must be performed.
- ***Approach use-case diagrams from two directions:*** Ask both: "What things would this actor want to do?" and "What actors would be involved in this particular action?" Some clients will naturally think of actors first and then their actions, while others reverse that order. Make sure you have all your bases covered.
- ***Consider using a UML diagramming tool***: There are several excellent ones available, including some that are free. A diagramming tool can save you time and make changes easy.
- ***Invest in a good book on UML techniques:*** UML is a very rich modeling language, and you may find other modeling techniques helpful. Some of the better books on UML are:

  – *The Unified Modeling Language User Guide*, by Grady Booch, et al. (Addison-Wesley; 1998)
  – *Writing Effective Use Cases*, by Alistair Cockburn (Addison-Wesley; 2000)
  – *Advanced Use Case Modeling: Software Systems*, by Frank Armour, et al. (Addison-Wesley; 2000)
  – *Use Cases: Requirements in Context*, by Daryl Kulak, et al. (Addison-Wesley; 2000)

Use cases are one technique – one proven to be very effective – of communicating with clients and documenting this communication. Try it – I think you'll find it adds clarity, precision, and depth to your requirements modeling.

About the Author

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

hal@fusebox.org

Compilation and Precompilation in CFMX Templates

Part 2: The nitty gritty

Last month I introduced a precompile "batch" file that allowed you to manually compile CFMX templates, saving the penalty paid by the first user to browse a CFMX template after it was created or edited.

By Charlie Arehart

But there will be curious folks (and bit-twiddlers) among you who will want to know more – maybe lots more. How much time is this really saving? If it's compiled to disk, how and when does CFMX read it into memory to execute it? What's the cost of that? What happens with CFINCLUDEd files? Where does the compiled code go? Can I look at it? Can I just delete the generated class files instead? How do I determine which class file was generated for which CF template? Can I distribute the compiled code on other servers without the source code?

I'll address these questions and more in this conclusion.

Evaluating the Execution Time

Before going on, it may be helpful to explain something about determining the impact in terms of execution (and other startup) times for running a template. It will be very helpful to turn on the "Report Execution Times" option in the ColdFusion Administrator settings for "Debugging Settings". This will produce a display (see Table 1) at the bottom of a page when executed in a browser.

As in previous releases, you will only see this debugging information if the Administrator has enabled it. Also, if any IP addresses are listed in the "Debugging IP Addresses" page of the Administrator, then you will only see it if you are in that list. If there are no values in that list, all who execute CF templates on the server will see this info. Finally, if you still don't see this information, be sure that your template (or the application.cfm that is called before it) does not have <CFSETTING ShowDebugOutput="no">.

This particular example shows a file called test.cfm, which included a file called test.html, and also had an application.cfm that was called. It also shows a separate value for "startup, parsing, compiling, loading, & shutdown". We'll get to that in a moment. Keep an eye on this set of timing information when trying to evaluate the impact of various choices and settings.

Compilation Doesn't Eliminate All the Waits

So, after reading my previous article you now know how to precompile a template. You've paid the cost of compiling the code to its underlying Java byte code and now it should execute without any parsing/compiling delay (assuming it's not been edited). What if you restart the server? Will the first user really experience no further delay? Not exactly.

You may recall that I said that CFMX writes the compiled code to disk. Did you wonder how it gets from disk into memory, for CFMX to execute? I mentioned at the outset of the previous article that even in ColdFusion 5 and before, the first execution of a template after a server restart would cause CF to interpret the template into "pcode" and then load that into a "template cache". The code was then executed from memory all day, rather than being read from disk (unless the cache filled, in which case the least recently used template would be flushed from the cache). This was a useful solution for making code run as quickly as possible when executed.

CFMX faces the same dilemma. While the compilation of code to a Java class file eliminates the interpretation step the first time the code is executed after each restart, the fact remains that the compiled code needs to be read from disk and loaded into ColdFusion's memory – in that same template cache that existed in previous releases.

The precompile batch file from the previous release does not perform this "load" of the compiled code into CFMX's template cache, so you can't save that time. Similarly, when the server has just been restarted, it will need to be "loaded" into memory. It's just the first user to run the code after restart (or a compile or precompile) that experiences that load time. So be aware that precompiling won't eliminate the entire wait. There's still the time needed to load the code into the template cache the first time it's executed after restart.

Total Time	Avg Time	Count	Template
10 ms	10 ms	1	D:\CFusionMX\wwwroot\demo\Application.cfm
0 ms	0 ms	1	D:\CFusionMX\wwwroot\demo\test.cfm
0 ms	0 ms	1	D:\CFusionMX\wwwroot\demo\test.html
10 ms			STARTUP, PARSING, COMPILING, LOADING, & SHUTDOWN
20 ms			TOTAL EXECUTION TIME

Table 1: Execution Time

Trusted Cache Still Applies

I mentioned in the previous article that ColdFusion automatically detects any new or newly edited templates and then automatically compiles them (in CFMX, or interprets them in CF 5 and before). It does this by checking to see if a CF template about to be executed is new or newly updated. And it does this before each invocation of a template, by every user.

That's a rather costly operation, but it's the cost of being able to freely make code changes and see them reflected immediately. If your production environment is designed so that you don't make changes to your code except on a scheduled basis (perhaps overnight), you should enable the "Trusted Cache" setting in the ColdFusion Administrator.

By enabling "Trusted Cache", you're telling ColdFusion that once it loads a program into memory, it should no longer check to see if its underlying source code template has been updated. This will remain in effect until the server is restarted. Sadly, disabling it and re-enabling it does not cause the template cache to be flushed.

One thing that has changed about the "trusted cache" is the specification of its size. Prior to CFMX, the size setting (on the same page in the Administrator) was set in bytes, and it was difficult to calculate an accurate size. Now, the setting is the total number of templates to cache, rather than their size. It still defaults to 1,024.

Also, be careful in evaluating the impact of these steps. If you watch the precompile.bat file process, you may notice that compiling even a single template seems to take longer than if you simply let CF compile it automatically. The compiler process in the precompile.bat file has some load time itself on the first template it compiles. If only one template is precompiled, you see that time reflected in the time it takes to compile. If multiple files are precompiled at once, you see each individual template taking less time than if it was compiled alone.

Handling Included Files

In my example above I included a test.html file, but what if I had included a cfm file? Do you wonder if there might be an effect on compilation if code in the included file changes? Would you expect ColdFusion to have to recompile the including template?

It does not. Precompiling the including file after the included file has changed isn't necessary. Indeed, if you look at the compile time reported by the precompile tool it will be reported as 0ms. This reiterates the point that CFINCLUDE is a runtime directive, not a compile time one. Many misconstrue that. (Further proof is demonstrated by the fact that the value of the TEMPLATE attribute naming a file in CFINCLUDE can in fact be a variable. It's clearly a runtime directive.)

What's the difference? Say you have a template that includes another and that included file changes often during the day. You will notice a delay in loading the file. But it would have been true in CF 5 and before. Compilation isn't the cause, and precompilation is not the answer. You just need to get CF to reexecute the template so that it runs the CFINCLUDE to detect that changed file and load it into memory. Again, it's a runtime issue.

I've done this and watched the time reported in the debug output on the first execution of the calling template (after the included file has changed). In one example it went from 1000 ms (reported as execution time for the calling template) to 0ms from then on. Note that this change is in the runtime of the template, not its compile time. So it's not really a compilation issue.

What can you do, especially if some automated process is causing updates to the included file? You could add to that automated process (or create another) that simply executes the calling page once after the included file has changed. This could be kicked off via CFSCHEDULE or CFHTTP. The point isn't to save the resulting output in any way (which is an option with these tags), but merely to cause execution of (or in other words to "touch") the calling page so that it does what it needs to do to interpret the change in the included file.

Where Does the Compilation Go?

I also mentioned in the previous article that CFMX compiles your cfm (and cfc) templates into .class files, which are Java byte code files. If you're curious, the files are written to (and executed from) the cfclasses subdirectory of [cfusionmx]\wwwroot\WEB-INF\ directory where CFMX is installed. This occurs whether you are using another Web server or have located your file outside the default wwwroot location. CFMX compiles and runs the code from this cfclasses directory, regardless of the location of the source file.

This is the reason why in the previous article we needed to specify the –webinf directive in the batch file, to tell it to place compiled code there. If it's not placed there, it won't run in CFMX.

Be warned, though, that the file names for these class files may not be at all apparent. A CF template named Setsession.cfm might lead to a class file named cfsetsession2ecfm1011928409.class.

All templates from all directories end up in this one cfclasses subdirectory. They're not stored here in any subdirectories related to their original location. Instead, CF includes a hash of the directory name in that set of numbers after the file name. Keep that in mind when trying to associate a given class file with its original cfm template. The hashing process is a bit convoluted. (See my blog entry of Oct. 5 for more details, at cfmxplus.blogspot.com.)

Perhaps the easiest way to detect which class file goes with which source file is to simply edit the file and then execute (or precompile) it. Look in the cfclasses directory for the most recently created class file. Assuming your server is not too busy with many compilations taking place, it should be pretty easy to associate the classname with the CF source code name.

Saving Java Source Code Produced by CFMX

So that's where the compiled source code goes. But what about seeing the actual uncompiled Java source code that your CF template is converted into? Normally it's of no concern to CF developers what CFMX is doing under the covers in converting our CFML to Java.

For the ardently curious among you, did you know that you can ask CFMX to save the Java code it creates, in source form? You can. It's an undocumented feature, and while I've had no trouble doing it, I must warn that this is something you do at your own risk. Neither I, nor SYS-CON, nor Macromedia accept responsibility for your trying this.

The setting can only be enabled by someone with administrative control of the server, and the setting is also server-wide. It will add a slight additional time to the compile process, so it's not something you'd want to turn on in production. It probably ought not be left on in development either.

You need to edit the file web.xml in the [cfusionmx]\wwwroot\WEB-INF directory. There, if you're familiar with XML files, you'll find a parameter called "coldfusion.compiler.saveJava". Change its value from false to true. Save the file. Restart the server.

Now, whenever a new or recently edited file is compiled (whether automatically by CFMX or by our precompile.bat file), CFMX will also create a ".java" file along with the ".class" file. This ".java" file will be found in that same [cfusionmx]\wwwroot\WEB-INF\cfclasses\ directory as the ".class" files (and will be subject to that same issue of the curious file naming mentioned above).

The Idea of Deleting the Generated Class Files

Some have proposed that instead of precompiling their code they'd just as soon delete the underlying Java class file that was created when it was last compiled. That may seem like overkill, but there are times when it might be worth trying. Just note that, as the previous sections discussed, finding the class file that's associated with a given source template can be challenging. While some may simply delete all the class files, that's certainly overkill.

Don't forget that in the previous article I mentioned there is a –f directive you can pass to the compile process (by modifying the precompile.bat file). That will force a recompile of a file even if CF doesn't think it's necessary. Sometimes that solves the same problem that deleting the class file would solve.

The Dream of Distributing Compiled Code Without Source

Finally, all this talk of creating compiled Java code has excited some. They wonder if they might finally be able to distribute the compiled code without the associated source files. From an intellectual property protection standpoint, this makes sense. Being able to do so is important to those who sell ColdFusion applications. They want their clients to be able to use the code, without them (or their competitors) seeing the actual source code.

Unfortunately, this is not going to happen with the current release of CFMX (if at all). Macromedia has designed the process so that the source code must be present for the file to be loaded. That's an interesting key point. Some were excited by a demo by Ben Forta at last year's DevCon where he was able to delete a source code file and the template continued to run. Some have even noticed that that can still work. Not always, though. Only if the trusted cache setting is enabled.

With what you know from our previous discussion of loading templates and using the trusted cache setting, you may be able to figure out what's happening. If you run a CF template, it's loaded into memory (after being compiled if it changed). Once it's in memory, that's where CF runs it from. And if the trusted cache setting is enabled, then CFMX no longer looks to the source code file to determine if it's changed. Indeed, it also doesn't look to see if it's been deleted.

But if you restart the server, then CF will try to load the template from disk and even with the trusted cache enabled, the first time the template is run, it will look to see if the CF source template has changed since it was last compiled to disk. Only if they disabled that would we be able to truly distribute the source code.

Macromedia folks have further asserted that they won't enable that feature because it's inappropriate to assume that code compiled on one machine would necessarily work on another machine. It's important to note that your ColdFusion code is more than just the underlying Java byte code it's been compiled into. There are libraries that come with CFMX that are called when your code is run. You would at least need to distribute those as well. Whether Macromedia will address this problem for commercial solution developers is still unclear.

Conclusion

The process of compiling and understanding the compilation process for CFMX templates is something that's not well documented. I don't even want to claim to fully understand it. I've encountered some things in preparing this article that still confound me, and since it's undocumented, maybe we can't expect to fully understand it. I just hope that the information shared here will help you, and that it may also motivate Macromedia to decide what aspects of the process they want to make more public.

You could argue that an Admin interface would be a useful addition in doing the compilation process. Perhaps someone has already started working on that. I welcome comments, questions, and concerns you may have about this compilation process. Feel free to e-mail me or post comments below the online version of this article at the www.sys-con.com site.

About the Author

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

carehart@systemanage.com

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
activePDF	www.activepdf.com	949.582.9002	4
CF ADVISOR	WWW.CFADVISOR.COM	201.802.3069	47
CFDYNAMICS	www.cfdynamics.com	866.CFDynamics	23
CFXHosting	www.CFXHosting.com	866.CFX.HOST	31
EDGE Web Hosting	www.edgewebhosting.net	866.edge.web	25
Ektron	www.ektron.com/free	603.594.0249	41
E-Zone Media	www.fusetalk.com/new	866.477.7542	11
HostMySite.com	www.HostMySite.com	877.215.HOST	37
Interland	www.interland.com	877.384.7938	3
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	52
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	45
Macromedia	www.macromedia.com/go/usergroups	877.460.8679	13
Macromedia	www.macromedia.com/go/CFMXlight	877.460.8679	17
NetQuest	www.nqcontent.com	357.22.813560	21
New Atlanta	www.newatlanta.com	678.256.3011	2
On-Line Data Solutions	www.coolfusion.com	631.737.4668	6
Pacific Online	www.paconline.net	877.503.9870	39
PaperThin	www.paperthin.com	800.940.3087	27
SAMS PUBLISHING	www.samspublishing.com		51
SYS-CON MEDIA	www.sys-con.com	201.802.3069	33, 45
Web Services Edge 2003	www.sys-con.com	201.802.3069	29

<CFIF> Performance Measurements

A jumping-off place for optimizing CFIF operations

It was an article by Michael Dinowitz, “Comparisons with CFIF,” that made me reevaluate how I was writing my CFIF statements.

By Tom Nunamaker

Since the common <CFIF myVar IS Value> was the slowest way to compare two strings, the table of data presented in the article (see *Fusion Authority*, 1/10/00, www.houseoffusion.com) seemed incomplete to me. I wondered what the performance data would show for integers – and for Boolean tests, as we all write tons of those in our code as well.

Other common CFIF tasks were also missing, such as tests for existence and bitwise operator tests. I wanted to know the best way to optimize all of these CFIF operations instead of just string comparisons. I also wanted to update the data with ColdFusion 5 to see if it was still valid or if CF5 had changed the results in any way.

I found that the String “COMPARE” function is still the best choice for string comparisons. For Boolean and integer uses “myVar IS value” had the best results.

Tests for existence showed that IsDefined() performance is almost identical to the older ParameterExists, so we know we aren’t giving up anything by using the newer function. The bitwise operator tests surprised me. It’s well known that bit operations are faster than multiplys. The CFML implementation of BitAnd was slower than using MOD to test for even or odd numbers. Bit Shift Left was slower than doing a simple multiply, which was also surprising. The good thing is, you don’t have to learn bitwise operators to get maximum CFML performance. If CFML had true bitwise operators instead of bitwise functions, we’d likely get that improved performance since we’d be eliminating the overhead of the function calls.

All tests were made with the following basic code; “TESTING CRITERIA” was where I entered in the operation I was testing.

```
<cfset tickBegin = GetTickCount()>
<cfloop index="i" from="1" to="#itera-
tioncount#">
  <cfscript>
    myvar = i;
    if ( TESTING CRITERIA )
      myvar = i + 3;
  </cfscript>
</cfloop>
<cfset tickEnd = GetTickCount()>
<cfset loopTime = tickEnd - tickBegin>
```

My testing platform was a 1.8GHz Athlon with 1GB of RAM running Windows XP Pro, CF 5.0, and Deerfield’s Website Pro Web server.

For each series of measurements I tested three conditions to measure if there was any difference in handling smaller loops versus larger loops:

- 10,000 iterations, 50 times
- 1,000 iterations, 100 times
- 100 iterations, 500 times

Iterations were how many times the loop executed before measuring the elapsed time. I looped over all of the iterations the indicated number of times to get an average sample.

I wanted to measure the difference in constant and variable comparisons. When I refer to an *integer constant*, I’m referring to a test like this:

```
<cfif myVar IS 1>
```

versus a variable test that would be a test like this:

```
<cfset myConstant = 1>
<cfif myVar IS myConstant>
```

For the Boolean tests I added the common construct of:

```
<cfif myVar>
```

to see if it offers any speed advantages besides less typing and being easier to read. This is such a common way of performing a Boolean test that even if it is close to being the fastest, many people would consider the readability to be more important than tiny speed improvements.

Tables 1–6 show the best result highlighted in bold print. Times shown are average times in milliseconds for the number of iterations indicated, averaged over the loop count times.

Note that in Table 1 the traditional “myVar IS value” and “myVar EQ value” have nearly identical speeds that are much faster than the “NOT Compare” and “Find” constructs. For all integer comparisons the traditional coding style is the fastest. Using a constant instead of a variable is about 19% faster, so use a constant if you can.

Loop Count	50	100	500	50	100	500
Iterations	10000	1000	100	10000	1000	100
Measurement	integer constant			integer variable		
	Ave ms	**Ave ms**	**Ave ms**	**Ave ms**	**Ave ms**	**Ave ms**
myvar IS value	**144.46**	**13.75**	**1.642**	**179.66**	**19.13**	**1.790**
myvar EQ value	**143.54**	**14.82**	**1.624**	**179.06**	**18.40**	**2.042**
myvar NEQ value	213.82	21.11	2.240	252.74	24.45	2.804
NOT Compare(myvar, value)	256.20	25.87	2.968	271.20	27.63	2.682
NOT CompareNoCase(myvar, value)	257.56	26.93	2.946	270.36	27.66	2.900
Find(myvar, value)	243.60	24.63	2.602	260.98	25.84	2.368
FindNoCase(myvar, value)	257.24	26.22	3.082	277.56	28.02	2.902
REFind(value, myvar)	327.30	31.83	3.588	365.74	36.04	3.446
REFindNoCase(myvar, value)	416.80	41.25	4.604	429.82	43.59	5.002

Table 1: Integer measurement results

Loop Count	50	100	500	50	100	500
Iterations	10000	1000	100	10000	1000	100
Measurement	string constant			string variable		
	Ave ms	**Ave ms**	**Ave ms**	**Ave ms**	**Ave ms**	**Ave ms**
myvar IS value	520.20	52.00	5.546	547.12	53.99	5.444
myvar EQ value	521.96	52.89	5.368	545.36	54.67	5.784
myvar NEQ value	597.80	59.73	6.088	625.56	62.79	6.534
NOT Compare(myvar, value)	**248.40**	**24.32**	**2.662**	**270.98**	**26.94**	**2.826**
NOT CompareNoCase(myvar, value)	253.18	24.92	2.624	278.16	27.74	2.900
Find(myvar, value)	237.72	24.04	2.624	260.82	26.64	2.890
FindNoCase(myvar, value)	254.14	26.61	2.506	277.42	27.53	2.944
REFind(value, myvar)	318.16	31.74	3.084	339.76	33.33	3.522
REFindNoCase(myvar, value)	409.86	41.93	4.526	434.18	42.97	4.486

Table 2: String measurement results

Loop Count	50	100	500	50	100	500
Iterations	10000	1000	100	10000	1000	100
Measurement	Boolean constant			Boolean variable		
	Ave ms	**Ave ms**	**Ave ms**	**Ave ms**	**Ave ms**	**Ave ms**
myvar	177.84	18.62	1.604	180.88	18.12	1.942
myvar IS value	**138.38**	**14.52**	**1.282**	**169.86**	**16.70**	2.000
myvar EQ value	**141.44**	**13.92**	1.384	**170.62**	17.21	**1.544**
myvar NEQ value	206.90	21.14	2.384	242.00	24.36	2.522
NOT Compare(myvar, value)	248.14	25.13	2.386	270.52	28.14	2.706
NOT CompareNoCase(myvar, value)	249.36	25.45	3.088	273.52	27.66	3.022
Find(myvar, value)	241.74	24.95	2.980	260.74	26.04	2.712
FindNoCase(myvar, value)	253.80	25.63	2.844	276.06	26.53	2.866
REFind(value, myvar)	314.02	31.45	3.122	339.34	33.84	3.562
REFindNoCase(myvar, value)	419.62	41.54	4.140	441.04	44.18	4.562

Table 3: Boolean variable results

“NOT Compare” is the preferred comparison method for string constants and strings, despite “Find(myvar,value)” being slightly faster (see Table 2). The reason is that Find can’t guarantee an exact match unless you also compare the string lengths, which would really slow you down. After all, comparing two strings is really what the Compare function is designed to do, so it follows that it should be the fastest method for string comparisons.

The Boolean constant I used was TRUE. I didn’t expect this result. The construct <cfif myvar> was *not* the fastest Boolean comparison method. The common “myvar IS value” was faster almost every time; “myvar EQ value” virtually tied for the best times and in one case was the fastest method. Compared to the constant, TRUE was from 10% to 35% faster than using a Boolean variable set to TRUE (see Table 3).

Even though <cfif myvar> wasn’t the fastest, many people will probably consider the readability benefit enough to take a 7% speed penalty compared to myVar EQ value. If you use a Boolean constant test compared to <cfif myvar>, the penalty is 20% to 25%. You have to decide whether the page processing time needs every ounce squeezed out of it. If it does, then pick the fastest method. If it’s a page with limited access (an admin form, for example), then readability and maintainability would probably be your preference.

Bitwise Operations

Bitwise operators allow low-level bit operations that are typically necessary in assembly language. Bitwise operators are commonly used to manipulate flags, but Bit Shift Left is often touted as being the fastest way to multiply an integer by 2. Shifting the bits left one place in binary multiplies the number (integers only) by 2 in the same fashion that shifting a base 10 number left one multiplies it by 10. For example, 11 base 10 shifted to the left becomes 110. In modern C compilers the optimizer will automatically rewrite integer multiplys by 2 to a bit shift left; in CFML, however, our Bitwise Shift Left operator asks us how many places we want to shift. This added overhead actually makes the CFML bit shift left slower than simply writing i = i * 2, which was 11% to 21% faster than the BitSHLN function. Table 4 shows the speed measurements for the bitwise shift left versus multiplying a number times 2.

Since we have an operator that does ask us how many bits to shift left, I wondered if the performance would be closer for multiplys times 4 or 8. The simple method ranges from 18% to 25% faster. Table 5 shows the results of that test.

Another common task in a CFIF statement is to determine whether a number is even or odd (see Table 6). This is useful when applied to the currentrow in a query output to alternate the colors of the rows. The question is what’s the best way to determine if currentrow is even or odd. Let’s try using the bitwise AND operator and compare that to using the MOD function.

Hmm. That’s an odd result. Using the CFML BitAnd function, except for a very small number of iterations, was about 16% slower than using MOD. For other programming languages bitwise operators are indeed faster than using MOD. For CFML however, stick to the MOD function.

Loop Count	50	100	500
Iterations	10000	1000	100
	Ave ms	**Ave ms**	**Ave ms**
BitSHLN(i,1)	179.60	18.43	1.986
i * 2	**149.08**	**14.52**	**1.762**

Table 4: Speed measurement for bitwise shift left

Loop Count	50	100	500
Iterations	10000	1000	100
	Ave ms	**Ave ms**	**Ave ms**
BitSHLN(i,2)	189.90	18.12	2.204
i * 4	**150.62**	**14.50**	**1.722**
BitSHLN(i,3)	188.50	18.83	1.860
i * 8	**153.38**	**14.11**	**1.422**

Table 5: Test results for multiplys times 4 or 8

Loop Count	50	100	500
Iterations	10000	1000	100
	Ave ms	**Ave ms**	**Ave ms**
BitAnd(i,1)	248.08	24.86	**2.362**
i MOD 2	**208.16**	**20.91**	**2.384**

Table 6: Test to determine whether number is even or odd

Loop Count	50	100	500
Iterations	10000	1000	100
	Ave ms	**Ave ms**	**Ave ms**
ParameterExists(xyz)	235.48	25.12	2.706
IsDefined("xyz")	233.80	24.52	2.480

Table 7: Test results for speed

INTEGER TESTS:	myVar IS value myVar EQ value
STRING TESTS:	NOT Compare (myVar, value)
BOOLEAN TESTS:	myVar IS value
BITWISE OPERATIONS:	Use MOD 2 instead of BitAND Use i * 2 instead of BitSHLN
TESTING FOR	Use IsDefined().
EXISTANCE:	ParameterExists has been deprecated

Table 8: CFIF usage guidelines

Testing for Existence

Another very common CFIF task is to test for the existence of a variable. ParameterExists has been deprecated in favor of IsDefined, but just the same I wondered what the speed differences between these two functions were. Table 7 provides the test results.

As you can see, there's practically no difference between the two functions in terms of speed. We can feel secure in knowing that IsDefined() doesn't impose any performance penalty compared to the older ParameterExists.

Summary

Distilling the data in the tables, I generalized the results to the CFIF usage guidelines in Table 8.

Comparing to a constant is *always* faster than comparing to a variable. The speed advantage can exceed 35%, but mostly ranged from 10% to 20% faster.

For a Boolean test, using myVar IS value was the fastest. It's common to see <cfif myVar> in code, and you may still want to continue using that for readability purposes. If you do, understand that you're paying a 7% to 25% penalty for that readability.

For string comparisons the "NOT Compare" construct was the most efficient, which makes sense since that's what that function is designed to do.

We should continue using the more widely used MOD function instead of what our intuition might tell us regarding BitAnd to test for even numbers.

We have only one approved function to test for existence. The older ParameterExists function is still in the language, but only for backwards compatibility. Speedwise, the two functions are nearly identical, so use the current IsDefined function.

These test results are valid *only* for my situation. Yours is probably different. I have no idea if these numbers are similar for Linux, Solaris, or even other versions of CF or Windows. The only accurate way for you to know is to run your own tests.

About the Author

Tom Nunamaker, a USAF T-37 instructor pilot, has been programming in various languages since 1974. He started ColdFusion programming in 1996 and is a certified advanced ColdFusion 5.0 developer. His work includes integrating pilot flight plans to the European Air Traffic Control System for the USAF and Web hosting. He has posted several custom tags at www.toshop.com.

tom@toshop.com

Editorial

—continued from page 5

CFDJ. The rest of the content we're leaving up to you – so please, stay in touch, don't be a stranger, and feel free to e-mail us at anytime. Tell us what we're doing right, what we're doing wrong, and what you'd like to see us cover.

Stay tuned in the more immediate future for next month's issue. We'll have full coverage of Macromedia DevCon, with all the latest news, some exciting Macromedia announcements, and some more good news concerning ***ColdFusion Developer's Journal*** itself.

In my own personal coding world, I just completed the leap to MX and have officially uninstalled ColdFusion Studio 5 from my primary machine, finding that Dreamweaver MX is where my current best development performance will come from. It was a bit of a learning curve that went in both directions – having to unlearn some things that I've been doing for ages, and having to learn some new ways of doing things that I've finally become comfortable with. The various changes have definitely made me more productive than I was before.

So here's to self-improvement and to magazine improvement. In keeping with this relentless spirit of all-around betterment, I've even arranged for a new headshot this month!

CF Community

—continued from page 7

When asked to explain his code, Timothy stated: "I am treating the " " space as the list delimiter, so just like a comma, it gets ignored. Everything after the last " " space is what I grab and get the len off of. You can treat any string as a list, provided you can find a way to make it work. A good example of this is in file-based operations. You can use listLast(cgi.script_name, '/') to get the name of the current template, and use list functions to parse out paths. List functions have a million and one uses."

I couldn't agree with you more, Tim.

Editorial note: It may be worth noting that another solution for getting the filename from a path like that is the GetFileFromPath, as in GetFileFromPath(cgi.script_name).

This is just one example of the expert advice available on the ***CFDJ*** List serve. The open exchange of ideas, tips, and tricks makes being a member of the list an invaluable resource in the toolbox of every ColdFusion developer. Those of you who have not yet subscribed (it's free!) are encouraged to do so. Just visit www.sys-con.com/coldfusion/list.cfm, supply your e-mail address, and enjoy!

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

Ask the Training Staff

A source for your CF-related questions

Greetings. I hope all of you who attended the DevCon last month enjoyed it, and came away with some great tips and tricks.

By Bruce Van Horn

This month's column is short, but hopefully addresses some problems that you've encountered recently. Keep those questions coming!

Q: I have an application that allows users to upload newsletter articles into a database memo field (I'm using MS Access for my database). My problem is this: when I query the articles from the database and output them using CFOUTPUT, some of the really long articles are getting truncated. I've double-checked, and all of the text is in the database, so CF is cutting it off either in the query or during output. Any ideas?

A: Yes! The problem is not with your database or with your output. The problem is actually with the way you have your datasource set up in the CF Administrator. To improve the performance of your queries, CF only retrieves the first 65,000 characters from long text fields by default. If your articles are over 65,000 characters, they will be truncated. What you need to do is modify a setting in your datasource. In the datasource editor, select your datasource, then click the "CF Settings" button. Check the "Enable retrieval of long text" box and then click "Update". Your queries will now retrieve all the text from your memo fields. You may want to create a second datasource for this purpose if this is the only place you need to retrieve long text from this datasource. Use your primary datasource for all of your other queries and use the "long text" datasource only where you really need it.

Q: I have a page (monthlyupdates.cfm) that forces the execution of a scheduled task using the CFSCHEDULE tag: <CFSCHEDULE ACTION="Run" TASK="CreateInvoices">. What bothers me is that my update page has to wait until the CreateInvoices page completes before running any more of the code below it. I want it to launch the scheduler and keep going without having to wait for the scheduled task to complete. Is there a way to do this?

A: I can really sympathize with you on this one! However, there is a very easy work-around! What I do to get around it is this: create a second page that has only the CFSCHEDULE tag in it (let's call it "LaunchScheduler.cfm"). Then, where you were calling the CFSCHEDULE tag in your MonthlyUpdates.cfm page, replace it with a CFHTTP call to the new LaunchScheduler.cfm page. The trick is to set a short time-out setting on your CFHTTP tag so it doesn't wait for the response. See Listing 1 for an example. The process is simple. Instead of calling the CFSCHEDULE tag that waits for completion before moving on, you now call a CFHTTP tag that waits only two seconds before moving on. The result is the same, but the wait is much less.

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.

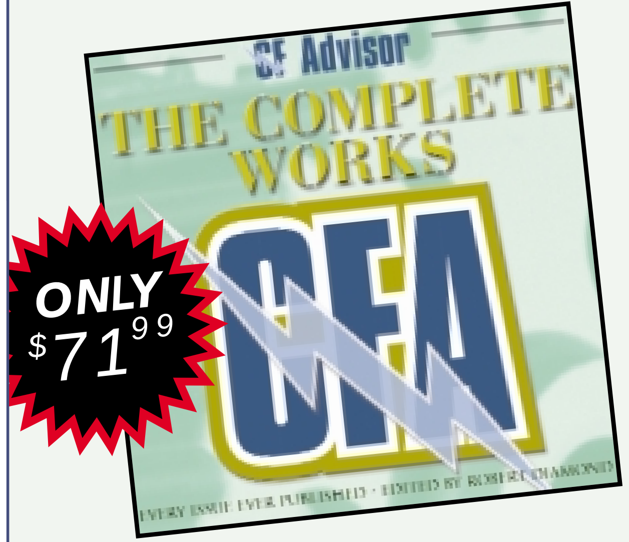
Listing 1

```
Code inside LaunchScheduler.cfm:
<CFSCHEDULE ACTION="Run" TASK="CreateInvoices">.
Code inside MonthlyUpdate.cfm
…
<CFHTTP URL="http://yourserver/LauchScheduler.cfm" METHOD="GET" TIMEOUT="2"/>
…
```

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.*

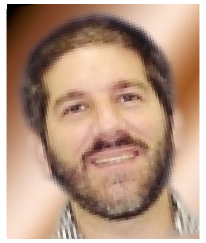
bruce@netsitedynamics.com



Software Engineering and ColdFusion

Is software engineering just a fancy name for programming?

One of the great things about ColdFusion is that it makes it so easy to put together a quality application. I've seen a lot of good work out there by people with little or no formal training. But there's only so much that ColdFusion can do for you. To advance from "good" to "great" you'll need to use software engineering practices.



By David Shadovitz

Software engineering is not just a fancy name for programming. The following three definitions of the term should make the distinction clear:

1. "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software." – *IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, 1991*
2. "A discipline whose aim is the production of quality software, software that is delivered on time, within budget, and that satisfies its requirements." – *Stephen Schach, Software Engineering, Richard D. Irwin/Aksen Associates, 1990*
3."What software engineers do." – *Peer Torngren, Industri-Matematik International (IMI), 1997*

The Software Life Cycle

Software engineers use the term "software life cycle" to encompass all of the activities of a software project. These activities include requirements analysis, design, coding, testing, and maintenance. Let's take a quick look at each of these activities, along with some methods that you can use to your advantage.

Some of these ideas really apply to a team effort, or to cases in which you're working for a client. Others are universal.

Requirements Analysis

You can't meet a goal if you have no goal. Requirements don't come easy; you'll have to ask a lot of questions and spend time to flesh out the answers. Identify the intended users. I've had analysts provide me with requirements that suit their needs and no one else's. If you're working for a client, make sure you know who is in charge.

The requirements should be independent of the implementation:

Good: Users shall be able to search for products by category, by price, and by color.

Bad: The page will use CFLOCATION to send users to the search page.

Your written, agreed-upon requirements may save you a lot of grief. As the work progresses, there's a strong tendency to try to add more functionality. This is called "requirements creep" or "feature creep", and should be resisted. Stand firm and point to your requirements document. I'm not saying that you should be completely inflexible. Just be careful, because changes or additions can blow your schedule and even ruin your project.

Design

Don't be in a rush to start coding. Work out a design. Sleep on it. Show it to someone else. Search the Web for ideas. Most ColdFusion sites involve a database, so remember to include the database in the design process. Your data is the heart of your application, so spend time on building its home. Make sure that the database is normalized, or at least that you understand the concept. If you intentionally denormalize the database, document your reasons.

If you have a good, solid design, the code will just flow right out of it. But a bad design will rear its ugly head at the worst possible time.

The best thing about adopting a design methodology – a set of standards – is that by definition it forces you to architect your project. Fusebox is one such methodology. I don't use it, but it's certainly worthwhile. With the introduction of CFMX there's been a flood of articles on the Model-View-Controller (MVC) design pattern and on Enterprise JavaBeans (EJBs). I get excited by these articles, because it's impossible to implement these ideas without careful planning – and that's engineering!

Record your design decisions. If you limit a user input to the range –95 to +95, write down the reason. It may be obvious to you, but not to the next guy (or even to yourself in three months). Prove to yourself that your design satisfies your requirements. Any holes?

Coding

As I said earlier, don't be in a rush to start writing code. Believe it or not, the other tasks in the software life cycle should take more of your time than coding.

Develop coding standards and write them down. Do you have a policy of scoping all variables? A naming convention for variables and/or templates? Review it every now and then to see whether you're following it.

Put a standard header on all of your templates. Include filename, description or purpose, author's name, and revision history. For CFML custom tags, put the calling sequence.

Include comments in your code. Let me restate that: include *useful* comments in your code. I categorize comments as absent, useless, or useful. For an example, consider the following line of code:

```
<cfif Variables.TotalExpense gt 100>
```

This is a simple operation, and perhaps a comment is not even required. But here's a comment that's useless since it doesn't add anything to what the code tells me:

```
<!--- Check whether the total expense exceeds
   $100 --->
```

Here's a comment which *is* useful since it provides the rationale for the code:

```
<!--- Check whether the total expense exceeds
   $100.
Amounts above this require departmental
   approval. --->
```

Actually, this brings up the topic of Magic Numbers. Rather than using the number 100, let's assign it a name and use that:

```
<cfset Variables.DeptApprovalThreshold = 100>
<!--- Check whether departmental approval is
   required for this expense --->
<cfif Variables.TotalExpense gt
   Variables.DeptApprovalThreshold>
```

Once you've got some decent code, conduct a peer review. That's just a fancy term for having another set of eyeballs look at your code. It's impossible to evaluate your own work objectively. By letting a colleague examine it, you'll improve your code and you'll reap the added benefit of having another person who understands it, and who could fill in for you if need be. The reviewer(s) should have a checklist whose first item is "Does the code follow our standards?" Record all defects, and make sure that they are all addressed.

Testing

You all test, right? Right?

If you test your application by using it, write down the steps as you do them. When you make an update, repeat those written tests and add new ones for any new features. If you don't, you'll be sure to forget to test some less-than-obvious functionality.

Be methodical in your debugging. Work to understand what's going on. Computers follow orders, so don't blame a bug on magic.

Maintenance

It's important to configure your code on a file and version basis. Can you identify exactly what changed between versions 1.0 and 2.0? Could you back up those changes if you had to? These tasks are handled by source control software such as Microsoft's Visual SourceSafe (VSS) or Perforce Software's Perforce. Both ColdFusion Studio and Dreamweaver make it easy to incorporate simple source control right in the development environment.

When someone asks for a change, or points out a problem, document it. Use a change-tracking system such as Mozilla's open-source Bugzilla, Fog Creek Software's FogBUGZ Bug Tracker, or TechExcel's DevTrack. With any of these tools, you can record information such as the details of each issue, the name of the originator, the software engineer, the files affected, and a description of the work done. The tool will assign a unique ID to each issue.

When managing releases, the ability to cross-reference issues and files is invaluable. Here's how to give yourself that ability. Remember my suggestion that each file should have a standard header, including a revision history? Well, whenever you modify a file, add a line to the revision history that includes the date, your name, and the issue ID. Source control software generally lets you provide a comment when you check in a file, so provide this same information in the comment.

It's a Wrap

Well, we've come to the end of the software life cycle. Does all this sound crazy? Overboard? Too expensive? I've heard all of that before, and I strongly disagree. Engineering not only yields a better product – it saves time and money. That's because the process is designed to avoid bugs, or at least to uncover them early on, and the cost of fixing a bug early in the development cycle is much lower than fixing it late in the cycle.

Lasting Value

Let's close with software engineer and author Steve McConnell's words on the value of software engineering:

> ***An investment in learning software engineering principles is a particularly good investment for a software professional to make because that knowledge will last a whole career – not be half obsolete within three years.***
>
> ***As a software engineering development professional, you need knowledge of specific technologies to do your job. But you need knowledge of software engineering principles to do your job well. A continuing pursuit of such knowledge is one mark of a true professional.***
>
> ***–Steve McConnell, IEEE Software, Mar/Apr 99, "From the Editor"***

Resources

- *Code Complete*, Steve McConnell, Microsoft Press, 1993.
- *Exploring Requirements: Quality Before Design*, Gerald Weinberg, Dorset House, 1989.
- Fusebox, www.fusebox.org
- IEEE Software, http://computer.org/software/
- StickyMinds, www.stickyminds.com/
- *The Mythical Man-Month*, Fred Brooks, Jr., Addison-Wesley, 1975 (updated 1995).
- *The Practice of Programming*, Brian Kernighan & Rob Pike, Addison-Wesley, 1999.
- *The Pragmatic Programmer*, Andrew Hunt & David Thomas, Addison-Wesley, 2000.
- *Writing Solid Code*, Steve Maguire, Microsoft Press, 1993.

About the Author

David Shadovitz is a senior software engineer at XonTech, Inc., in Los Angeles, and a member of Team Macromedia.

david@shadovitz.com

Macromedia Flash Remoting MX Available for Java and Microsoft .NET Framework-based Application Servers

(*San Francisco*) Macromedia, Inc. has announced Macromedia Flash Remoting MX for Microsoft .NET Framework and Java technology–based application servers. Already a native feature of Macromedia ColdFusion MX and JRun 4, Macromedia Flash Remoting MX now supports rich Internet application development.

Macromedia Alliance partners are already praising the ability to bring their existing applications online using this technology. This new server product delivers on Macromedia's commitment to provide an open deployment solution for rich Internet applications, and is available from the Macromedia Online Store. The trial version is available for immediate download at www.macromedia.com/go/remoting/.

Macromedia Introduces Authorware 6.5

(*San Francisco*) – Continuing its strong commitment to the e-learning market, Macromedia, Inc., has announced the availability of Macromedia Authorware 6.5, the latest version of the industry-leading visual authoring software for creating interactive e-learning applications. The new version adds support for Macromedia Flash MX, accessibility, and extensibility.

Macromedia Authorware 6.5 includes support for Macromedia Flash MX and Windows Media Player file formats; accessibility support to ensure e-learning applications; and courseware that meets governmental guidelines for accessible content and improved extensibility features.

Authorware 6.5 beta users are enthusiastic about the new and updated features in the product, which is available for download from the Macromedia Online Store at www.macromedia.com/go/buyaw65/.

CQL and Wolverine Worldwide Choose TrueSpectra

(*San Mateo, CA*) TrueSpectra, Inc., provider of high-performance, server-based imaging solutions, has announced that Wolverine Worldwide, the distributor of Merrell Boots, Cat Footwear, and Hush Puppies, has implemented TrueSpectra Image Server on its portfolio of brand Web sites. Michigan-based CQL Corporation, a custom application development firm that creates and maintains Wolverine's Web sites, handled the implementation.

"We evaluated a number of technologies before choosing TrueSpectra for Wolverine's business," said Mark Carpenter, president of CQL. "Our client needed a product that would allow their consumers to quickly see product images and zoom in on them for more detail. In selecting a technology, we looked for a product that we could easily work with as well as customize to meet the needs of our client. TrueSpectra was the only one that met these criteria."

TrueSpectra recently released Image Server v4, which enables faster Web application development and simplifies integration with leading products and platforms including Macromedia Dreamweaver MX, Flash MX, ColdFusion MX, Java applications, and servers such as BEA WebLogic, IBM WebSphere, Apache Web server, and Microsoft IIS Web server. The second-generation TrueSpectra Dynamic Zoom, with an enhanced user interface, is one of the most widely used features in Web-site development. An easy-to-use generator interface allows developers to create it quickly, enabling fast Website development without requiring plug ins or lengthy downloads. www.truespectra.com

FuseTalk Community Edition Now Delivers Interactive Collaboration

(*Ottawa, ON*) – FuseTalk Inc., provider of online collaboration solutions for ColdFusion environments, has announced the release of Virtual Meeting for FuseTalk Community Edition. The three modules that comprise Virtual Meeting bring the power of real-time collaboration through streaming video/audio, chat, and whiteboarding to the easy familiarity of discussion forums powered by FuseTalk.

Designed for FuseTalk Community Edition, Virtual Meeting uses Flash MX and the new Flash Communication Server, and is the first commercially available application to deliver on the promise of rich Internet content. By leveraging the capabilities of Flash MX, Virtual Meeting delivers a thin-client approach that radically reduces the burden of managing client software and the uncertainty of purchasing per-seat licensing.

Virtual Meeting's architecture also allows companies to use a single common port, thereby minimizing port allocation problems, firewall issues, and security concerns while making it easy for meeting participants outside the firewall to join in.

"Virtual Meeting delivers on the promise of real-time Web collaboration without the high cost and technology challenges," says Corien Kershey, president and CEO of FuseTalk Inc. "Most people today are comfortable using discussion forums so they are a natural environment for creating a personal and interactive experience. Virtual Meeting combined with FuseTalk Community Edition reduces the costs of travel, and builds team unity while keeping the costs of collaboration low."

Virtual Meeting consists of three modules that provide *n*-way, streaming video/audio, chat, and whiteboarding. The only client required is Flash 6. Unlike many other video/audio solutions, no special hardware other than a simple webcam and microphone are needed to join in. For more information, visit www.fusetalk.com.