SYS-CON

FIVE COOL THINGS I'VE DONE WITH COLDFUSION PG 7

# COLDFUSION Developer's Journal

ColdFusionJournal.com

October 2006 Volume:8 Issue: 10

Presorted
Standard
US Postage
PAID
St. Croix Press

www.AjaxWorldExpo.com

AJAXWORLDEAST™ CONFERENCE & EXPO

SEE PAGE 43

## ColdFusion Hosting is our Complete Focus

### POWERFUL HOSTING PLANS
FREE SQL server access | FREE account setup | Unlimited email accounts Generous disk space & data transfer
30 day money-back guarantee | Great value

### RELIABLE NETWORK
99.99% average uptime | State-of-the-art data center with complete redundancy in power, HVAC, fire suppression, bandwidth and security | 24/7 network monitoring

### FANTASTIC SUPPORT SERVICES
24/7 suppport services | Knowledgeable phone support | We focus on your individual needs

**CFDynamics**

866.233.9626 CFDYNAMICS.COM

For years we have been involved in the Cold Fusion community and have come to know what developers and project managers look for in a web host. The combination of our powerful hosting plans, reliable network, and fantastic suppport sets us apart from other hosts.
***Real service. Real satisfaction. Real value. Real support. Real Freedom.***

macromedia ALLIANCE PARTNER

BlueDragonCertified V.I.P. Hosting Partner

< Guru: Logic/ >

A shortcut doesn't take away the fun.

It is the fun.

Get the free Adobe® Flex™ trial beta at adobe.com/flex.

The fastest way to create rich Internet applications.

Real time. Cross-platform.

Expressive. Standards-based programming model. Data intensive app.

Extend Ajax. And go beyond.

< Download Adobe Flex now/ >

Adobe

## For answers call us at 1-866-EDGEWEB

3 3 4 3 9 3 2

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

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

### By the Numbers:

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

EDGE WEB HOSTING

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

http://edgewebhosting.net

CFDJ READERS' CHOICE AWARD · MXDJ READERS CHOICE AWARD

2003 - 2006

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

COLDFUSION Developer's Journal

**editorial**

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

**technical editor**
Raymond Camden raymond@sys-con.com

**executive editor**
Nancy Valentine nancy@sys-con.com

**associate editor**
Lauren Genovesi laureng@sys-con.com

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

**production**

**lead designer**
Abraham Addo abraham@sys-con.com

**art director**
Alex Botero alex@sys-con.com

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

**graphic designer**
Mandy Eckman mandy@sys-con.co

**editorial offices**

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

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

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

# Adobe University Evangelists

By Simon Horwith

Spending as much time as I have speaking before audiences, I try very hard to keep an eye on trends and attitudes within demographic groups, including university students. I have also had the opportunity in the past to represent Macromedia as a lecturer at Ivy League universities.

My general experience – when talking with computer science students about the tools, technologies, and programming languages that they are exposed to in school – is that Macromedia wasn't really doing a very good job ensuring that graduates entered the real world with knowledge of or exposure to their products. This was (sadly) particularly true for ColdFusion.

I was very happy to read Mike Potter's recent blog entry (http://blogs.adobe.com/mikepotter/2006/09/were_looking_fo.html) about a program I've never heard of: "Adobe Ambassadors." I don't know many of the details, but it's an Adobe program (I think it's a new program but I'm not sure) that creates and supports Adobe technology evangelists on university campuses.

I can't begin to describe how relieved and impressed I am at the thought of Adobe technical evangelists promoting the products, spreading technical information, and relaying student population feedback to Adobe from university campuses.

If anyone from Adobe who is involved in this program reads this editorial, I not only give you praise but also wish to extend an invitation to leverage any services I may in order to assist the program and its members.

## About the Author
*Simon Horwith is the editor-in-chief of* ColdFusion Developer's Journal *and is the CIO at AboutWeb, LLC, a Washington, DC based company specializing in staff augmentation, consulting, and training. Simon is a Macromedia Certified Master Instructor and is a member of Team Macromedia. He has been using ColdFusion since version 1.5 and specializes in ColdFusion application architecture, including architecting applications that integrate with Java, Flash, Flex, and a myriad of other technologies. In addition to presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com.*
*simon@horwith.com*

# CF_UNDERGROUND VIII PRE-MAX EVENT

```
<CF_BEER
  FROM = "[imported/domestic]"
  COLOR = "[light/dark]"
  FERMENT = "[lager/ale/bock]"
  FLAVOR = "[porter/stout]"
  AUTOREFILL = "[on/off]">
```

**DATE: SUNDAY, OCTOBER 22, 2006**

**LOCATION: THE BEACH, LAS VEGAS**

*Speakers include:*
*Simon Horwith*
*Michael Smith*
*Glenda Vigoreaux*
*and more...*

365 Convention Center Drive, Las Vegas, Nevada

Meet with old friends and new before the MAX event. A full day of networking, ColdFusion, great talks, drinks, food, and fun.

WWW.CF-UNDERGROUND.COM

only **$99**

TeraTech
www.teratech.com

COLDFUSION Developer's Journal

# Five Cool Things I've Done with ColdFusion

## Mailing labels, bar codes, credit cards, and graphing

By Jeffry Houser

I'm at my best when I'm challenged. In my consulting business I tend to gravitate towards small businesses with delusions of grandeur. I want to be the one to help them realize their vision and turn their delusion into reality. Looking back, this has been an interesting week. I thought I'd share some of the interesting things I've done and how ColdFusion helped make them possible.

### Monday: Bar Codes

I've been working over the past year to Web enable an MS Access based scheduling system. One item of the application generates attendance cards with a bar code on it for all users. When a user shows up for a class, that user will scan in, and the information is used to track attendance. The client wanted to be able to generate the attendance cards (with bar codes) from the Web. How do you do that?

First, you have to realize what bar codes are. They are just text, or data of some sort, that looks like a bunch of parallel lines. If you don't know what I mean, just flip over your copy of ColdFusion: The Complete Reference, and you'll find one. (Or you can look at any book or CD). These bar codes are generated using a special font. The client provided me with the font, named "3 of 9 Barcode." A simple Web search will reveal many places where you can get this font. I recommend downloading it straight from the source at http://www.squaregear.net/fonts/free3of9.shtml.

Now you're probably thinking that since you have the font name, you can just use the HTML font tag like this:

```
<font name="3 of 9 Barcode">My Bar Code</font>
```

And you don't even have to use ColdFusion, right? Well, you're on the right track, but things aren't that simple. The first mistake is that you need to put an asterisk (*) before and after the text you want to be able to scan as a bar code. Even with that tidbit of information, you still aren't in the clear. For this approach to work, the bar code font must be installed on the client computer.

When you use a font in HTML like that, that font must be installed on the client computer (not the server), or else a default font is used. The resulting bar code won't scan if it's generated using Helvetica or Times New Roman. (You can trust me on that one.) This font is not a standard font installed on most users computers. Expecting it to be isn't a realistic expectation for Web development. Most people don't have uses for Bar Code fonts in their daily activities. Getting the bar code font installed on all relevant clients of your system would bring you back to the old days of client/server technology where you have to walk through a big dark office building with fluorescent lighting and spend months installing things so the requirements change by the time you're finished with the roll out. There must be a better way, right? Yes!

The answer is to use the ColdFusion Report Builder to generate the attendance cards with bar codes. First, install the bar code font on your development machine and on your production (and/or testing) server. Then, create a new report (or open an existing report). Create a field and give it an expression something like this:

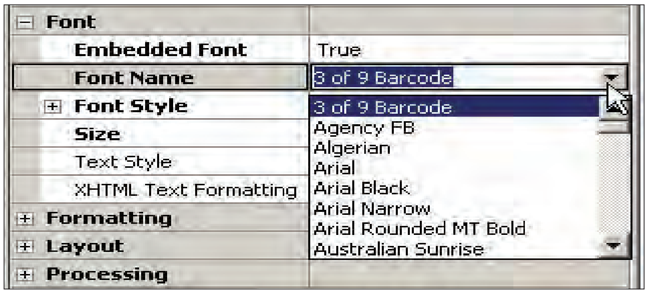
```
"*" & barcodedata & "*"
```

In the properties dialog, you should see the 3 of 9 Barcode font listed in the Font name list, as shown in Figure 1. Select that

## TABLE OF CONTENTS

font and your field will look like a bar code. Above the font selection, there is another option for "embedded font." Be sure to select true for this option. Normally, you wouldn't want to embed fonts in a PDF because it makes the overall file size larger, but then you have to ensure that the client computer has the font installed, or it will default itself. Since we don't want to make sure that the client computer has the bar code font installed (for reasons discussed earlier), we embedded it here.

To test the bar codes, you have to scan them. I used a CueCat. The CueCat company has a long disastrous history which results in CueCat scanners being an extremely cheap buy on eBay. Perform a Web search for all the gory details; they are beyond the scope of this article.


Font
Embedded Font: True
Font Name: 3 of 9 Barcode
Font Style
Size
Text Style
XHTML Text Formatting
Formatting
Layout
Processing

3 of 9 Barcode
Agency FB
Algerian
Arial
Arial Black
Arial Narrow
Arial Rounded MT Bold
Australian Sunrise

Figure 1

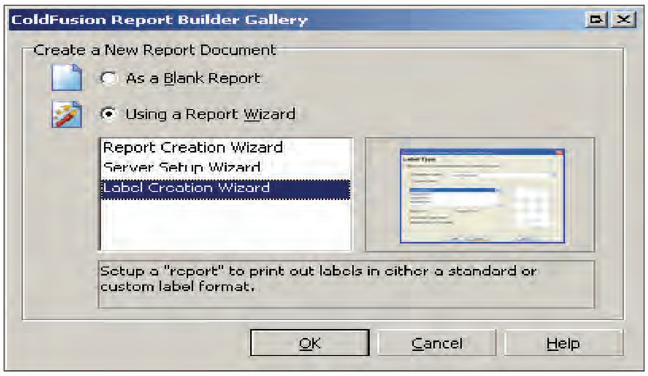

ColdFusion Report Builder Gallery
Create a New Report Document
As a Blank Report
Using a Report Wizard
Report Creation Wizard
Server Setup Wizard
Label Creation Wizard
Setup a "report" to print out labels in either a standard or custom label format.

Figure 2: New on the CF Report Builder

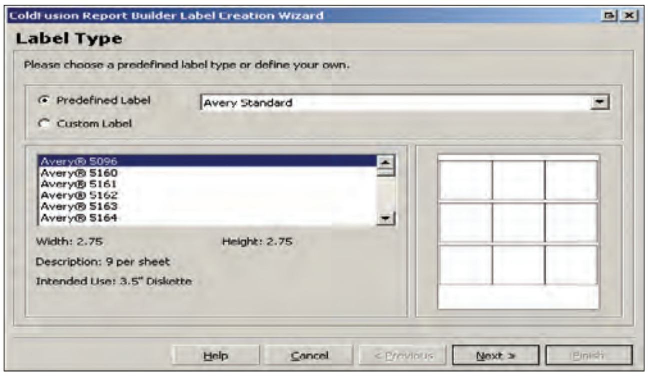

ColdFusion Report Builder Label Creation Wizard
Label Type
Please choose a predefined label type or define your own.
Predefined Label: Avery Standard
Custom Label
Avery® 5096
Avery® 5160
Avery® 5161
Avery® 5162
Avery® 5163
Avery® 5164
Width: 2.75 Height: 2.75
Description: 9 per sheet
Intended Use: 3.5" Diskette

Figure 3: The first step in the Label Creation Wizard

## Tuesday: Mailing Labels

The mid-week point brought another challenge from above. A client wanted to be able to generate mailing labels. Doing this sort of task with HTML is grueling, tedious, and prone to errors. Especially since different browsers can display the same HTML code in different ways. HTML is not designed for print, and forcing what is on screen to line up with what is on your label sheet is going to be the cause some serious headaches. So, what are the options?

I turned once again to my new best friend, the CF Report Builder. The nice man, Dean, at cfreport.org (no last name is known, but I get the impression he works for Adobe) has created a label wizard for the ColdFusion Report Builder. You can download the wizard at http://www.cfreport.org/index.cfm?mode=entry&entry=38BBAD81-3048-2D03-0A777CDC2FB60D74. You'll need to have at least 7.01 of the Report Builder installed (and I recommend getting the 7.02 update). In my case, I already had the wizard installed. Since I don't remember installing it, it may have been included in the 7.02 update of the Report Builder. Check your install and you may not have to worry about manually downloading and installing the wizard.

You can find out if you have the label wizard installed by selecting File ‡ New. You should see the Label Creation Wizard as an option in the new dialog, as shown in Figure 2. Select the label creator wizard and click next.

You'll see the first step in the Label Creation Wizard, as shown in Figure 3. Select the label template you want to use. Most labels are based on some form of Avery label template (My client asked for 5160). Select that and click next.

Now you have to define the query fields you want to use to display information on your label, as shown in Figure 4. I manually entered them (The query builder button didn't work for me). Click the next button

The final step in the report builder is to define the layout of your label. Double-click the elements in the left window, and they'll be entered in the right window. The right window is an expression window, but you don't have to worry about writing an expression, because when you click the finish button, text (such as commas) will automatically be escaped when creating the finished template. Save your template and you're good to go.

The last step in the Report Builder is just to run the report using the cfreport tag. Your template will look something like this:

```
<cfquery name="Getusers" datasource="myDSN">
 select firstName, LastName, State, City,. Zip
 from users
</cfquery>

<cfreport format="PDF" template="LabelReport.cfr" query="Getusers">
</cfreport>
```

– CONTINUED ON PAGE 20

## Meet Robert

### A business executive at a popular social media site

### Challenges

- Encountering poor video quality due to exponential growth in traffic.
- Increasing bandwidth costs due to growing audience size.
- Experiencing compatibility issues due to user-generated video arriving in multiple formats.

### Solutions

- VitalStream Streaming Services – Improved quality of end-user video experience using a scalable and global content delivery network.
- VitalStream Advertising Services – Transformed the delivery of digital media from a cost center into a profit center.
- VitalStream Transcoding Services – Automatically converted all user-generated content into the leading streaming media format.

### Providing End-to-End Solutions for Your Business

VitalStream is more than a rich media delivery company. We are your professional partner providing solutions that meet your unique business challenges. To learn more about VitalStream solutions, call 800-254-7554 or visit www.vitalstream.com/go/solutions/

VitalStream®

*Digital Media Solutions for Your Business*

# Rock Solid Storage via Web Services

By Joe Danziger

Storage and bandwidth – these have traditionally been the two hardest things to scale up as an application grows. Many a dot-com million has been spent building out rock solid storage infrastructures, sometimes for applications which never saw the light of day. Fast forward to 2006. These are good times for developers, with open source technologies, open APIs, and low-cost, commoditized services to help the little guys play along with the big boys.

If you've struggled with adding storage to your application, Amazon's new Simple Storage Service may be just what you're looking for. Amazon S3, as it is commonly known, provides developers with access to the same rock solid, enterprise-class storage that Amazon uses for its own network of sites – all at a low cost with no minimum usage amounts.

Amazon has always tried to define itself as being more than just a bookstore. It was one of the first companies to embrace web services and to create a community around these developers. The launch of S3 has taken this even further. It's now possible to launch your app without worrying about what happens when you wind up with the next big Web 2.0 phenomenon.

Some of the features of S3 are:

- It allows you to read, write and delete an unlimited number of objects from 1 byte to 5gb each
- You pay only for what you use - $0.15 per GB-Month of storage and $0.20 per GB of data transferred
- It uses standards-based REST (Representational State Transfer) and SOAP interfaces
- Objects can be made public or private, and links can be set to expire
- It is scalable, reliable and fast

The S3 web services are made available as either SOAP or REST-based web services. SOAP was the first web services technology to gain acceptance, but REST has quickly caught up due to the simple interface it provides using standard HTTP requests. REST does not suffer from the additional messaging layer and overhead that SOAP needs, and it is generally more lightweight and easier to work with. We're going to use the REST interface for all of our methods, using the CFHTTP tag and subsequently parsing the XML that is returned.

The following operations make up the core of the S3 service:

- List all buckets for user
- Put a new bucket
- Get contents of a bucket
- Delete a bucket
- Put a new object
- Get an object
- Delete an object

Some of the things to keep in mind about working with the S3 service:

- Accounts are limited to 100 buckets each and bucket names must be unique across all users
- Buckets cannot contain other buckets, only objects
- The number of objects within a bucket is unlimited

Detailing each method is outside the scope of this article, but we'll look at a sample REST request to give you an idea of how things work. You should be able to take it from there.

One of the keys to interacting with the S3 service is the signature that is required to be sent with each request. This is created by using a hash of your Amazon secret access key combined with the HTTP request string for the action being performed. To create the necessary signature, you'll need the <CF_HMAC> custom tag created by Tim McCarthy, which can be downloaded from the Adobe Developer Exchange at http://tinyurl.com/jhn9b.

The code to create a new bucket can be found in Listing 1. CF7 includes all the other tags you'll need, but, on CF6, a Hex2Bin function is necessary which you can find at http://tinyurl.com/knx22. With CF6, you should change line 17 to read:

```
<cfset signature = ToBase64(Hex2Bin("#digest#"))>
```

The signature gets created on line 14, which generates a hash of the canonical string being sent to the web server using

Adobe
Solution Partner
2006

INTERGRAL
information solutions

# <cf_essentials>

## Indispensable CFMX tools.

**FusionReactor**

**CFMX / J2EE Server, database and application monitoring**

from **$249**

**FusionDebug**

**Interactive Debugging for ColdFusion MX**

from **$99***

fusion
**www.fusion-reactor.com**

* Using GOT2DEBUG discount coupon. Offer ends October 31st.

the secretAccessKey provided by Amazon as the encryption key. Once the signature is created, we submit the REST request via <cfhttp> on lines 20-24, and pass the appropriate <cfhttp-param> fields into the header of the request. The results are returned as standard XML. The main difference between the various S3 methods lies in the canonical string that gets sent along with the header variables passed in. Amazon provides raw HTTP samples in their Developer Resources which can help with forming the necessary HTTP requests. The raw HTTP request for putting a bucket is found in Listing 2.

Rather than have everyone reinvent the wheel, I created a CFC wrapper to the S3 service which handles all of the basics of adding and deleting buckets and objects. You can download the S3 CFC, which contains a sample application for interacting with the service, from the projects page on my blog at http://www.ajaxcf.com.

Keep in mind that this is a new service just launched by Amazon in March 2006. It is still evolving and more functionality will be added in the future. To find out more about Amazon S3 and to create a web services account to get started, visit the S3 homepage at http://aws.amazon.com/s3/. There are some alternatives on the horizon from OmniDrive, Streamload, and Box.net, which may suit your needs as well.

## About the Author

*Joe Danziger is a senior web applications developer with Multimax, Inc., a provider of enterprise IT services and solutions supporting the critical missions of the Air Force, Army, Navy, and other Department of Defense components. He is certified as an Advanced Macromedia ColdFusion MX Developer, and maintains a blog on ColdFusion and Ajax development at http://www.ajaxcf.com.*

*joe@ajaxcf.com*

### LISTING 1: Creating A New Bucket

```
1:   <cffunction name="putBucket" access="public" output="false"
returntype="boolean" description="Creates a bucket.">
2:      <cfargument name="bucketName" type="string" required="yes">
3:
4:      <cfset var signature = "">
5:      <cfset var dateTimeString = GetHTTPTimeString(Now())>
6:
7:      <!--- Create a canonical string to send based on operation
        requested --->
8:      <cfset var cs = "PUT\n\ntext/html\n#dateTimeString#\
   n/#arguments.bucketName#">
9:
10:     <!--- Replace "\n" with "chr(10) to get a correct digest
--->
11:     <cfset var fixedData = replace(cs,"\n","#chr(10)#","all")>
12:
13:     <!--- Calculate the hash of the information --->
14:     <cf_hmac hash_function="sha1" data="#fixedData#"
  key="#variables.secretAccessKey#">
15:
16:     <!--- fix the returned data to be a proper signature --->
17:     <cfset signature = ToBase64(binaryDecode(digest,"hex"))>
18:
19:     <!--- put the bucket via REST --->
20:     <cfhttp method="PUT" url="http://s3.amazonaws.com/#arguments.
    bucketName#" charset="utf-8">
21:        <cfhttpparam type="header" name="Content-Type"
value="text/html">
22:        <cfhttpparam type="header" name="Date" value="#date
  TimeString#">
23:        <cfhttpparam type="header" name="Authorization"
value="AWS #variables.accessKeyId#:#signature#">
24:     </cfhttp>
25:
26:     <cfreturn true>
27:  </cffunction>
```

### Listing 2: HTTP Create Bucket Request

```
PUT /[bucket-name] HTTP/1.0
Date: Wed, 08 Mar 2006 04:06:15 GMT
Authorization: AWS [aws-access-key-id]:[header-signature]
Host: s3.amazonaws.com
```

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

## CFDJ Advertiser Index

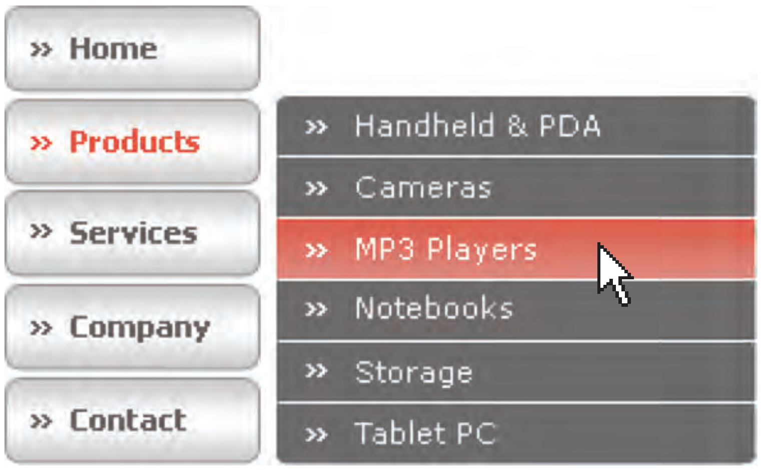
| ADVERTISER | URL | PHONE | PAGE |
|---|---|---|---|
| CFDynamics | www.cfdynamics.com | 866-233-9629 | 2 |
| Adobe | www.adobe.com | | 3 |
| EdgeWebHosting | http://edgewebhosting.net | 866-334-3932 | 4 |
| CF_Underground | www.CF-Underground.com | | 6 |
| Vitalstream | www.vitalstream.com | 800-254-7554 | 10,11 |
| Intergral | www.fusion-reactor.com | | 13 |
| Interakt | www.interaktonline.com/mxcss | | 15 |
| CommunityMX | www.communitymx.com/trial | | 25 |
| Adobe MAX | www.adobe.com/go/cfdj | | 27 |
| AJAXWorld East | www.AjaxWorldExpo.com | 201-802-3020 | 43 |
| Hot Banana | www.hotbanana.com | 866-296-1803 | 49 |
| Hal Helms | www.halhelms.com | | 49 |
| Straker Interactive | www.zoomflex.com | (64 9) 309-5252 | 51 |
| HostMySite | www.hostmysite.com | | 52 |

# The new MX CSS Menus2 for Dreamweaver

## Create and customize CSS menus in Dreamweaver !

The new MX CSS Menus 2 from InterAKT is a Dreamweaver extension that offers you the choice to create your own CSS menu, implement your own ideas in a customized skin and perfectly blend your menu into your site's design. With visual step by step implementation and no CSS knowledge required this new version includes:

- CSS Skin Editor to create your own design
- Support for HTML and database driven sites
- Faster loading in the browser
- Vertical, tabbed, horizontal or expandable menu types
- 23 pre-built CSS skins
- PNG and PSD files included
- Animation effects on sub-levels
- New wizards and interfaces



**Buy at only $99**

Free trial available at: **http://www.interaktonline.com/mxcss**

Simplifying Web Development

Interakt

# Encapsulating Recordsets

## Do you need getters and setters for your recordsets?

By Peter Bell

One of the first things that you encounter when moving to object-oriented (OO) programming are beans. Beans are simple representations of a business object (like a user or a product) that hide all of the information stored in the bean behind methods (functions) for getting and setting the information (called, unsurprisingly, getters and setters).

Typically, if you want to display a product view screen, you'll get the product information from the database, load the resulting query recordset into a bean and then, instead of displaying the variables from the query, call the methods from the bean. In the past your product view may have looked like this:

```
<cfoutput query="GetProduct">
    #Title#<br />
    $#Price#<br /><br />
</cfoutput>
```

With a bean, the product view will change to the following:

```
<cfoutput>
    #Product.get("Title")#<br />
    $#Product.get("Price")#<br /><br />
</cfoutput>
```

The only difference is that you are calling a "get" method that hides (encapsulates) the way the title and the price are retrieved, but this small difference can have a big impact on the maintainability of your code. (The sharp eyed among you will have noticed that I'm using a generic getter. The sidebar explains why this might be a good idea for you to consider).

### Using Generic Getters and Setters

The problem with writing getters and setters is that it is just so tedious. If you have an object with a hundred attributes, you are going to have to write two hundred getter/setter methods for that object and the vast majority of them will do nothing but get or set a specific value within the bean.

Of course, you could use a passive code generator to create your beans, but as the business requirements change (and they will change), you will be responsible for adding, renaming and deleting the getters and setters every time someone wants to change the properties for an object.

You could use an active generator and regenerate your getters and setters every time the object model changes, but you'd still have thousands of lines of extra code in your application, and the more code you have, the more likely it is to become a problem (plus your application is now dependent on a separate generator to continue to be maintainable).

With a generic getter and setter, there is a single get method and a single set method for each object (usually inherited from a base business object class). A well designed getter/setter will check against an attribute list to make sure you are allowed to get or set the attribute. For example, a User bean may be capable of displaying the user's age by calculating the age based on their date of birth. You would want to provide the ability to get the calculated age property, but you would only want to allow the application to set the date of birth property (not the age), as that would be the value actually stored in the database.

The generic getter/setter should then look to see if there is a custom get/set method. For example, there may well be a getAge() method that calculates the user's age based on their date of birth. If there is a custom getter/setter, the generic method should just call that method and return whatever it receives as a response. If there isn't, the method should get or set the appropriate value within the beans "variables" scope.

Generic getters and setters combined with a gettableAttributeList and settableAttributeList provide all of the encapsulation and customization benefits of manually created getters and setters in a much simpler, more maintainable package. A simple generic getter and setter is included as part of the IBO in Listing 1.

## Why Encapsulate?

Object-oriented programming is all about writing more maintainable code. Many Object Oriented patterns take longer to code initially than a traditional procedural approach, but they are easier to maintain. Given that we tend to spend much more time maintaining than developing applications, this is usually a worthwhile trade-off.

The benefit of encapsulating business object attributes behind getters and setters is that if the logic required to calculate (or save) the attribute changes, you only have to change the code in one place. Let's say that the product price is calculated based on a sale price. In the past you might have put that logic right into the product view:

```
<cfoutput query="GetProduct">
    #Title#<br />
    $<cfif SalePrice>#SalePrice#<cfelse>#Price#</cfif><br /><br />
</cfoutput>
```

The problem is, there may also be a product list screen, a product search results screen, and the cart and checkout may also need to know the price of a product, so if you changed the way you calculated the price, you would have to remember to make that change in five places. Even worse, there is the risk that you might forget to update one of the screens, so products displayed on the search screen would display a different price than when added to the basket.

With getters and setters encapsulating the logic, there would be no change at all to the display screens. They would still be:

```
<cfoutput>
    #Product.get("Title")#<br />
    $#Product.get("Price")#<br /><br />
</cfoutput>
```

The only difference is that you would change the getPrice() custom method within the product object to be something like:

```
<cffunction name="getPrice" returntype="numeric" access="public"
output="no" hint="I return the price to display and charge for the
product">
    <cfset var Local = structNew()>
    <cfif variables.get("SalePrice")>
            <cfset Local.ReturnValue = variables.get("SalePrice")>
    <cfelse>
            <cfset Local.ReturnValue = variables.Price>
    </cfif>
    <cfreturn Local.ReturnValue>
</cffunction>
```

So, encapsulation of getters and setters using beans is a key element of Object Oriented programming, and it is a best practice that most OO developers would use most of the time. So, what happens when we have more than one user or product or article?

## The Performance Problem

In Java, this would be easy. You would take your recordset, use it to create a collection of objects (one for each record), and you would still have all of the benefits of encapsulated getting and setting of properties. Unfortunately, in ColdFusion, object creation is expensive (in terms of performance). Creating a large number of objects as part of every page load can substantially affect the performance of a ColdFusion script.

Because of this, a lot of ColdFusion developers currently use recordsets for displaying their lists, losing all of the benefits of encapsulation. If they want to change the way the price is calculated for a product list, they either need to push back the price calculation onto the database; write a script in their service object to loop through the query implementing any custom data transformations; or replicate their business logic all over their list screens.

## Introducing the Iterating Business Object

The iterating business object (IBO) is a simple solution to the problem of encapsulating access to getters and setters for recordsets without sacrificing performance. It is implemented as a base class that your business objects can extend. As well as providing generic getters and setters, it can also load itself with a query, and it provides a basic set of iterating methods (first(), next(), and isLast()) for looping through the recordset.

The IBO allows you to get all of the benefits of encapsulated getting and setting of attributes – whether you are working with a single object or a collection of them. It also allows you to use exactly the same code for both your single objects and object collections. For example, it really doesn't matter whether you are viewing a single product on a product detail page or a list of products matching a search query. If you want to display the price for the product, you want to use the same business logic to calculate it in both cases. With an IBO, you always load the same object up with the results from your query, and you can maintain all of your getter and setter calculations in that one object. In addition, you are only instantiating a single object per page view, so the performance hit is nominal. You can see the code for a simple IBO in Listing 1.

## Using the Iterating Business Object

When you are using the IBO to display a single record, it is exactly the same as using a bean:

```
<cfoutput>
    #Product.get("Title")#<br />
    $#Product.get("Price")#<br /><br />
</cfoutput>
```

If you want to display a list of records, it is almost as easy. You take the same basic display code and just wrap it with a loop based on the iterating methods of the object:

```
<cfset Product.first()>
<cfloop condition="NOT #Product.isLast()#">
<cfoutput>
    #Product.get("Title")#<br />
            $#Product.get("Price")#<br /><br />
</cfoutput>
<cfset Product.Next()>
</cfloop>
```

## Improving the Object

Previously I showed an example of a custom getter for calculating the price of a product. If there was a sale price, it displayed that, otherwise it just accessed the variables scope to get the price. Well, that works for a single record, but if there are multiple records (as with the IBO), the actual code to access a particular record would be more like:

```
<cfset Local.ReturnValue = variables.Data[variables.IteratorCount].Price>
```

Where the variables.IteratorCount is a pointer to the current record within the recordset being displayed. This isn't "wrong", but over time all of the custom methods would have to include this code, and if I ever decided to change the structure of the data storage within the variables scope, all of those methods would have to be rewritten.

To encapsulate that change, I added one more pair of generic methods to the base IBO: access() and mutate(). Accessors and mutators are alternate terms for getters and setters, but these methods are private and are only to be used by the customer getters and setters to access the underlying data structure, so if I ever wished to change the structure, I would only have to change those two methods. Below is simplified code for the access() method (the full code is available in Listing 1):

```
<cffunction name="access" returntype="any" access="private" output="no"
hint="I encapsulate access to the attribute values within this object,
accepting an attribute name and returning the appropriate attribute
value.">
    <cfargument name="AttributeName" type="string" required="yes" hint="I
am the name of the attribute to return the value for">
    <cfset var Local = structNew()>
<cfset Local.ReturnValue = variables.Data[variables.
IteratorCount][arguments.AttributeName] >
    <cfreturn Local.ReturnValue>
</cffunction>
```

With this in place, we can now simplify the data access in the custom methods to just:

```
<cfset Local.ReturnValue = variables.access("Price")>
```

## Conclusion

I have used the IBO on a number of production projects over the last couple of months. One project in particular required almost daily changes to the (very complex) object model as we continually added new attributes and changed the rules for calculating existing attributes. I found the application extremely easy to maintain and it is now in production, powering over 20 different sites and performing very well under load.

If you don't have any calculations for getting or setting any of your object attributes and really don't expect any in the future, any kind of encapsulation is overkill, and you would be better just to display using recordsets and cfoutput – for one record or for many. If you do have calculated fields, give the IBO a try and see how it works for your projects.

### About the Author

*Peter Bell is CEO/CTO of SystemsForge (http://ww.systemsforge.com) and helps Web designers to increase their profits and build a residual income by generating custom web applications – in minutes, not months. An experienced entrepreneur and software architect with fifteen years of business experience, he lectures and blogs extensively on application generation and ColdFusion design patterns (http://www.pbell.com).*

*pbell@systemsforge.com*

### Listing 1.

```
<cfcomponent displayname="Base Object" hint="I provide the base methods
for Iterating Business Objects (IBOs). DO NOT EDIT.">

<cffunction name="init" returntype="BaseObject" access="public"
output="false" hint="I initialize the base object with application
specific parameters.">
  <cfscript>
      variables.IteratorRecord = 1;
      variables.gettableAttributeList = "*";
      variables.settableAttributeList = "*";
  </cfscript>
  <cfreturn This>
</cffunction>

<cffunction name="get" returntype="any" access="public" output="false"
hint="I am a generic getter that handles getting of all attributes.
I confirm they are gettable, try to use a custom method, and if that
fails, I just ask access() to get the attribute value directly.">
  <cfargument name="AttributeName" type="string" required="yes"
hint="The name of the attribute to get">
  <cfscript>
      var Local = StructNew();
      If (ListFindNoCase(variables.gettableAttributeList,arguments.
AttributeName) OR variables.gettableAttributeList EQ "*")
      {
          // The method should be run
          if (structKeyExists(variables,"get#arguments.AttributeName#"))
          {
              // There is a custom method, use it
              Local.ReturnValue = evaluate("variables.get#arguments.
AttributeName#()");
          }
          Else
          {
              // Otherwise just pull the attribute using private "ac-
cess" method
              Local.ReturnValue = variables.access(arguments.AttributeN-
ame);
          };
      }
      Else
      {
          // The attribute is private (or invalid)
          Local.ReturnValue = "";
      };
  </cfscript>
  <cfreturn Local.ReturnValue>
</cffunction>

<cffunction name="set" returntype="any" access="public" output="false"
hint="I am a generic setter that handles setting of all attributes.
I confirm they are settable, try to use a custom method, and if that
fails, I just ask mutate() to set the attribute value directly.">
  <cfargument name="AttributeName" type="string" required="yes"
hint="The name of the attribute to set">
  <cfargument name="AttributeValue" type="any" required="yes" hint="The
value of the attribute to set">
  <cfscript>
      var Local = StructNew();
      If (ListFindNoCase(variables.settableAttributeList,arguments.
AttributeName) OR variables.settableAttributeList EQ "*")
      {
         // The method should be run
         if (structKeyExists(variables,"set#arguments.AttributeName#"))
         {
            // There is a custom method, use it
            Local.ReturnValue = evaluate("variables.set#arguments.
AttributeName#(arguments.AttributeValue)");
         }
         Else
         {
            // Otherwise just set the attribute using private "mutate"
method
            Local.ReturnValue = variables.mutate(arguments.
AttributeName,arguments.AttributeValue);
         };
      }
      Else
      {
         // The attribute is private (or invalid)
         Local.ReturnValue = "";
      };
  </cfscript>
  <cfreturn Local.ReturnValue>
</cffunction>

<cffunction name="access" returntype="any" access="private"
output="false" hint=" I encapsulate access to the attribute values
within this object, accepting an attribute name and returning the ap-
propriate attribute value.">
  <cfargument name="AttributeName" type="string" required="yes"
hint="The name of the attribute to get">
  <cfset var Local = StructNew()>
  <cftry>
      <cfscript>
         // Just pull the attribute
         Local.ReturnValue = variables.Data[variables.IteratorRecord][a
rguments.AttributeName];
      </cfscript>
      <cfcatch>
         <cfset Local.ReturnValue = "">
      </cfcatch>
  </cftry>
  <cfreturn Local.ReturnValue>
</cffunction>

<cffunction name="mutate" returntype="any" access="private"
output="false" hint=" I encapsulate access to the attribute values
within this object, accepting an attribute name and setting it to the
provided attribute value.">
  <cfargument name="AttributeName" type="string" required="yes"
hint="">
  <cfargument name="AttributeValue" type="string" required="yes"
hint="">
  <cfset var Local = StructNew()>
  <cftry>
  <cfscript>
      // Just set the attribute
      variables.Data[variables.IteratorRecord][arguments.AttributeName]
= arguments.AttributeValue;
  </cfscript>
      <cfcatch>
         <cfset Local.ReturnValue = "">
      </cfcatch>
  </cftry>
  <cfreturn Local.ReturnValue>
</cffunction>

<cffunction name="loadQuery" returntype="numeric" access="package"
output="false" hint="I take a query and use it to load the business
object with its data.">
  <cfargument name="Recordset" type="query" required="yes"
displayname="Recordset" hint="I am the query that needs to be loaded.">
  <cfscript>
     // Based on code by Peter J. Farrell (pjf@maestropublishing.com)
via cflib.org
      var theQuery = arguments.Recordset;
      var theStructure = StructNew();
      var cols = ListToArray(theQuery.columnlist);
      var row = 1;
      var thisRow = "";
      var col = 1;
      If (theQuery.recordcount LT 1)
      {
         theStructure[row] = variables.setDefaultValues(theQuery.
columnlist);
         THIS.Recordset = theStructure[row];
      }
      Else
      {
         for(row = 1; row LTE theQuery.recordcount; row = row + 1)
         {
            thisRow = StructNew();
            for(col = 1; col LTE arraylen(cols); col = col + 1)
            {
               thisRow[cols[col]] = theQuery[cols[col]][row];
            }
            theStructure[row] = Duplicate(thisRow);
            THIS.Recordset = arguments.Recordset;
         }
      };
      variables.Data = theStructure;
      variables.NumberofRecords = Row;
      THIS.PropertyNameList = theQuery.columnlist;
      THIS.NumberofRecords = Row - 1;
  </cfscript>
  <cfreturn variables.NumberofRecords>
</cffunction>

<cffunction name="First" returntype="void" access="public"
output="false" hint="I set the business object to point to the first
record as part of the iterator functionality.">
  <cfscript>
      variables.IteratorRecord = 1;
  </cfscript>
</cffunction>

<cffunction name="Next" returntype="boolean" access="public"
output="false" hint="I increment the position of the business object
iterator, returning success or failure as a boolean. If failure (al-
ready on final record), I leave the count the same.">
  <cfscript>
      // Declare a local structure for all local variables
      var Local = StructNew();
      If (variables.IteratorRecord GTE variables.NumberofRecords)
      {
         Local.ReturnBoolean = false;
         variables.IteratorRecord = variables.NumberofRecords;
      }
      Else
      {
         Local.ReturnBoolean = false;
         variables.IteratorRecord = variables.IteratorRecord + 1;
      };
  </cfscript>
  <cfreturn Local.ReturnBoolean>
</cffunction>

<cffunction name="isLast" returntype="boolean" access="public"
output="false" hint="I return whether the business object is displaying
the last record or not as part of the iterator functionality.">
  <cfscript>
      // Declare a local structure for all local variables
      var Local = StructNew();
      If (variables.IteratorRecord EQ variables.NumberofRecords)
         Local.ReturnBoolean = true;
      Else
         Local.ReturnBoolean = false;
  </cfscript>
  <cfreturn Local.ReturnBoolean>
</cffunction>

</cfcomponent>
```

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

# Five Cool Things I've Done with ColdFusion

– CONTINUED FROM PAGE 9

Your query will most likely be different, depending on the information you want to print on a label, but you should be able to apply these concepts to your own development.

## Wednesday: Credit Card Swipes

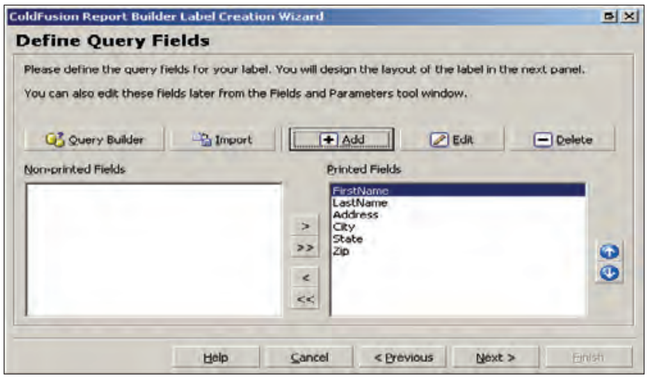
One of my clients runs an e-commerce business, and the site implementation is about as classic as an on-line shopping cart can come. The client is in the process of opening up a retail store, and I spent some time modifying the shopping cart to accommodate retail transactions. The biggest difference between retail transactions and Web-based transactions is that you have a credit card to swipe. Swiped cards get a much better merchant rate than Internet or phone orders received, so that is an added benefit of the retail store vs. the Internet store.

The client provided me with a USB credit card swiper so that I could test the process. As far as the computer is concerned, the swiper is not much different than a keyboard; you just get a bunch of data in one scan instead of having to type it out manually. The scanned data is sent to the active program running on your machine. This can be notepad (which is great for testing) or the active form field on a Web browser. I used the latter to integrate into their Web based shopping cart.

The data contained on a credit card consists of the card number, some bank routing information, the customer's name, and other miscellaneous information. More details on the specifics of the credit card swipe data are at http://money.howstuffworks.com/credit-card1.htm. To process the credit card data you don't need to know the specifics of what it is. The PayFlow Pro Web API has a field for sending swipe data, so it's easy to send it off the transaction and receive information back just as you would a normal Web transaction. I would expect that most payment processors have a similar option.

## Thursday: Processing Bar Codes and Credit Card Swipes

Now that the bar codes were generated, the client wanted to be able to process scanned bar codes in the Web browser. The CueCat, like other bar code readers, operates in the same manner that the credit card scanner does. When you swipe a card or scan a bar code, the scanned (or swiped) information goes to the active screen. If you have a Web browser open, it will go to


ColdFusion Report Builder Label Creation Wizard

Define Query Fields

Please define the query fields for your label. You will design the layout of the label in the next panel.

You can also edit these fields later from the Fields and Parameters tool window.

Non-printed Fields

Printed Fields

FirstName
LastName
Address
City
State
Zip

Figure 4: Enter your query Fields

the selected form field. To start the input page, first we'll need the form:

```
<form action="BarCodeSCanner.cfm" method="post"
         name="BarCodeScanner" id="BarCodeScanner">
 Scanned Info: <input type="text" name="ScannedInfo">
</form>
```

This is a form that will submit back onto itself. The form has one input field, with no submit button. For attendance purposes in the scheduling system, the scan had to be as painless as possible without any user interaction, so the form is submitted through JavaScript.

How does the browser know we've scanned something? How do we know when to submit? The answer is through the use of a JavaScript timer. The timer is a JavaScript that counts down from some number to 0. When it reaches 0, it checks whether there is data in the ScannedInfo field (or not) and then submits the form if there is.

I used the JavaScript code located at http://www.mcfedries.com/JavaScript/timer.asp as a base for this. (Much thanks to Paul McFedries' for his assistance). For space reasons, I'll refrain from copying the full script here. There are four variables to take into account:

- *Secs:* The secs variable specifies how many clock ticks until the timer reaches 0. This will count down from the initial value to 0.
- *Delay:* This is the amount of milliseconds before the timer checks its next status. 1000 milliseconds are equal to one second, so that is the default.
- *TimerID*: The timerID is a variable internal to the script. It specifies when the next clock tick will occur.
- *TimerRunning:* The timerRunning variable is a Boolean value that tells us whether the timer is running or not. For all intents and purposes, it is internal to the script and you don't need to worry about it.

There are three functions that make up the timer:

- *InitializeTimer:* The initialize timer function sets the secs variable, stops the timer (if currently running), and then starts the timer.
- *StopTheClock:* The stop the clock function runs some script to prevent the clock from running.
- *StartTheTimer:* The StartTheTimer function is the meat of the execution. It counts down from the clock tick and executes certain code if the count down is done, or moves onto the next clock tick. Of these three functions, it is the StartTheTimer function that needs a more in depth examination.

```
function StartTheTimer(){
 if (secs==0){
  StopTheClock()
  if (document.BarCodeScanner.ScannedInfo.value != ''){
   document.BarCodeScanner.submit();
   } else {
      InitializeTimer();
    }

  }else {
   secs = secs - 1
   timerRunning = true
     timerID = self.setTimeout("StartTheTimer()", delay)
  }
}
```

The StartTheTimer function starts by checking to see if the secs variable has counted down to zero yet. If it has, first it stops the clock. Then it checks to see if the form field has data in it. If the form field has data, submit the form. Otherwise, restart the clock.

If the time hasn't counted down to zero yet, subtract one from the counter, and set the function to execute again in one second. The function is recursive, calling itself. Load the page, and the timer waits, checking to see if something had been entered. Once it has, it submits the form, which does some further processing.

To start the ball rolling, you need to execute the InitializeTimer function, like this:

```
InitializeTimer();
```

As long as that is somewhere on the page, you're all set. This example is a bit more simplified than what actually went into production (obviously since we didn't create a processing page). I did some Ajax-y stuff with iFrames to show a running status of what attendance was entered and when. Errors were also shown in the status list without interrupting future scans. It worked out well.

## Friday: Graphing Types

I was sitting with a client discussing a report. He wanted to view the report information in a graphical format, so we had decided to implement this report using cfchart. I made the mistake of asking if he would prefer a bar chart or a pie chart. The client, being like normal clients, asked if I could do both. It turns out I could.

There are three tags that are used to generate charts in ColdFusion: cfchart, cfchartseries, and cfchartdata. Cfchart defines overall attributes for graph, such as the format (PNG, JPG, or Flash), height, width, and other display attributes that affect the full chart. Full information is at the livedocs at http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000226.htm#2619630. cfchartseries is used to define the chart style, such as bar or pie. There are eleven different types of charts supported. Full documentation on the tag is at http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000228.htm#2741830. You can have multiple chart series inside a chart, but this tag must be nested inside a cfchart. cfchartdata is used to define a single point on the graph, and it is the simplest of the three tags, as it only defines the name of the point and its value. Cfchartdata must be located inside a cfchartseries. Its full documentation is located here: http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000228.htm#2741830.

I didn't want to have to build multiple versions of the same report, one with a bar chart and one with a pie chart. I decided to pass the chart type as a URL variable into the template, and use that to decide what type of chart to display. This gives me the option of easily adding any of the 11 types supported by cfchart when the client asks for more in the future.

First I cfparamed a charttype variable to default it:

```
<cfparam name="chartType" default="pie">
```

That should cover me in the case of nothing being passed in. The next step was to get the data. I decided to use a query. In this example, I'll pull from my MyFriend's RSS Aggregator database and display the number of items for each RSS Feed:

```
<cfquery name="getTotals" datasource="#request.dsn#">
 select RSSFeeds.title, count(Items.ItemID) as total
 from RSSFeeds join items on (RSSFeeds.RSSFeedID = Items.RSSFeedID)
 group by RSSFeeds.title
</cfquery>
```

And finally the page can display the chart like this:

```
<cfchart format="flash">
 <cfchartseries itemcolumn="title" valuecolumn="total"
                          query="getTotals" type="#chartType#" />
</cfchart>
```

Since the chart is getting data from a query, the cfchartdata tag is not needed. The cfchart, simply enough, tells us that we're creating a flash chart. I can load the page using URLs like this:

```
ChartTest.cfm?chartType=pie
ChartTest.cfm?chartType=bar
ChartTest.cfm?chartType=line
```

( and so on)

It was a simple, elegant solution that gave me another happy client.

## Conclusion

It was a good week. I love dealing with ColdFusion as a language because of the way it makes complicated development tasks seem simple. I feel confident knowing that it will help me address the next "crazy" idea. What are your favorite "odd" requests you've had to deal with? How did you solve them? Just fill out the contact form on my blog (www.jeffryhouser.com) and let me know!

### About the Author

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

*jeff@instantcoldfusion.com*

# Use Flash Forms and Flex To Give Applications New Life

## A webmaster's view

By Michael Markowski

I'm a Webmaster for the Air Protection Division (APD), EPA Region 3 in Philadelphia and in April 2006, I wrote an article for *CFDJ* entitled "How ColdFusion MX 7 Made Me a Hero at the Office" (Volume 8, Issue 4). That article described how I harnessed the power of ColdFusion to improve access to our most vital business information.

Now, I'm back to fill you in on how I've been able to use Flash Forms and Flex to revitalize our applications and make them even more potent than before.

Since writing my previous article I've been busy reading up on and learning everything I can about Flex 2, MXML, Actionscript, and Flash Forms. I've also been hard at work incorporating these new technologies into my applications. I say "new" because although they've been around for some time, I haven't actually used these technologies in my own applications until

now. First, I'll describe how I used a Flash Form to create an application that unified and enhanced our salient issues ColdFusion applications. Then I'll explain how I used ColdFusion and Flex to create a "kicked up" version of our permits Web page (apologies to Emeril). Last and most importantly, I hope to inspire you to experiment with Flash Forms and Flex for yourself, as well as give you ideas on how to breathe new life into your applications.

In my previous article I described how ColdFusion MX 7 enabled me to create the Air Salient Issues ColdFusion Application (ASICA) and three similar applications that together provide our employees with access to over 5,000 salient issue records on our intranet. Salient issues are short (one-page) Microsoft Word documents that describe our most significant activities, accomplishments, issues, and events. Typically, a total of 30 to 40 salient issues are written each week. Currently, there are four salient issues applications on our intranet, one for each of the Region 3 Division offices APD, EAID, WCMD and HSCD. We also have four Microsoft Access databases, one for each salient issues application. In case you're wondering why we're using Microsoft Access instead of a "real" database management system as the back-end for these applications, I have just two words for you: limited budget. The good news, however, is that we haven't encountered any serious problems using Access; in fact, so far it has exceeded our every expectation. Template execution times in the ASICA rarely exceed 250ms even with the large number of database records that we have.

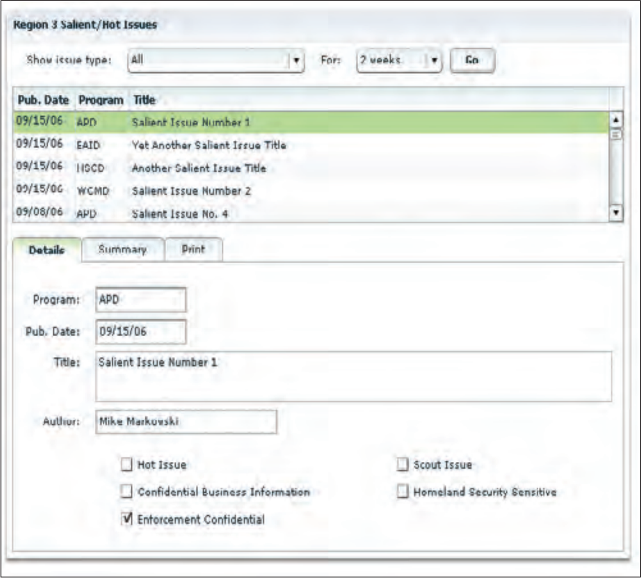
## Ready, Set, Flash

During the past year I developed (pun intended) a keen interest in both Flash Forms and Flex because I recognized that these technologies could, if used properly, greatly enhance the user experience while enabling developers to write less code. So, I began reading everything I could find about Flash Forms and Flex on the Web, in books, and of course, in CFDJ. That's when I saw Laura Arguello and Nahuel Foronda's "Completing the Real Estate Sample Application" article in the February 2006 issue of CFDJ (Volume 8 Issue 2). Using this article as a guide, I began building a new kind of salient issues ColdFusion application, one that would integrate and unify our salient issues databases and give users a rich, interactive Flash interface. To facilitate development of this new application and, frankly, keep the learning curve manageable, I decided to concentrate solely on the data presentation and leave the administrative (data entry) templates alone, at least for now. The first problem I faced was the large number of records and the fact that they were stored in four separate databases. I considered copying the records into a single large database but that would be too troublesome plus it might create other problems such as synchronizing data between multiple databases. I'd also read that Flash Forms were not the best solution for applications that must display thousands of records. Somehow I had to drastically reduce the number of salient issue records so that a Flash Form data presentation would be feasible while working with the four databases that I already had.

I resolved this dilemma by deciding that the new application would retrieve, process, and display only the most recent records from each of the four databases. In other words, this new application would retrieve and display only those salient issues that were published over the last month. This approach reduced the number of records the application would have to display from over 5,000 to about 150. Having reduced the record count to a reasonable number, the application was now a candidate for a Flash Form interface. And, of course, users looking for older salient issues that were published more than a month before could still access them by using one of the original salient issues applications.

The toughest challenge in writing this application was getting the most recent salient issue records out of the four databases and into a single query result set that I could then display in a Flash Form control. The first step was to query each salient issues database separately and retrieve only those salient issues that were published in the last month. The query to accomplish this is shown in Listing 1.

The query in Listing 1 is run four times, once for each salient issues database. The Now(), DateAdd, and CreateODBCDate functions are nested to create an "SQL-friendly" comparison date (i.e., it can be safely used in an SQL statement) one month in the past that can be compared to the date values in the PubDate column. The "WHERE" clause then retrieves only those records having a publication date later than (or equal to) the comparison date. In other words, the WHERE clause eliminates the vast majority of records that have publication dates more than a month old. This approach eliminates unnecessary records early on so that it reduces the processing burden for later queries. When executing multiple queries it's a good idea to run your most restrictive queries first. This minimizes the number of records that must be processed by later queries and boosts application performance. The record set generated by Listing 1 will have eight columns. The first seven columns come directly from the salient issues database, but the eighth column named "IssueType" is created "on-the-fly" by using an alias in the SQL statement; in Listing 1 the IssueType column is given a constant value of "APD." The other database queries, which are similar to Listing 1, will each have a different (but still constant) value for the IssueType column (i.e., "WCMD," "EAID," or "HSCD") depending on which database is being queried. I'll explain why the IssueType column is so important in a moment.


Region 3 Salient/Hot Issues

Show issue type: All For: 2 weeks Go

| Pub. Date | Program | Title |
|---|---|---|
| 09/15/06 | APD | Salient Issue Number 1 |
| 09/15/06 | EAID | Yet Another Salient Issue Title |
| 09/15/06 | HSCD | Another Salient Issue Title |
| 09/15/06 | WCMD | Salient Issue Number 2 |
| 09/08/06 | APD | Salient Issue No. 4 |

Program: APD
Pub. Date: 09/15/06
Title: Salient Issue Number 1
Author: Mike Markowski

☐ Hot Issue ☐ Scout Issue
☐ Confidential Business Information ☐ Homeland Security Sensitive
☑ Enforcement Confidential

Figure 3

## Come Together Right Now

The next step was to figure out how to combine all four database queries into a single query result set that I could then display in a Flash Form control. This is where ColdFusion's query of queries (QoQ) feature was invaluable. For those unfamiliar with it, QoQ lets developers "re-query" queries that have already been run. You can even use QoQ to combine queries from different databases or re-query a query that was itself created with a QoQ. As shown in Listing 2, you set the DBTYPE attribute in the <CFQUERY> tag to "Query," remove the "datasource" attribute, and use the query name as a table in your SQL.

In Listing 2 the "UNION" operator is used to combine the two database queries "getRecentIssuesAPD" and "getRecentIssuesEAID" into a single result set named "getRecentIssuesR3." It's possible to combine more than two queries using two or more UNION operators, but for clarity's sake only two are shown in Listing 2. All four salient issues databases are identical except for the data, i.e., the databases are structured the same with exactly the same tables and columns, so there was no problem combining query results this way. The IssueType column, which was created using an alias in the four database queries (see Listing 1), identifies the program that each salient issue record is associated with: APD, EAID, HSCD, or WCMD. Since the query in Listing 2 mixes all the records together, without the IssueType column there would be no way of knowing which salient issue record was associated with which program. The IssueType column lets us label each record with the name of the program it belongs to.

The desired end product of these five queries is a single result set named "getRecentIssuesR3" that holds the recent salient issue records from all four salient issues databases. The ORDER BY clause puts the records in descending order by publication date and in ascending order by IssueType. This puts the most recent records at the top of the display where they belong and records having the same publication date are then ordered by IssueType.

The QoQ in Listing 2 obviously relies heavily on the SQL "UNION" operator to create the final (combined) result set. UNION tends to get a lot of negative attention because it can hinder database performance. In this case, however, the performance impact from using UNION was negligible. The average template execution time was consistently below 300ms, possibly due to the small number of records retrieved. However, just to be on the safe side, I implemented query results caching in all four database queries, which improved performance even more. The fact that our server is running CFMX 7 Enterprise Edition on a dual-Pentium 4 box with 2GB of memory doesn't hurt either. If you've been doing Web development for very long you know that sooner or later we all must make compromises to get our projects finished on time, on budget, and with the expected features in place. I have no doubt that using UNION was the best approach in this situation, but every situation (and application) is different so you should always weigh your options carefully, test thoroughly, and then make the decision that's right for your particular circumstance.

## The Power of the Grid

With the queries and other business logic out of the way, I

# BALANCE

Designer/developer, front-end/back-end, clients/sanity. . .web development is a balance and we can help you maintain it. Join now and experience a wealth of training resources tailored to the tools you use every day.

**www.communitymx.com**

Visit www.communitymx.com/trial/ for your free 10 day trial.

COMMUNITY MX

could now turn my attention to the "fun" part of this project, the (Flash) data presentation. While studying the Real Estate Sample Application article, I was pleased to learn that I could use a Flash Form datagrid to "store" query results and display a master-detail interface on the same page. Using the "bind" attribute of the <CFINPUT> and <CFTEXTAREA> form controls, I could easily display the details of each record selected by the user in the datagrid. Best of all, there would be no need to re-query the database each time a different record was selected in the datagrid because the entire query is stored in the datagrid. "This changes everything," I thought to myself once I'd grasped the significance of data binding in Flash Forms. For me, at least, Flash Forms were no longer something I could ignore in my ColdFusion development.

As it turns out, it was very easy to display ColdFusion query data in a datagrid. This simplicity is achieved largely through the datagrid's "query" attribute in which you specify the name of the ColdFusion query you want to display in the grid, e.g., <CFGRID QUERY="getRecentIssuesR3" NAME="myGrid">. Implementing basic datagrid functionality is really that simple. Next, I used the <CFGRIDCOLUMN> tag to specify three columns to be displayed in the datagrid: Publication Date, Program (i.e., IssueType), and Title. These are the only columns that users would actually see in the grid. However, all of the query data is stored in the grid and, therefore, is available for display elsewhere in the Flash Form through the use of data binding. For more detailed information on how to bind Flash Form controls to a datagrid, see the "Completing the Real Estate Sample Application" article.

I must say that using a datagrid greatly simplified and facilitated my development efforts. First, the datagrid made it possible to display both a "master" list of records and a "record details" section on the same page. When the user selects an item in the datagrid, the details of the selected record appear in the various Flash Form controls on the page. This eliminates the need for separate "master" and "details" templates in the application. The second benefit of using a datagrid is that the amount of code that I had to write was substantially reduced. Numerous HTML and CFML tags were replaced with just three tags: <CFGRID>, <CFGRIDCOLUMN>, and <CFINPUT>. All right, it's really four tags if you count the <CFTEXTAREA> tag. Another benefit of using a datagrid is fewer calls to the database and better performance since the master-detail interface in the Flash Form uses the query data stored in the datagrid, not the database, to populate the form controls.

I soon realized, however, that I had another problem. The "Title" and "Summary" text areas contained large amounts of text and consumed too much vertical space in the form. This made the page too big to see without vertical scrolling. Once again it was Flash Forms to the rescue, this time via the Tab Navigator layout container. The code in Listing 3 shows how I used a Tab Navigator container to distribute the text areas (and other content) among three distinct "tabbed" sections in the form. The content in any of these tabs could be accessed simply by selecting the appropriate tab.

Now, with the Tab Navigator container dividing the content evenly among three tabbed sections in the form, there was plenty of room for all of the form controls, including the two large text areas.

Once I had figured out how to bind the text areas and text input controls to the datagrid, the remaining development work went quickly. I had to consult the Real Estate Sample Application article once more to figure out how to bind checkboxes to the datagrid, but that wasn't too difficult. By now the application was almost ready and I began testing it to ensure cross-browser compatibility. One of the greatest benefits of Flash is its ability to render consistently across a variety of different platforms and browsers. This alone is sufficient reason to consider using Flash Forms for your next ColdFusion project.

Next, I added two <CFSELECT> list menus above the datagrid to let users filter records in the datagrid. The first of these menus filters salient issues according to their confidential status, e.g., Enforcement Confidential, Business Confidential, etc. The other menu allows selection of salient issues that are one, two, three or four weeks old for display in the datagrid. By setting up the page so that the Flash Form submits to itself, I was able to keep it all to a single page. However, in the interest of giving users a "print version" option, I created two "print version" templates that use several URL variables (passed from the main page) to display detailed information on any salient issue in FlashPaper, Acrobat, or HTML format. In the third tab of the tabbed section in the main page (see Listing 3) I employed three <CFFORMITEM> tags with the Type="html" attribute to create three dynamic HTML links. Thanks to the <CFFORMITEM> "bind" attribute, the URL parameters in these links change according to the record that's selected in the datagrid. These links pass the necessary parameters to the "print version" templates, allowing the latter to query the correct salient issues database and retrieve data on the desired record. The code for this is shown in Listing 4

Any salient issue displayed in the datagrid could now be printed in HTML, Acrobat, or FlashPaper format by clicking on the appropriate link in the third tab of the Tab Navigator.

Figure 1 shows the completed Flash UI for the application. I added some fictitious records so that it would look similar to the actual application. By using a Flash Form to handle the data presentation, I had more time to focus on the database queries and other application logic. Frankly, I was amazed that such a sophisticated interface could be implemented so easily and with so little code. But then I remembered that this was ColdFusion, and we all know how ColdFusion excels at making even the most complex tasks seem trivial. Indeed, that's what I like most about ColdFusion, its unique ability to make quick work of complex programming chores so that developers can get real work done. Now, whenever I start a new ColdFusion project I ask myself, "Will Flash Forms work here?" I strongly encourage you to do the same because Flash Forms can save you precious time, simplify your code, and deliver a compelling user experience.

## Beyond Flash Forms: Flex 2

As you may have guessed, over the past few months I've been on a mission to "Flash-enable" my applications, and I didn't want that mission to end with Flash Forms. So I began experimenting with Flex 2. My experience with Flash forms made

Adobe

## Beyond boundaries.

In the right environment, imagination has no limits.

MAX 2006, the annual Adobe user conference, offers a unique opportunity to learn about Adobe software, interact with industry experts, connect with the Adobe community, and explore ideas yet to be imagined.

Choose from over 100 different workshops and hands-on sessions presented by Adobe experts and industry leaders. Exchange ideas with other community members at a variety of networking events. Experience current and emerging Adobe technology.

Join us for MAX 2006 this **October 23-26** in Las Vegas to learn, network, and move beyond the boundaries of what you believe is possible.

**Online registration ends Oct 16.**
**Register now: www.adobe.com/go/cfdj**

MAX

Beyond boundaries. The 2006 Adobe conference.

working with Flex much easier. Flash Forms and Flex are quite similar and have much in common such as the controls (e.g., datagrid), data binding, Actionscript, compiled .SWF file output, etc. Therefore, many of the things I learned while using Flash Forms could be applied to my Flex development. Once again I decided to start slow and tackle a relatively simple project. The project I chose was to convert one of our existing ColdFusion-generated permit pages (http://www.epa.gov/reg3artd/permitting/petitions2.htm) to a Flex page. I say "ColdFusion-generated" because this page really is generated automatically, on a weekly basis, by our ColdFusion server via the <CFSCHEDULE> tag, but that's a story for another article.

Flex applications can't interact directly with a database as ColdFusion does, so the data for Flex applications must come from data providers, i.e., services that Flex recognizes and can interact with. Data providers include arrays, Web Services, HTTP services and external XML files to name a few. That last option interested me the most. If I could somehow use a separate XML file as the data provider for my Flex page then I could update the application simply by replacing the XML file. Plus, there would be no need to concern myself at all with the presentation code when performing data refreshes. The idea of creating a Flex application with separate data and presentation code was very compelling, and so I fired up Flex Builder determined to create a Flex page that gets its data from a separate XML file.

I had previously studied the tips and resources available in the Adobe Flex 2 Developer Center, and the Flex Builder IDE made it easy to construct my Flex application. Working primarily in Source mode, I added panel, datagrid, form, textinput, and other components and used data binding and Actionscript to bind everything together, similar to what I had done when working with CFML and Flash forms.

After some trial and error writing MXML (Flex's XML-based markup language) and Actionscript code, my Flex application was finished. I have since put it on my division's public Web site at http://www.epa.gov/reg3artd/permitting/flex/t5permits2.htm. The .SWF file, which is the application's Flash UI, is invoked by an HTML "wrapper" page that will prompt the user to download and install Flash Player 9, if needed. A basic wrapper page is automatically created by Flex Builder when an application is compiled, but I modified it to look like a standard EPA Web page so it would be consistent with the rest of our site. I think that creating a wrapper page is a good idea because some users may not have the latest version of Flash Player installed and Flex applications require Flash Player 9.

I'm happy to report that my Flex page uses a separate local XML file named "permits.xml" (http://www.epa.gov/reg3artd/permitting/flex/permits.xml) as its data provider. So it's possible for even a Flex novice like me to create Flex applications that use data from an "external" source. I even wrote a special ColdFusion template that queries our internal Title V Operating Permits (Oracle) database and constructs the permits.xml file from the query results. When I want to refresh the Flex application with new data, I simply run this ColdFusion template and replace the old XML file with the new one generated by the template. I think that in the future we'll see a lot of database information being converted into XML since Flex can't interact directly with a database but can easily access and retrieve data "packaged" in XML.

Separating presentation and data code certainly has its advantages, but for me the best thing about Flex is that no ColdFusion or other server-side support is needed for basic standalone Flex applications (like mine). Basic Flex applications can be deployed just like HTML files and will work on a standard (non-ColdFusion) Web server. This is important in my situation because, as I've mentioned before, the cost of ColdFusion support on EPA's public Web server is high. Therefore, my Division's public Web site doesn't have ColdFusion support at this time. However, with Flex I can still present our data via a Flash interface.

I've already begun thinking of ways to enhance my Flex page with images, links, or even charts. But I'm out of space for now and this will have to wait for another article. Thanks for reading and I hope that I've given you some interesting ideas on how to use Flash Forms and Flex to revitalize your own applications.

## About the Author

*Michael Markowski works for the Air Protection Division at the Environmental Protection Agency and is a Macromedia/Adobe Certified Professional.*

*markowski.mike@epa.gov*

### Listing 1

```
<!--- Retrieve the most recent records from each salient issues data-
base --->
<CFQUERY NAME="getRecentIssuesAPD" DATASOURCE="APDSALIENTS">
SELECT IssueID, PubDate, Title, AuthFirst, AuthLast, ConfStatus, Sum-
mary, 'APD' AS IssueType
FROM Issues INNER JOIN Authors ON (Issues.AuthorID = Authors.AuthorID)
WHERE PubDate >= #CreateODBCDate(DateAdd("M", -1, Now()))#
--- SQL abbreviated for clarity ---
</CFQUERY>
```

### Listing 2

```
<!---  QoQ to combine records from multiple databases into a single
result set --->
<CFQUERY DBTYPE="query" NAME="getRecentIssuesR3">
SELECT *
FROM getRecentIssuesAPD
UNION
SELECT *
FROM getRecentIssuesEAID
--- SQL abbreviated for clarity ---
ORDER BY PubDate DESC, IssueType
</CFQUERY>
```

### Listing 3

```
<!--- Use Tab Navigator container to split up content among 3 tabs --->
<CFFORMGROUP type="tabnavigator">

<!--- 1st Section of Tab Navigator --->
<CFFORMGROUP type="page" label="Details">
```

```
Display Title, IssueType, PubDate, etc. here
</CFFORMGROUP>

<!--- 2nd Section of Tab Navigator --->
<CFFORMGROUP type="page" label="Summary">
Display the large Summary text area here
</CFFORMGROUP>

<!--- 3rd Section of Tab Navigator --->
<CFFORMGROUP type="page" label="Print">
Display the three "Print option" links here
</CFFORMGROUP>

</CFFORMGROUP>
```

### Listing 4

```
<CFFORMITEM type="html" bind="<a target='_blank' href='printsalient.
cfm?docType=flashpaper&prog={salientsGrid.selectedItem.IssueType}&Issu
eID={salientsGrid.selectedItem.IssueID}'>Adobe Flashpaper</a>" />
```

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

## CFDJ Advertiser Index

| ADVERTISER | URL | PHONE | PAGE |
|---|---|---|---|
| CFDynamics | www.cfdynamics.com | 866-233-9629 | 2 |
| Adobe | www.adobe.com | | 3 |
| EdgeWebHosting | http://edgewebhosting.net | 866-334-3932 | 4 |
| CF_Underground | www.CF-Underground.com | | 6 |
| Vitalstream | www.vitalstream.com | 800-254-7554 | 10,11 |
| Intergral | www.fusion-reactor.com | | 13 |
| Interakt | www.interaktonline.com/mxcss | | 15 |
| CommunityMX | www.communitymx.com/trial | | 25 |
| Adobe MAX | www.adobe.com/go/cfdj | | 27 |
| AJAXWorld East | www.AjaxWorldExpo.com | 201-802-3020 | 43 |
| Hot Banana | www.hotbanana.com | 866-296-1803 | 49 |
| Hal Helms | www.halhelms.com | | 49 |
| Straker Interactive | www.zoomflex.com | (64 9) 309-5252 | 51 |
| HostMySite | www.hostmysite.com | | 52 |

# ColdFusion U

**For more information go to...**

## U.S.

**Alabama**
Huntsville, AL CFUG
www.nacfug.com

**Arizona**
Phoenix, AZ CFUG
www.azcfug.com

**California**
Bay Area CFUG
www.bacfug.net

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

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

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

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

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

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

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

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

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

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

**Indiana**
Indianapolis, IN CFUG
www.hoosierfusion.com

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

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

**Maryland**
Maryland CFUG
www.cfug-md.org

**Massachusetts**
Boston CFUG
http://bostoncfug.org/

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

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

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

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

**Minnesota**
Twin Cities CFUG
www.colderfusion.com

**Missouri**
Kansas City, MO CFUG
www.kcfusion.org

**Nebraska**
Omaha, NE CFUG
www.necfug.com

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

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

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

**New York**
Albany, NY CFUG
www.anycfug.org

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

**New York**
Syracuse, NY CFUG
www.cfugcny.org

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

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

**Oregon**
Portland, OR CFUG
www.pdxcfug.org

**Pennsylvania**
Central Penn CFUG
www.centralpenncfug.org

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

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

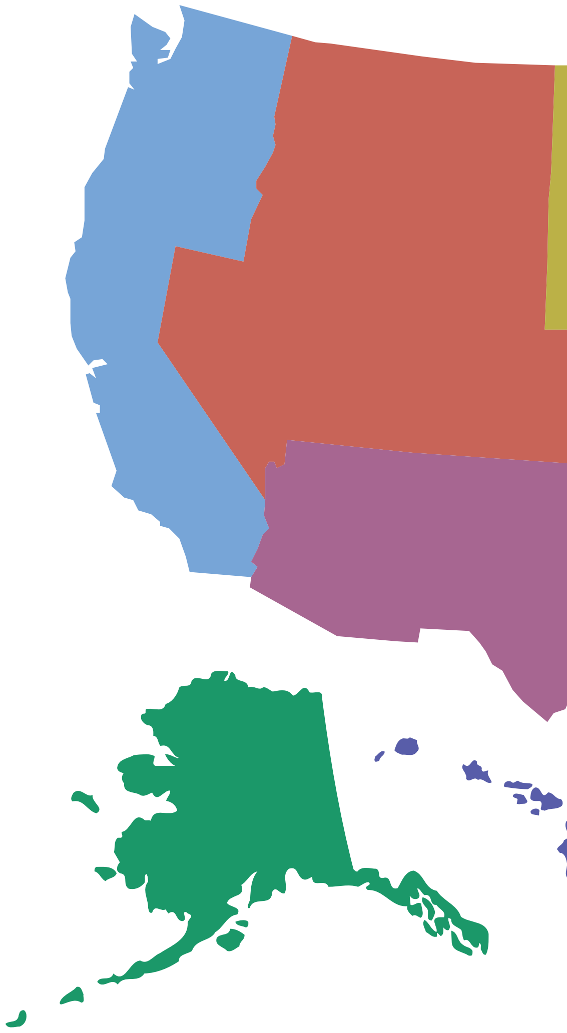
**Tennessee**
Nashville, TN CFUG
http://www.ncfug.com

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

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

**Texas**
Dallas, TX CFUG
www.dfwcfug.org/

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



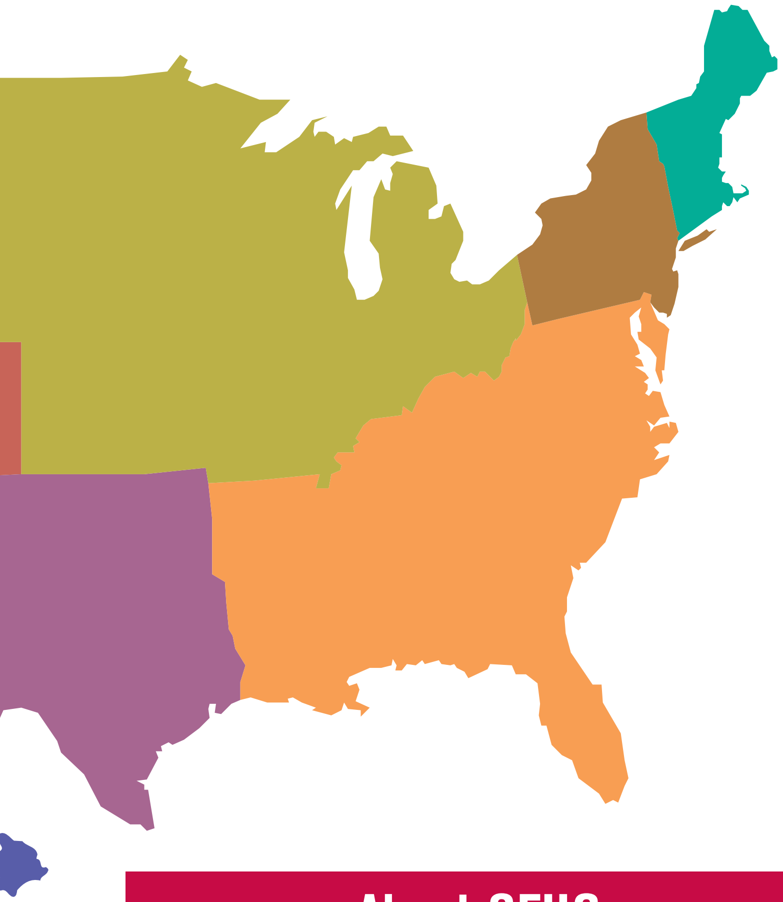
**Utah**
Salt Lake City, UT CFUG
www.slcfug.org

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

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

Jser Groups

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



## About CFUGs

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

## INTERNATIONAL

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

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

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

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

**Canada**
Kingston, ON CFUG
www.kcfug.org

**Canada**
Toronto, ON CFUG
www.cfugtoronto.org

**Germany**
Central Europe CFUG
www.cfug.de

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

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

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

**Scotland**
Scottish CFUG
www.scottishcfug.com

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

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

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

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

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

**Turkey**
Turkey CFUG
www.cftr.net

**United Kingdom**
UK CFUG
www.ukcfug.org

# Monitoring Your CF Environment with the Free

**Solve common challenges in ways no other monitoring feature in CF or even the dedicated monitoring tools can do**

By Charlie Arehart

There are many resources we should analyze to ensure optimal ColdFusion operation or to help diagnose problems. Fortunately, there's an awesome free tool that comes to our aid to turn voluminous data into useful information.

In this article, I'd like to introduce you to the free Log Parser tool from Microsoft. Yes, it's free. And while you may not run ColdFusion on Windows, that's okay. You can use it on a Windows machine to monitor resources on a Linux machine. The tool applies just as well to those running BlueDragon or any CFML, PHP, .NET, or other environment.

I'll show you the many ways you can use the tool to solve common challenges in ways that no other monitoring feature in CF or even the dedicated monitoring tools like SeeFusion or FusionReactor can do. In one example, I'll show you how it can provide application-specific error log information that many struggle to obtain.

## Basics of the Tool

Despite its name the tool is about more than just log files. What kind of resources can Log Parser monitor? It started out focused on Web server log files, and indeed it can do that (in IIS, W3C, and NCSA formats), but it can also analyze all manner of tabular text files (CSV and TSV), as well as XML files, the Windows Registry, the file system, the IIS metabase, the Windows event logs, the Windows Active Directory, and NetMon files.

And while it can produce simple textual results, it can also generate output to a file in plain text, CSV, TSV, or XML formats, as well as produce charts, and store results in a database

But the best, most compelling, and truly unique aspect of the tool is how you go about analyzing the input files. There's no interface for the tool. So how do you describe what data to retrieve? Would you believe you use SQL? That makes it an especially compelling tool for CFML developers, since we're used to using SQL already.

I'd like to focus here on particular forms of information that would be useful for CF developers and administrators to analyze using the tool, so I won't focus on its installation or basic use. I'll direct you to other resources at the end of this article that will get you started.

I'd just like to clarify that Log Parser is meant to be used at the command line (logparser.exe). I'll assume that you have it installed and configured so you can issue commands such as its help option:

```
Logpart -h
```

Of course, you can also use such a tool from the CFEXECUTE tag, but again that's beyond the focus of this article. I'll point you to a resource later that will cover such additional details.

# Log Parser Toolkit

## Analyzing CF Log Files

It may seem odd at first to contemplate using SQL to analyze log files and the other non-database resources mentioned above, but it really is effective. While most of the existing resources that introduce the tool focus on analyzing Web server logs, I'd like to start by showing an analysis of ColdFusion logs.

Now, how could a tool that analyzes log files regard the columns in that file as columns to be used in SQL? Well, many log files do have a first line called a "header" line that provides a list of names that identify each column in the log file.

Even if the log file doesn't offer a header file, the Log Parser tool has a clever mechanism to analyze the first 10 lines of the file to try to determine what kind of data is in each column and create generic column names.

For this article, let's consider the application.log file that tracks errors in your CFML code and is stored in the "logs" directory where CFMX is installed. On my machine, that's c:\cfusionmx\logs. To query all the columns in all the records of that file, I could issue:

```
logparser "select * from C:\CFusionMX\logs\application.log" -i:csv
```

Note the familiar "select *" SQL statement that says "select all columns" and the "from" clause that names the actual log file to be processed, all of which is embedded in double quotes. The additional "-i:csv" argument tells the logparser engine that the file is a CSV (comma-separated value) format file. A subset of the result as it might appear is:

```
     Filename                    RowNumber Severity    ThreadID Date
Time     Application Message
-------------------------------- --------- ---------- -------- -------- --
------ ----------- -------------
C:\CFusionMX\logs\application.log 2        Information jrpp-0
06/21/06 19:53:15 <NULL>     C:\CFusionMX\logs\app
lication.log initialized
C:\CFusionMX\logs\application.log 3        Error      jrpp-0
06/21/06 19:53:15 <NULL>     Error Executing Datab
ase Query.Data source not found. The specific sequence of files
included or processed is: C:\Inetpub\wwwroot\test.cf
m
```

Yes, it's a jumble for now (all dumping onto your screen at the command line), but later we'll see how to limit what columns to show as well as how to write the output to a file for more effective observation.

## Understanding Column Headers

The first line above shows all the column headers that were found in that header file. You can also have the tool list the columns that it's found or detected using the option -queryinfo, such as this:

```
logparser "select * from C:\CFusionMX\logs\application.log" -i:csv -
queryinfo
```

In the data shown will be the list of columns found. In this example, it would be:

```
Query fields:
Filename (S)   RowNumber (I)   Severity (S)      ThreadID (S)
Date (S)       Time (S)        Application (S)   Message (S)
```

Note that the "S" and "I" indicators mean the columns hold strings or integers, respectively. Other types are "T" (timestamp) and "R" (real). We can use any of these column names in the SELECT statement to limit what columns we display. More important, we can use them to limit what "records" we display.

## Limiting the Records Found

Going back to the earlier example that just did a "select *" the result was basically the same as if we'd just dumped the file to the screen. Where the tool gets powerful is in using additional SQL clauses to refine the search. For instance, since we see that one of the columns is "severity," which has values such as "Error" or "Information," we could limit the list to just those with errors using:

```
logparser "select * from C:\CFusionMX\logs\application.log where
severity='Error'" -i:csv
```

Note the addition of where severity='Error.' As with most SQL engines, the string value must be surrounded with single (not double) quotes. On the other hand, notice that the first character of the value is capitalized because, unlike most SQL engines, the comparison here is case-sensitive.

Of still more interest may be to limit the errors to a given application. Since that's another field, you can use this:

```
logparser "select * from C:\CFusionMX\logs\application.log where
```

```
severity='Error' and application='test'" -i:csv
```

This would list only those errors for the application "test."

## Grouping Records

Now, perhaps a different interest would be to see a breakdown of errors by application. Again, this is easy in SQL and therefore easy in Log Parser:

```
logparser "select application, count(*) as NumErrors from C:\CFusionMX\
logs\application.log where severity='Error' group by application" -i:csv
```

First note the use of the "group by application" clause, which changes the SQL to group all the records of the same application together. Then note the use in the SELECT clause of "COUNT(*) as NumErrors," which will count how many records there are in each such group (by application), reporting it as a NumErrors column. Finally, note the addition of application to the Select list to show the name of each application. The result of this might be:

```
Application              NumError
-------------------------- --------
<NULL>                    473
cfadmin                   9
App1                      1
test_admin                60
demoapps                  3
ContactManager            1
```

Of course, if we wanted the list to be sorted either by the application names or the number of errors, we could add an "Order by" to the SQL.

## Creating an Output File

The examples to this point have simply output the results to the screen (command line), but you may prefer to write the results to a file instead. You do this using an INTO clause, which is placed at the end of the SELECT (and before the FROM), to name a file, such as:

```
logparser "select * into test.txt from C:\CFusionMX\logs\application.log
where severity='Error'" -i:csv
```

Notice here that I've switched back to the earlier example of listing all the errors from the application.log file. Note as well that the INTO clause can specify a full or relative path (or a UNC path) for the file name, otherwise if none is provided, it will simply be stored in the current directory where the command takes place.

The file output format will default to what has been shown on screen to this point, which Log Parser calls a "NAT" format (native or tabular data format). But as mentioned at the outset, you can also cause the output (on screen or to a file) to be any of several other formats such as CSV, TSV, and XML. For instance, you could change the output file to an XML file:

```
logparser "select * into test.xml from C:\CFusionMX\logs\application.log
where severity='Error' " -i:csv o:xml
```

## Creating an Output File Per Application: Finally Error Files Per App!

A unique feature of the Log Parser tool helps resolve a problem that has long plagued CF administrators. If you have multiple applications and development teams on one server, you (or they) have probably wished you could create a separate error log file for each application. Well, Log Parser makes this a snap, using a feature called "multiplexing."

It's really as simple as using wildcard characters in the INTO clause, which then correspond to the first column listed in the SELECT clause. Continuing the example above, recall that it's listing all the columns for all the messages that reflect an error. We can alter that to name the application column first, and while we're at it we may as well limit it to just that column, the filename, message, and date and time, as in:

```
logparser "select application,filename,message,date,time into errors_*.txt
 from C:\CFusionMX\logs\application.log where severity='Error' " -i:csv
-o:csv
```

Using the data in the log file reflected so far, where there were errors for the applications such as "App1" and "demoapps," this would create a file such as errors_App1.txt, with the errors for App1, and errors_demoapps.txt, with the errors for demoapps. You'll notice that I've changed the output format to CSV. For some reason, the default NAT format isn't supported for multiplexing, though TSV, XML, and others are. See the product docs for more information. This is really a very powerful mechanism. Imagine that after creating each app-specific file, you could go further and send each file to the appropriate application owner.

In fact, you can go further still to create a directory for each application instead. The secret is that you may use the * anywhere in the path. Just as the command above will create new files even if they didn't exist, you could have the tool create new directories as well. Consider the following:

```
logparser "select application,filename,message,date,time into *\errors.txt
 from C:\CFusionMX\logs\application.log where severity='Error' " -i:csv
-o:csv
```

Note the use of "*\errors.txt" for the Into clause. This will create a new directory for each application and store the file as errors.txt. (You could certainly repeat the * again if you wanted the application name in the file name too.) Imagine how you could now give the owners of each application access to "their" directory. And again, the path you name for the output directory could have used a UNC path to put the output on some shared server.

(As an aside regarding multiplexing, note that the first column listed is used only for determining the file name and doesn't appear in the result. If you really want it in the result, simply list the column twice.)

## But How Do I Get Only the Latest Log Entries?

We're about out of space, but I'd like to demonstrate just one more really compelling feature. Regarding the last example, you may be thinking, "this all sounds kind of interesting, but it will stink as the logs grow and grow in size. I wish I could just get the latest log entries." That's a very logical concern. And like the old steak knife commercials on TV, the Log Parser tool makes me want to say, "but wait, there's more!"

The tool has a feature called checkpointing that's really as simple as naming a checkpoint file that will track the row number in the log file where processing stopped. It's really as simple as naming a checkpoint file. Consider the following:

```
logparser "select application,application,filename,message,date,tim
e into *\errors_*.txt from C:\CFusionMX\logs\application.log where
severity='Error' " -i:csv -o:csv
-icheckpoint:test.ipc
```

Note the specification of the "–icheckpoint" argument, naming a file, in this case "test.ipc." Again, the checkpoint file specification can include a path so it can be put anywhere, or it will default to the directory where the logparser command is executed. The feature tracks where log processing has stopped and where subsequent log analysis should start. It can be used whether you're outputting the results to screen or file.

There's one thing to think about regarding use of this feature with the multiplexing (or indeed any log file output) operation: it will overwrite any previously existing file. This operation may make most sense when you're running the operation periodically and immediately sending the logfile somewhere else. That way, the recipient will get a new file whenever there's new content of interest to them. Nifty!

## Other Log Parser Features: Too Many to Name

I could go on. Clearly, this is a powerful tool, and we've only scratched the surface. I've provided several resources to help you learn more and see more examples. Consider briefly the following additional features:

- Again beyond log files, the tool can analyze Web server logs, the metabase, the registry, the file system, windows event logs, the Active Directory, and so on
- Input files can not only be pointed to on local and UNC paths, but they can also be specified as URLs as well. So, for example, you can use this tool to read remote CSV and XML files (including RSS feeds)
- You can pipe the output of command-line tools to be processed as input to the tool
- You can store the report output in a database
- You can generate charts from the tool
- Besides simple tabular output to the screen, there is also an available Windows datagrid
- You can choose to store the SQL in a file and point to it from the command line, and even pass in parameters to such a file
- You can name more than one file in the FROM (filename,filename), and the tool will automatically connect the files together (a "union" in SQL terms)
- Since you can read in XML files, consider that this lets you perform SQL against the XML, which CFML can't currently do
- The SQL grammar available is quite substantial. I've only used the simplest examples
- It also offers many useful SQL functions, including ones unique to the tool, like PropCount and PropSum
- The –h command-line option also offers help on SQL grammar and shows examples as well
- And this is just some of what it does

Also, I can think of additional CF-specific kinds of analysis, of various input sources, that I'd like to elaborate. For instance, I've only showed working with the application.log file, and only the logs in the \cfusionmx\logs directory. Note that there are many other log files in ColdFusion that could be useful to analyze. (You can even use Log Parser's ability to query the file system to report what log files exist.) Of course, it's often useful to analyze Web server logs (whether IIS, Apache, or the built-in Web server in CFMX) to help identify performance problems.

Beyond that, there's data in the event logs (related to running CF services), as well as possibly data in the registry (such as client variables, perhaps existing unexpectedly). And since the tool can read XML files generated and available via a URL, it could be used to monitor and report on the output of both SeeFusion and FusionReactor, which can output XML files. I'll offer a URL at the end to name a place where I plan to elaborate on such opportunities as an extension to this article.

## Resources for Learning More

Clearly, there's much more to learn. Perhaps the most important resource to start with is how to download the tool. There are a couple of links to note. The Microsoft link is www.microsoft.com/technet/scriptcenter/tools/logparser/default.mspx. The tool includes a help file with considerable extra detail. Beyond that, there's also an "unofficial" Web site at www.logparser.com/. It includes a knowledge base, forums, and links to articles.

But the best resource of all, to really learn and appreciate the tool, is the book Microsoft Log Parser Toolkit, available from Syngress at www.syngress.com/catalog/?pid=3110. Since it's a couple of years old, the book is also available used at Amazon. Even so, it's an awesome resource. You'll learn things you never dreamed were possible.You may enjoy the following articles:

- "How Log Parser 2.2 Works"
  - www.microsoft.com/technet/community/columns/profwin/pw0505.mspx
- "Analyze Web Stats with Log Parser"
  - www.microsoft.com/technet/technetmag/issues/2006/08/InsideMSFT/
- "What's New in Log Parser 2.2"
  - www.microsoft.com/technet/scriptcenter/tools/logparser/lpfeatures.mspx

Finally, as I alluded to above, I plan to create an area on my Web site to expand on this discussion and elaborate more CF-specific aspects of using Log Parser. Look for that at http://www.carehart.org/logparser/. I welcome your feedback.

### About the Author

*Charlie Arehart is a longtime contributor to the community and was recently selected to the Adobe Community Expert program. A certified Advanced CF Developer and Instructor for CF 4/5/MX, he frequently speaks at developer conferences and user groups. Formerly CTO of New Atlanta (BlueDragon), he is now an independent contractor.*

*charlie@carehart.org*

# Make Your Flash Forms More FLEXible

**Like what you can do with Flash Forms? Wish you could do more?**

By Ian Bale

Want to make your Flash forms more FLEXible? Well, now you can! But, is there any point, you say, now that Flex 2 is out and effectively free (if you can make do without FlexBuilder)? Well, if you can go with Flex 2 then do so, but maybe like me, it's currently off limits to you…

Since a bunch of the work I'm doing right now is for Sun Microsystems, it's vital that the application runs on Solaris, and for now there's no sign of FlashPlayer 9, which is required for Flex 2, being near to release. Same goes for Linux and just about every portable device out there.

Assuming that you're not taking the Flex 2 route, read on…

Why don't we start by whetting your appetite? Take a look at Figures 1–4 for some ideas of the additional features available that you can't access directly from Flash Forms.

To use these controls as well as pretty much anything else Flex (version 1.5) has to offer you really need to know about just one thing: the PopUpManager. In Flex it's used to open a new window dynamically on top of the calling application or window.

If you're not already familiar with Flex, I'd suggest you start by looking at Adobe's Flex 1.5 documentation at livedocs.macromedia.com/flex/15/.

Here's the ColdFusion code needed to load the Flex code:

```
<cfform format="Flash" height="600" width="800" onLoad="initApplication
()">
```

```
   <cfformitem type="script">
      function initApplication()
      {
         mx.managers.PopUpManager.createPopUp(this, Demo1, false);
      }
   </cfformitem>
</cfform>
```

As you can see, it's somewhat minimal: Simply a <CFFORM> tag that defines the height and width of my application, and an "onLoad" attribute that calls an embedded function (initApplication) that has a single line of code – the call to the PopUpManager. And that is all! ColdFusion simply provides a wrapper. Everything else is now done in Flex (MXML) code.

The createPopUp function has the following parameters:

- *parent*: Movie clip that will be the window's parent. Just use "this" to open as a child of the root flash form.
- *className:* Classname of the object that will be created. modal: If true, then the user can no longer access any of the application outside of our window until it's closed. Useful, but not needed for our main window.
- *initObj:* This lets us set parameters in our MXML file or pass data to it.
- *broadcastOutsideEvents:* Boolean value that determines if flash should generate mouseDownOutside events if we click outside of the window. Again, not needed for our examples.

My example opens a non-modal window as a child of the flash form. And the interesting part: the second parameter tells it to load an MXML file: Demo1.mxml. Yes, it loads an MXML file! This is what opens up the world of Flex to us since you can write pretty much anything in the native MXML and simply use it in this manner.

As you can see below, the MXML code for these demos is also very simple. (Note: The code below is simplified; see Listings 1-4 for the full code).( Listings 1-9 can be downloaded from the online version of this article at http://coldfusion.sys-con.com.)

```
Demo1.mxml (Figure 1)
<mx:Canvas>
   <mx:HSlider/>
   <mx:HSlider/>
   <mx:VSlider/>
</mx:Canvas>

Demo2.mxml (Figure 2)
<mx:Canvas>
   <mx:TitleWindow>
      <mx:FormItem>
         <mx:NumericStepper/>
      </mx:FormItem>
      <mx:ControlBar>
        <mx:Button/>
        <mx:Button/>
     </mx:ControlBar>
   </mx:TitleWindow>
</mx:Canvas>
```

```
Demo3.mxml (Figure 3)
<mx:Canvas>
   <mx:MediaPlayback/>
   <mx:MediaDisplay/>
   <mx:MediaController/>
</mx:Canvas>
```

Okay, so these demos are interesting, but if you're happy with the controls provided by ColdFusion Flash Forms then why bother? Well, if you've ever tried to write anything larger than a few forms, then chances are you've run into the dreaded 32k or 64k limit error. Build your applications this way and I doubt you'll see one of those again. Instead of cramming all the different forms and their associated code into a single form, thus rapidly reaching the size limitation, you can put each form into a separate MXML file, each of which has its own size limits. You can effectively build an application as large as you want.

I've built a large application for Sun using this technique. It loads faster than a normal Flash Form, presumably because it doesn't have to process the form tags to build the MXML. It consists of a main page with a tab navigator. Each tab has multiple

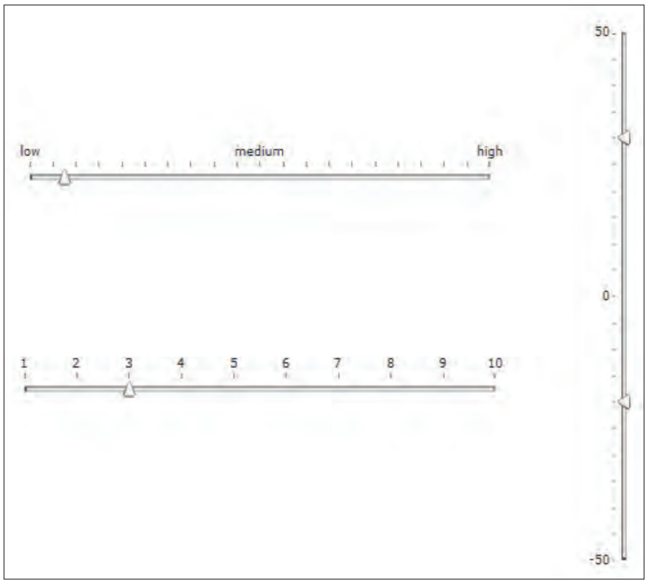


**Figure 1 Slider controls**

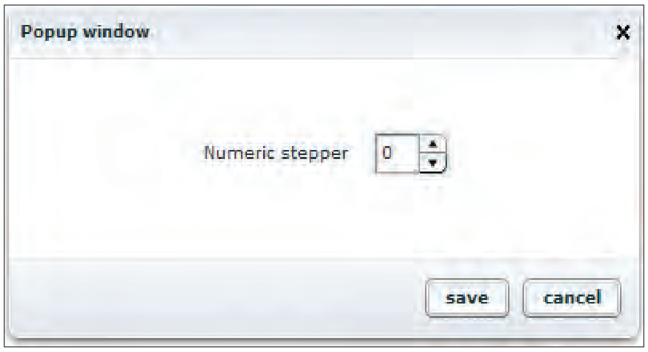


**Figure 2 Window with button control bar and a numeric stepper control**

form elements, and each opens a whole range of different pop-ups, many of which open further pop-ups. I made extensive use of the modal attribute of the createPopUp function since this means I don't have to worry about the user accessing anything but the topmost window.

If you have access to FlexBuilder then you have the added benefit of being able to build your forms visually, which is a welcome relief after fiddling for hours with CF tags to get the layouts just right.

A word of warning: take small steps! The compiler will warn you about some errors, but mostly you'll just get a blank screen if you have something wrong. Check the logs and you're unlikely to find anything more useful than a Java null pointer error.

You'll want to edit the flex-config.xml file on your development machine to give you as much debugging information as possible (typically located at C:\CFusionMX7\wwwroot\WEB-INF\cfform\flex-config.xml). Locate and change the values listed below, then restart the ColdFusion service.

```
show-all-warnings : true
show-binding-warnings : true
```

While you're editing this file, two other items of interest are:

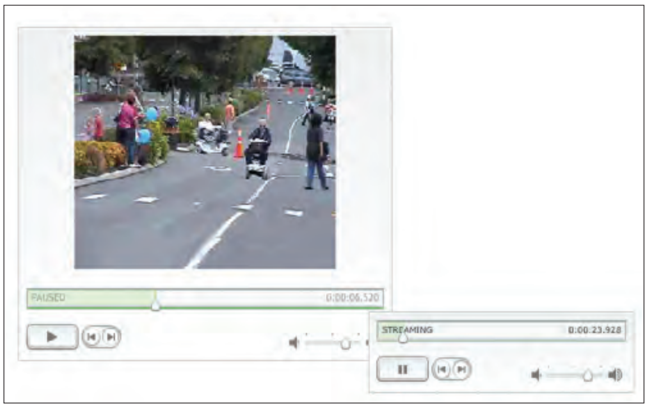


Figure 3 FLV video streamer and controls for MP3 streaming

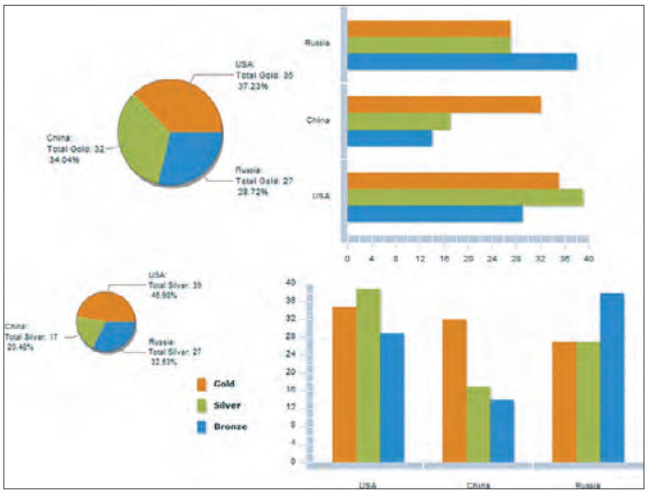
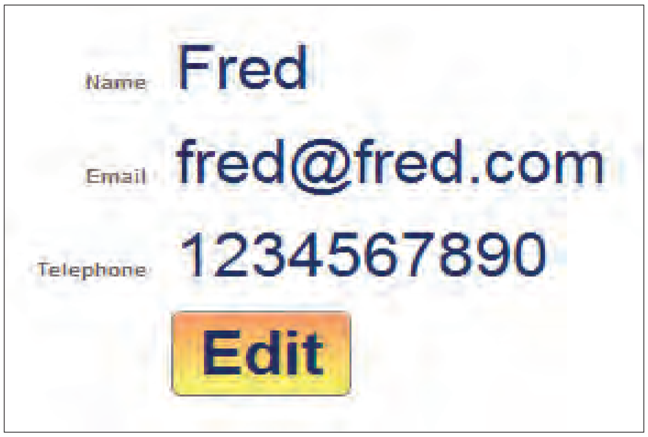


Figure 4 Charts and graphs



Fred

fred@fred.com

1234567890

Figure 5 Simple form with edit button

- ***accessible:*** If set to true generates screen reader accessible flash forms. This adds about 64k to the size of the final generated SWF file.
- ***global-css-url:*** Tells the compiler where to find the CSS file, which is used to determine how your forms will look. I set mine to "/global.css," which means it looks for a global.css file in the root of my Web server (thus allowing different CSS for each Web site on the server).

You can set default attributes for each of the controls and containers available to Flex, as well as setup styles that can be applied to individual controls via the "styleName" attribute present on all of these. See Listing 5 for a sample CSS file.

Hopefully simply discovering that you can use the PopUpManager to load native Flex has set your minds to thinking about all the things you can now achieve. I'm going to show you one more example showing a simple form with a modal pop-up. I'll show you how to pass a data structure back and forth, and also show you a way around a well-known Flex bug.

We start with a simple form that shows three fields and an edit button (note: The CSS file listed in Listing 5 has been used to style this form). Figure 5 shows our example page, and Figure 6 shows a popup form to edit the details.

As you can see in Listing 6, the three text fields are bound to values in userObj, the data that's initialized in the function initData() that's called when the form loads. When you click the edit button, this code is run:

```
mx.managers.PopUpManager.createPopUp(this, UserEditForm, true,{parentref:
this,userobject:userObj});
```

This opens our pop-up form (userEditForm.mxml – see Figure 6 and Listing 7) as a modal window. It also passes two pieces of data to our new window: parentref, which allows the pop-up to communicate back to our main form, and userobject, the object containing our user data. Also note the function setUserData() that our pop-up window will use to send the edited data back to the main form. This is defined as a public function, not private like our other (internal) functions.

As with the main form, the TextInput controls' values are bound to the data object, this time to userobject, which was passed from the parent form:

```
    <mx:TextInput id="namefield" text="{userobject.name}" />
```

Looking at the function saveUser you'll notice that we validate before saving (Figure 6 shows that our form input values are not validated!) by making this call:

```
if (mx.validators.Validator.isStructureValid(this,'userdata'))
```

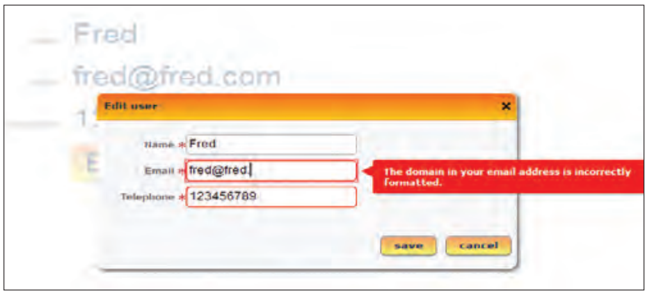
Notice though that rather than validate the input controls themselves I'm validating userdata. This is a data model that holds all the user data in one object, and it is this that's passed back to the parent window after saving.

```
<mx:Model id="userdata">
   <user>
      <name>{namefield.text}</name>
      <phone>{phonefield.text}</phone>
     <email>{emailfield.text}</email>
   </user>
</mx:Model>
```

This in turn is bound to the input controls, and as such is always "up-to-date." Validation is handled using the Flex validation controls:

```
<mx:StringValidator             field="userdata.user.name"/>
<mx:PhoneNumberValidator        field="userdata.user.phone"/>
<mx:EmailValidator              field="userdata.user.email"/>
```

This ensures that the user supplies required data and that the telephone number and e-mail address are correctly formatted.



The domain in your email address is incorrectly formatted.

Figure 6 Popup edit form

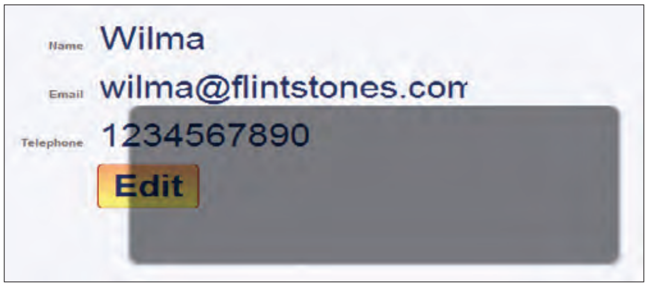


Figure 7 Drop shadow bug

Assuming that the user supplies all the correct data then the save function eventually passes the data back to the parent window before closing itself. It does this by using the parentref that our main form passed when it opened the window and calls the public function that we created there to get the updated data.

```
parentref.setUserData(userdata.user);
```

The setUserData function in Demo5.mxml simply assigns the data objects passed to it to userObj, the object to which our Label fields are bound, so they immediately update with the newly saved values.

Now for a few "gotchas" that you'd might like to watch out for. The first is the pop-up window's close button. You'll notice in the code that I disable the entire TitleWindow control when I'm saving to prevent the user from making any further changes or pressing submit a second time. That works great, but the close button is always active. Function closePopup checks the window's enabled status prior to closing.

In this example the window closes just fine. However, there's a well-known Flex bug whereby, if you close a pop-up window with a drop shadow in the response handler of a remoting call then the window is removed but the drop shadow remains behind (see Figure 7).

One solution is simply to use styles to disable the default drop shadow. If, however, you'd like to use the drop shadow then these two lines of code will sort the problem out for you:

```
    windowObject.setStyle("dropShadow",false);
    windowObject.redraw(true);
```

The first simply removes the drop shadow. We do this before closing the window. The second is needed because of the way flash does its redraws. Without it you'll find that the window is closed before the drop shadow is removed. The code forces an immediate redraw. See Listing 8 for an example that uses remoting. Simply comment out those two lines to see the shadow bug in action. Listing 9 is a very simple CFC that handles the saveUser function.

Finally, few other things to note:

- In userEditForm.mxml (Listings 7/8), you'll notice the <mx:TitleWindow> tag has a modalTransparency attribute that I've set to "75." This makes everything behind the pop-up window 75% transparent. This looks great and is a clear indicator to the user that the remaining application is disabled while the pop-up is open. See Figure 5.
- You can center your pop-up by adding this.centerPopUp() to your load attribute.

## About the Author

*Ian Bale is the technical director of Celtic Internet ltd (www.CelticInternet.com), an Internet services and consulting company based in the UK. Since graduating with a degree in computer science in 1990, he has worked as a software engineer in the aircraft and telecommunications industries. In 1999 he moved into Internet work and immediately specialized in ColdFusion and Flash and, more recently, Flex.*
*ian@celticinternet.com*

# Adobe Flex 2: Advanced DataGrid

## Part 1: Destination-awareness formatters & renderers

By Victor Rasputnis, Yakov Fain, and Anatole Tartakovsky

In any GUI tool, one of the most popular components is the one that shows data in a table format like JTable in Java or Datawindow in PowerBuilder. The Adobe Flex 2 version of such a component is called DataGrid. In any UI framework, the robustness of such a component depends on formatting and validating utilities as well as a whole suite of data input controls: CheckBoxes, ComboBoxes, RadioButtons, all sorts of Inputs, Masks, and so on. Using theatrical terminology, the role of the king is played by his entourage. Practically speaking, touching up the DataGrid is touching up a large portion of the Flex framework.

We'll start by upgrading the standard DataGrid to a "destination-aware" control capable of populating itself. Next, we'll look at the task of formatting DataGrid columns and that would naturally lead us to a hidden treasury of the Flex DataGrid – the DataGridColumn, which, once you start treating it like a companion rather than a dull element of MXML syntax, can help you do truly amazing things.

After setting the stage for the DataGrid's satellite controls, we'll lead you through making a professional library with a component manifest file and namespace declaration that supports flexible mappings of your custom tags to the required hierarchy of implementation classes.

### Making DataGird Destination-Aware

Introducing Flex remoting substantially increased the size of our application code. Imagine a real-world application with two-dozen different ComboBoxes or DataGrids. If we put all the relevant RemoteObjects in a single application file it will become unmanageable in no time. For similar reasons, any decent sized application is usually partitioned into different files.

As far as partitioning goes, there's a compelling argument to encapsulate instantiation and interoperation with RemoteObject inside the visual component, making it a destination-aware component, if you will. Embedding a Remote Object directly into the DataGrid will make it fully autonomous and reusable. This design approach sets application business logic free from low-level details like mundane remoting.

The concept of destination-aware controls eliminates the tedious effort required to populate your controls with Remoting or DataServices-based data. Here's how an MXML application with a destination-aware DataGrid might look if we apply the familiar remoting destination com_theriabook_composition_EmployeeDAO:

```
<!-- DestinationAwareDataGridExDemo.mxml-->
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx=http://www.adobe.com/2006/mxml
layout="vertical"
              xmlns:fx="com.theriabook.controls.*">
    <fx:DataGridEx id="dg"
          destination="com_theriabook_composition_EmployeeDAO"
          method="getEmployees"
          creationComplete="dg.fill()"
    />
</mx:Application>
```

In Listing 1 we have sub-classed the standard Flex DataGrid.

The data are coming from a server-side database with the help of a Java object called EmployeeDAO. If we run the application, our screen will show the grid of employee records in its default formatting:. (See Figure ).

### Formatting with labelFunction

We have just looked at the default data formatting provided by DataGrid out-of-the-box. The easiest way to improve column formatting is by supplying a labelFunction for each column that requires extra attention:

```
        <mx:DataGridColumn   dataField="PHONE"  labelFunction="ph
oneLabelFunction" />
```

Once the labelFunction is declared it automatically gets called by the DataGrid, which supplies the appropriate data item plus information about the column so that you can technically apply one function to more than one column. As far as the formatting techniques themselves, Flex offers plenty of pre-defined formatters, which can be used out-of-the-box or customized to your specific needs. For instance, to format Social Security numbers (SS_NUMBER), we could have used mx.formatters.SwitchSymbolFormatter to create the following function:

```
          import mx.formatters.SwitchSymbolFormatter;
          private var sf:SwitchSymbolFormatter;
          private function ssnLabelFunction(item:Object, column:
DataGridColumn):String {
             if (!sf) {
                sf = new SwitchSymbolFormatter();
             }
             return sf.formatValue("###-##-####", item["SS_NUMBER"]);
             }
```

Then inside the DataGridColumn we can mention this function name:

```
             <mx:DataGridColumn  dataField="SS_NUMBER" labelFunction="s
snLabelFunction"  />
```

Similarly, in Listing 2 we can apply the same technique to the PHONE field setting the formatString to (###)###-####.

Figure 2 shows how our formatting looks on the screen.

## Formatting with Extended DataGridColumn

The labelFunction formatting does the job. The price tag is hard-coding labelFunction(s) names in the DataGridColumn definitions. If you packaged a set of label functions in the class mydomain.LabelFunction your column definitions might look like this:

```
   <mx:DataGridColumn   dataField="PHONE"  labelFunction="mydomain.
LabelFunctions.phone" />
   <mx:DataGridColumn   dataField="SS_NUMBER" labelFunction=" mydomain.
LabelFunctions.ssn"  />
```

A more pragmatic approach is to provide the formatString as an extra attribute to the DataGridColumn and have it encapsulate the implementation details. We're talking of the following alternative:

```
   <fx:DataGridColumn   dataField="PHONE"  formatString="phone" />
   <fx:DataGridColumn   dataField="SS_NUMBER" formatString="ssn"  />
```

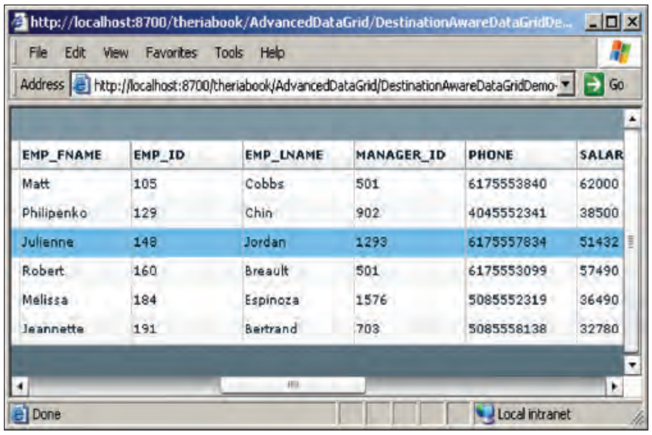
Implementing such syntax is within arm's reach. We just have to extend – well, not the arm, but the DataGridColumn so that instead of mx:DataGridColumn we would use our, say, fx: DataGridColumn. The mx.controls.dataGridClasses.DataGridColumn is just a respository of styles and properties to be used by the DataGrid. In the Flex class hierarchy it merely extends CSSStyleDeclaration. Nothing prevents us from extending it further and adding an extra attribute. In the case of formatString the setter function will delegate assigning the labelFunction to the helper FormattingManager:

```
   public class DataGridColumn  extends mx.controls.dataGridClasses.
DataGridColumn {
       public function  set formatString( fs:String ) : void{
          FormattingManager.setFormat(this, fs);
       }
   }
```

Wait a minute, where did the FormattingManager come from? We'll get to the implementation of this class later. At this point, we have to eliminate the possible naming collision between our to-be-made DataGridColumn and the standard mx.controls.dataGridClasses.DataGridColumn.

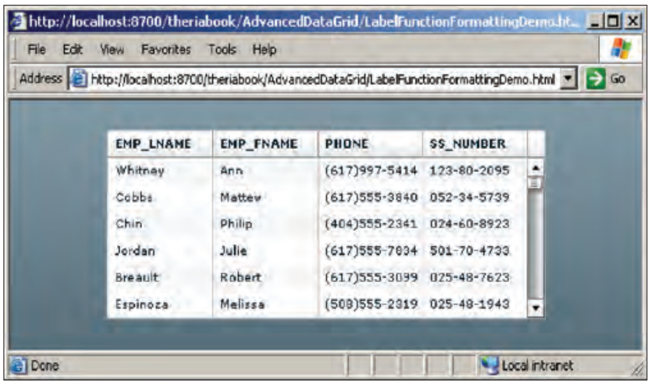
## Introducing the Component Manifest File

Up till now we've been keeping folders with classes of our custom components under the folder of application MXML files. To reference these components we've been declaring namespaces pointing to some hard-coded, albeit relative paths, such as xmlns:lib="com.theriabook.controls.*" or xmlns="*". The problem with this approach is that these namespaces point to one folder at a time. As a result, we either end up with multiple custom name spaces or we have a wild mix of components in one folder.


http://localhost:8700/theriabook/AdvancedDataGrid/DestinationAwareDataGridDe...
File Edit View Favorites Tools Help
Address http://localhost:8700/theriabook/AdvancedDataGrid/DestinationAwareDataGridDemo-
Go
Done
Local intranet

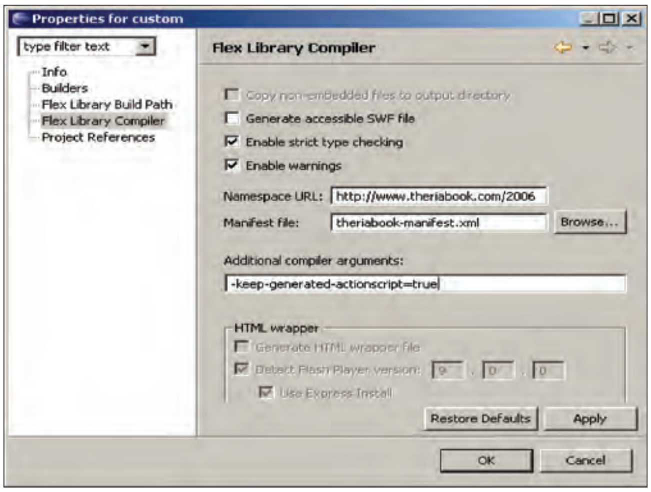
| EMP_FNAME | EMP_ID | EMP_LNAME | MANAGER_ID | PHONE | SALAR |
|---|---|---|---|---|---|
| Matt | 105 | Cobbs | 501 | 6175553840 | 62000 |
| Philipenko | 129 | Chin | 902 | 4045552341 | 38500 |
| Julienne | 148 | Jordan | 1293 | 6175557834 | 51432 |
| Robert | 160 | Breault | 501 | 6175553099 | 57490 |
| Melissa | 184 | Espinoza | 1576 | 5085552319 | 36490 |
| Jeannette | 191 | Bertrand | 703 | 5085558138 | 32780 |

Figure 1 Demo of "destination-aware" DataGrid running


http://localhost:8700/theriabook/AdvancedDataGrid/LabelFunctionFormattingDemo.ht...
File Edit View Favorites Tools Help
Address http://localhost:8700/theriabook/AdvancedDataGrid/LabelFunctionFormattingDemo.html
Go
Done
Local intranet

| EMP_LNAME | EMP_FNAME | PHONE | SS_NUMBER |
|---|---|---|---|
| Whitney | Ann | (617)997-5414 | 123-80-2095 |
| Cobbs | Mattew | (617)555-3840 | 052-34-5739 |
| Chin | Philip | (404)555-2341 | 024-60-8923 |
| Jordan | Julie | (617)555-7834 | 501-70-4733 |
| Breault | Robert | (617)555-3099 | 025-48-7623 |
| Espinoza | Melissa | (508)555-2319 | 025-48-1943 |

Figure 2 Screenshot of the running LabelFunctionFormattingDemo


Properties for custom

Flex Library Compiler

Info
Builders
Flex Library Build Path
Flex Library Compiler
Project References

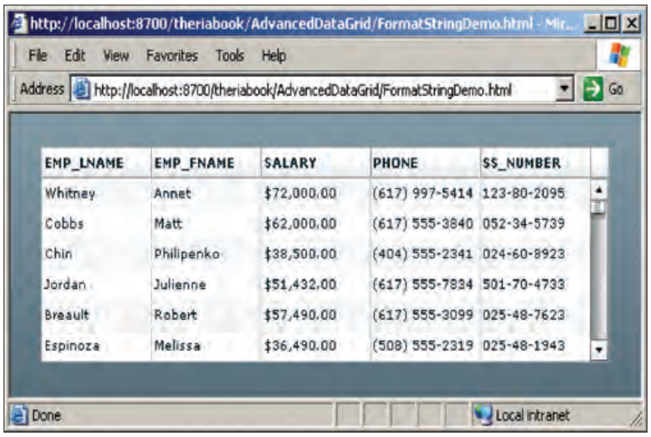
☐ Copy non-embedded files to output directory
☐ Generate accessible SWF file
☑ Enable strict type checking
☑ Enable warnings

Namespace URL: http://www.theriabook.com/2006
Manifest file: theriabook-manifest.xml

Additional compiler arguments:
-keep-generated-actionscript=true

HTML wrapper
☐ Generate HTML wrapper file
☑ Detect Flash Player version: 9 . 0 . 0
☑ Use Express Install

Figure 3 Manifest file and namespace definition for Flex Library Project


http://localhost:8700/theriabook/AdvancedDataGrid/FormatStringDemo.html

| EMP_LNAME | EMP_FNAME | SALARY | PHONE | SS_NUMBER |
|---|---|---|---|---|
| Whitney | Annet | $72,000.00 | (617) 997-5414 | 123-80-2095 |
| Cobbs | Matt | $62,000.00 | (617) 555-3840 | 052-34-5739 |
| Chin | Philipenko | $38,500.00 | (404) 555-2341 | 024-60-8923 |
| Jordan | Julienne | $51,432.00 | (617) 555-7834 | 501-70-4733 |
| Breault | Robert | $57,490.00 | (617) 555-3099 | 025-48-7623 |
| Espinoza | Melissa | $36,490.00 | (508) 555-2319 | 025-48-1943 |

Figure 4 DataGrid formatted with the columns' formatString attributes

To break the spell and abstract the namespace from the exact file location we need to use the component manifest file. Component manifest is an XML file that allows mapping of component names to the implementing classes. Below is the example of component manifest, which combines our custom DataGrid and DataGridColumn located in different folders:

```
<?xml version="1.0"?>
<componentPackage>
    <component id="DataGrid" class="com.theriabook.controls.DataGrid"/>
    <component id="DataGridColumn"
          class="com.theriabook.controls.dataGridClasses.DataGridCol-
umn"/>
</componentPackage>
```

To benefit from using this component manifest you have to compile your components with the compc or use the Flex Library project. To be more specific, you have to instruct compc to select the URL that your application can use later in place of the hard-coded folder in the xmlns declaration. So we'll create a new FlexLibrary project – theriabook, where we'll put the theriabook-manifest.xml containing the XML above and set the relevant project properties, as shown in Figure 3.

Now we can move DataGrid from our application project to theriabook and replace xmlns:fx="com.theriabook.controls" with xmlns:fx="http://www.theriabook.com/2006" provided that our application project will include a reference to theriabook.swc. As a result our application will reference fx:DataGrid and fx:DataGridColumn irrespective to their physical location.

Having done that, let's get back to the custom DataGridColumn.

## More on Customizing the DataGridColumn

We'll put our DataGridColumn in the dataGridClasses subfolder as a sign of our respect for the well-thought-out directory structure of the Flex framework:

As mentioned above, the "dirty" job of locating and assigning the proper label function has been delegated to the helper class FormattingManager. This class, presented in Listing 4, should be put in our theriabook project.

Tada! And the winner is…the developer. Once we add theriabook.swc (with DataGrid, DataGridColumn, and FormattingManager) to the library path, the application code gets reduced to Listing 5.

## Improving FormattingManager

In the previous examples we've used SwitchSymbolFormatter for both phone and ssn formatting. As soon as we start formatting numbers or currency values, it's natural to use NumberFormatter or CurrencyFormatter – descendants of mx.formatters.Formatter. In fact, Flex offers a dedicated formatter even for the phone formatting.

While SwitchSymbolFormatter derives from Object, all the rest of the formatters descend from Formatter. By encapsulating this specific of SwitchSymbolFormatter in the custom class MaskFormatter we'll help ourselves in basing the next version of FormattingManager entirely on Formatters as in Listing 6.

Look how this MaskFormatter simplifies our FormattingManager: we can replace all the private methods with an anonymous function, as shown in Listing 7. Please note that the reference to the appropriate formatter is preserved with the closure.

The testing application FormatStringDemo is in Listing 8. (Listing 8 -16 can be downloaded from the online version of ths article at http://coldfusion.sys-con.com.) If you run it, the DataGrid dg will be formatted as shown in Figure 4:

Let's focus on the hard-coding that we allowed in case of the money value:

```
case "money":
    formatter = new CurrencyFormatter();
    CurrencyFormatter(formatter).precision=2;
break;
```

This hard-coding reflects, perhaps, the most "popular" case. But what if we want to have the full advantage of the proper-

REGISTER TODAY AND $AVE!

**Rich Internet Applications: AJAX, Flash, Web 2.0 and Beyond...**

www.AjaxWorldExpo.com

# AJAXWORLD™ EAST CONFERENCE & EXPO

## NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

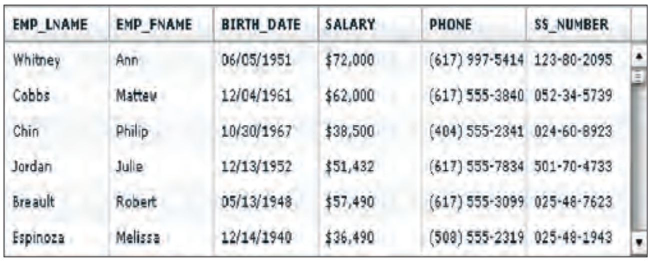
**SYS-CON Events is proud to announce the AjaxWorld East Conference 2007!**

**The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.**

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this unique, timely conference – especially the web designers and developers building those experiences, and those who manage them.

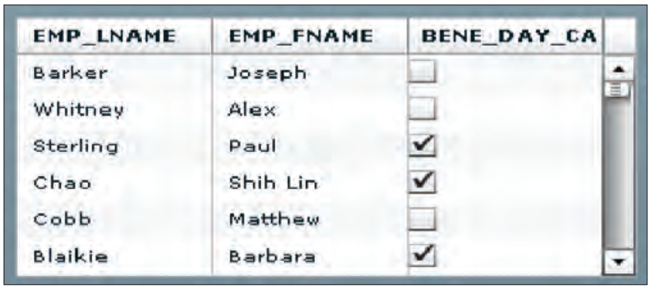
**BEING HELD MARCH 19 - 21, 2007!**

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested in learning about a wide range of RIA topics that can help them achieve business value.



| EMP_LNAME | EMP_FNAME | BIRTH_DATE | SALARY | PHONE | SS_NUMBER |
|---|---|---|---|---|---|
| Whitney | Ann | 06/05/1951 | $72,000 | (617) 997-5414 | 123-80-2095 |
| Cobbs | Mattew | 12/04/1961 | $62,000 | (617) 555-3840 | 052-34-5739 |
| Chin | Philip | 10/30/1967 | $38,500 | (404) 555-2341 | 024-60-8923 |
| Jordan | Julie | 12/13/1952 | $51,432 | (617) 555-7834 | 501-70-4733 |
| Breault | Robert | 05/13/1948 | $57,490 | (617) 555-3099 | 025-48-7623 |
| Espinoza | Melissa | 12/14/1940 | $36,490 | (508) 555-2319 | 025-48-1943 |

**Figure 5 DataGrid formatted with formatString and formatData attributes**



| EMP_LNAME | EMP_FNAME | BENE_DAY_CA |
|---|---|---|
| Barker | Joseph | ☐ |
| Whitney | Alex | ☐ |
| Sterling | Paul | ✓ |
| Chao | Shih Lin | ✓ |
| Cobb | Matthew | ☐ |
| Blaikie | Barbara | ✓ |

**Figure 6 Custom CheckBox used as drop-in renderer with extended properties**

ties of the corresponding formatter, such as precision, in case of CurrencyFormatter? To address these cases we're going to introduce one more fx:DataGridColumn property – formatData. Here's how it will be used in the application MXML:

```
<fx:DataGridColumn    dataField="SALARY" >
    <fx:formatData>
        <mx:Object formatString="money" precision="0"/>
    </fx:formatData>
</fx:DataGridColumn>
```

The elegance of MXML lets us implement this extension with just a few lines of extra code in com.theriabook.controls.dataGridClasses.DataGridColumn:

```
public function  set formatData(fd :Object) : void{
   FormattingManager.setFormat(this, fd);
}
```

Then, to accommodate the change on the FormattingManager side, we'll iterate through all the properties of the formatData object and attempt to assign them to the appropriate properties of the formatter with an emphasis on the word appropriate. The MXML compiler isn't going to help us check the properties of the unsealed <mx:Object> against the properties of the formatter. Accordingly, to protect ourselves from the no-such-property-exceptions, we surround property assignments with try/catch:

```
public static function setFormat(
        dgc:mx.controls.dataGridClasses.DataGridColumn,
        formatData:Object
):void {
    .  .  .  .  .
    if (!(formatData is String)) {
        for (var property:String in formatData) {
    try {
                formatter[property] = formatData[property];
            } catch (err:Error) {
                // Property does not match formatter type
            }
        }
    }
    .  .  .  .  .
}
```

The complete listing of renewed FormattingManager is in Listing 9. While maintaining your own framework, you would transform this class to accommodate your requirements

Finally, Listing 10 has the sample application to test our changes, FormatDataDemo.

When you run the application it will produce the DataGrid shown in Figure 5.

We'll continue beefing up our custom DataGridColumn after a short detour into CheckBox and RadioButton controls.

## CheckBox as a Drop-In Renderer

As we warned the reader at the beginning of this article, DataGrids rarely come alone. In this section we're going to suggest customizimg the CheckBox, which will help us illustrate custom DataGridColumns from a different perspective.

The state of a CheckBox control is managed by the Boolean property selected. At the same time, many business systems use either Y/N or, sometimes, 0/1 flags. As a result, translating business-specific values into selected and vice versa burdens the application code. Listing 11 presents the custom CheckBox, which supports application-specific on and off values along with the current value.

So, using this CheckBox, we could have written:

```
<fx:CheckBox value="Y" onValue="Y" offValue="N"  />
```

to have selected CheckBox, or

```
<fx:CheckBox value="N" onValue="Y" offValue="N"  />
```

to set selected to false.

## DataGridColumn as ItemRenderer's Knowledge Base

Now let's get back to the DataGrid world. What if we wanted to use our CheckBox as the DataGrid item renderer? Here's the suggested use case example:

```
<fx:DataGridColumn  dataField="BENE_DAY_CARE"
        itemRenderer="com.theriabook.controls.CheckBox" >
</fx:DataGridColumn>
```

Obviously, we need to modify the CheckBox more to take

Subscribe Today!

SAVE 16%

12 Issues for $89.99

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SPECIAL ISSUE: COLDFUSION SECURITY

COLDFUSION Developer's Journal

ColdFusionJournal.com

September 2004 Volume: 6 Issue: 9

SANDBOX SECURITY

By Alex Hübner page 8

Web Services Edge 2005 East February 15-17, 2005 Hynes Convention Center Boston, MA

Editorial Reaching Deeper Simon Horwith page 5

Community Focus Problems Securing Your Applications? Simon Horwith page 7

CF101 ColdFusion's Application Framework Jeffry Houser page 28

CFUGs ColdFusion User Groups page 40

FOUNDATIONS 'Selling' ColdFusion Hal Helms page 46

Security: ColdFusion Security Best Practices Knowing the security risks are there is half the battle — Bryan Murphy 16

Security: Public-Key Encryption Making strong encryption nearly painless — Wayne Graham 20

cfcUnit: ColdFusion Unit Testing Framework Paul Kenney's version — Harry Klein 30

<bf> on< cf>: Defending ColdFusion Against... — Ben Forta 36

Real-World Uses: Using ColdFusion for Network Monitoring Find the exact piece of data you need — Ryan Emerle, Matthew I. Fusfield 42

Security: Top Ten Web Security Tips You can't be too careful — Michael Smith 48

AN ADVANCE LOOK AT BLAC

COLDFUSION

Leveragin or Just F

Editorial Blackstone Robert Diamond page 5

CF Community Tales from the List Simon Horwith page 7

cf101 What's the Best Approach to Software Development? Jeffry Houser page 32

CFUGs ColdFusion User Groups page 42

Training: Fast Track to Fl

CFML: Harnessing the P SQL Server Using Store

Foundations: A We

<BF> on <CF>: C and Data Abstra

PDFs: Creatin Your CF Appl

- Exclusive feature articles
- Latest *CFDJ* product reviews
- Interviews with the hottest names in ColdFusion
- Code examples you can use in your applications
- *CFDJ* tips and techniques

That's a savings of $29.89 off the annual newsstand rate. Visit our site at www.sys-con.com/coldfusion or call 1-800-303-5282 and subscribe today!

COLDFUSION Developer's Journal

care of the value within the data setter:

```
override public function set data(item:Object):void
{
    super.data = item;
    if( item!=null ) {
        value = item[DataGridListData(listData).dataField];
    }
}
```

But how will we communicate the offValue and onValue properties to our CheckBox-turned-itemRenderer? Ideally, we would need something like:

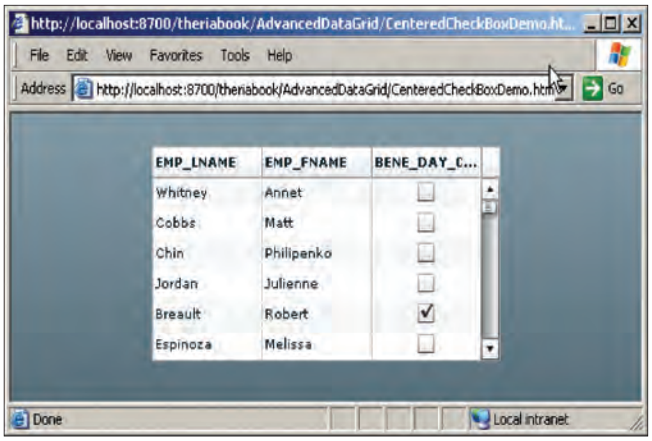
```
<fx:DataGridColumn  dataField="BENE_DAY_CARE"
            itemRenderer="com.theriabook.controls.CheckBox" >
    <fx:extendedProperties>
        <mx:Object onValue="Y" offValue="N" />
    </fx:extendedProperties>
</fx:DataGridColumn>
```

Flex creators thought of this in advance. An object referenced by itemRenderer is not a CheckBox but rather an instance of mx.core.ClassFactory wrapped around the CheckBox. The mechanism of mx.core.ClassFactory allows Flex to generate instances of another class – in our case com.theriabook.controls.CheckBox. Importantly, each instance created by the factory is assigned identical properties borrowed from the properties property of the factory object. Accordingly, all we have to do is pass the value of the extendedProperties as the properties of the itemRenderer as shown in Listings 12.

The test application, ExtendedPropertiesDemo, is in Listing 13. When you run it, it produces the data grid shown in Figure 6.

We should have mentioned the alternative run-of-the-mill approach with inline itemRenderer. It's in Listing 14.

Arguably, the extendedProperties approach is more efficient, since it absolves MXML of generating an extra nested class (mx:Component) for each column of this kind and we've introduced you to yet another mean of customizing a DataGridColumn. We'll continue building on top of it in the following sections.


http://localhost:8700/theriabook/AdvancedDataGrid/CenteredCheckBoxDemo.ht...
File
Edit
View
Favorites
Tools
Help
Address
http://localhost:8700/theriabook/AdvancedDataGrid/CenteredCheckBoxDemo.htm
Go
Done
Local intranet

| EMP_LNAME | EMP_FNAME | BENE_DAY_C... |
|---|---|---|
| Whitney | Annet | ☐ |
| Cobbs | Matt | ☐ |
| Chin | Philipenko | ☐ |
| Jordan | Julienne | ☐ |
| Breault | Robert | ✓ |
| Espinoza | Melissa | ☐ |

**Figure 7 CenteredCheckBoxDemo screenshot**

## Nitpicking CheckBox

There are some additional remarks that we ought to add to our CheckBox implementation at this point. The first one is related to the horizontal alignment of the CheckBox. Instincts tell us that label-free checkbox should be centered in the column, rather than stuck in the leftmost position. At first, you may try the textAlign style of the DataGridColumn to no avail. Then you may resort to another run-of-the-mill approach to center the Checkbox by putting it inside a container, such as HBox. Here's the performance-based advice endorsed by Flex Framework engineers: avoid containers inside the datagrid cell at any reasonable cost. In particular, instead of using HBox, why not sub-class the CheckBox and override the updateDisplayList method? It gets quite natural once you've stepped on this path, so we'll add the code shown below to our CheckBox (the complete code of com.theriabook.controls.CheckBox is in Listing 15):

```
  import mx.core.mx_internal;
  use namespace mx_internal;
    .  .  .  .
    override protected function updateDisplayList(
        unscaledWidth:Number, unscaledHeight:Number):void
    {
        super.updateDisplayList(unscaledWidth, unscaledHeight);
        if (currentIcon) {
            var style:String = getStyle("textAlign");
            if ((!label) && (style=="center") ) {
                currentIcon.x = (unscaledWidth - currentIcon.measured-
Width)/2;
            }
        }
    }
```

Please note the use of namespace mx_internal. It's required so the reference to currentIcon visualizes the checkbox picture, since the currentIcon, the child of the original CheckBox, is originally scoped as mx_internal.

Now we modify the testing application to include textAlign="center":

```
<fx:DataGridColumn  dataField="BENE_DAY_CARE" textAlign="center"
    itemRenderer="com.theriabook.controls.CheckBox" >
    <fx:extendedProperties>
        <mx:Object onValue="Y" offValue="N" />
    </fx:extendedProperties>
</fx:DataGridColumn>
```

And, when we run it, all checkboxes are in their proper places:

The second nitpicking point is related to undefined as the possible value of a property. Under our current business scenario, we can assume that some of the employees are not eligible for day care benefits, and relevant items in the dataProvider's collection are lacking BENE_DAY_CARE property, which can be expressed as item.BENE_DAY_CARE=="undefined" for dynamic items.

Does it make sense to show checkboxes for non-eligible employees? Perhaps, it doesn't. In these cases we would make currentIcon invisible. You may select a different approach and show a fuzzy checkbox image instead, but that is beside the point. The following modification of updateDisplayList does the job of removing checkBox when the value is undefined:

```
override protected function updateDisplayList(unscaledWidth:Number,

                                 unscaledHeight:Number):void
    {

        super.updateDisplayList(unscaledWidth, unscaledHeight);
        if (currentIcon) {
            var style:String = getStyle("textAlign");
            if ((!label) && (style=="center") ) {
                currentIcon.x = (unscaledWidth - currentIcon.measured-
Width)/2;
            }
            currentIcon.visible = (_value!=undefined);
        }
    }
```

To accommodate this change we also have to loosen up the class definitions for value as shown in Listing 15, where we change Object to undefined.

Next and probably the most important fix is that our CheckBoxes have been silenced. Try to click on one, scroll the row out of view, and scroll it back in. The checkbox doesn't retain your selection and it shouldn't: we have never communicated the change to the underlying data. To remedy this situation we'll add the constructor method, where we'd start listening on the "change" event; once an event is intercepted we'll modify the data item with the CheckBox value. That, in turn, will result in either onValue or offValue as per our value getter:

```
    public function CheckBox() {
        super();
        addEventListener(Event.CHANGE,
            function(event:Event):void{
                if (listData) {
                    data[DataGridListData(listData).dataField] = value;
                }
            }
        );
    }
```

The complete code for the second version of CheckBox is in Listing 15.

Next comes the test application in Listing 16. We've added the "Revoke day care benefit" button, which makes DAY_CARE_BENE undefined on the currently selected DataGrid item. We also had to notify the collection with the itemUpdated() call.

When you run the application, you'll see that checkboxes retain the selection after scrolling out and back into view. If you click the "Revoke" button for the first two rows, you're going to see a picture similar to Figure 8.

The last CheckBox fix will come in handy once you declare the DataGrid editable. Why declare it editable in the first place if we seem to be editing the DataGrid already? Let's not forget that the only field we've edited so far is the checkbox BENE_DAY_CARE. Should you also decide to allow editing the text fields you'd have to change the definition of the DataGrid as shown in bold in this snippet:

```
    <fx:DataGrid  id="dg" creationComplete="dg.fill();dg.selectedIndex=0;"
editable="true"
        destination="com_theriabook_composition_EmployeeDAO"
method="getEmployees"
    >
```

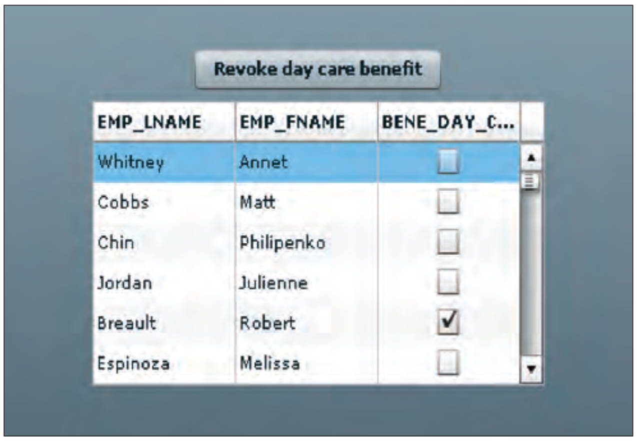
But once you do that, a click on our beautiful checkbox would turn it into default editor – TextInput, quite like in a Cinderella story. To make the miracle last, you'd declare that your renderer is good to go as an editor as well:

```
    <fx:DataGridColumn  dataField="BENE_DAY_CARE" textAlign="center"
            itemRenderer="com.theriabook.controls.CheckBox"
rendererIsEditor="true">
            .  .  .
    </fx:DataGrid>
```

By default, DataGrid reads the text property of the item editor. You can nominate a different property via editorDataField (in our case that would be value). Alternatively, and what will help us later, you can "upgrade" the checkbox to carry the text property itself:

```
    public function set text(val:String) :void {
        value = val;
    }

    public function get text():*  {
        return value;
    }
```



| EMP_LNAME | EMP_FNAME | BENE_DAY_C... |
|---|---|---|
| Whitney | Annet | ☐ |
| Cobbs | Matt | ☐ |
| Chin | Philipenko | ☐ |
| Jordan | Julienne | ☐ |
| Breault | Robert | ☑ |
| Espinoza | Melissa | ☐ |

**Figure 8 UndefinedCheckBoxDemo partial screenshot**

## Summary

While DataGrid is a powerful component right off-the-shelf, but the fact that it's truly extendable can substantially increase its usability. In the next part of this article, we'll continue experimenting with the DataGrid by using radio buttons as renderers and introduce computed columns.

## About the Authors

*Dr. Victor Rasputnis is a managing principal of Farata Systems. He's responsible for providing architectural design, implementation management and mentoring to companies migrating to XML Internet technologies. He holds a PhD in computer science from the Moscow Institute of Robotics.*

*Yakov Fain is a managing principal of Farata Systems. He's authored several Java books, dozens of technical articles. SYS-CON Books will be releasing his latest book, "Rich Internet Applications with Adobe Flex and Java: Secrets of the Masters" this Fall. Sun Microsystems has nominated and awarded Yakov with the title Java Champion. He leads the Princeton Java Users Group. Yakov teaches Java and Flex 2 at New York University. He is Adobe Certified Flex Instructor.*

*Anatole Tartakovsky is a managing principal of Farata Systems. He's responsible for creation of frameworks and reusable components. Anatole authored number of books and articles on AJAX, XML, Internet and client-server technologies. He holds an MS in mathematics.*

*vrasputnis@faratasystems.com*

*yfain@faratasystems.com*

*atartakovsky@faratasystems.com*

### Listing 1 DataGrid.as, the first version with "destination-awareness"

```
// DataGrid.as
package com.theriabook.controls {
  import mx.controls.DataGrid;

  public class DataGrid  extends mx.controls.DataGrid {

  import mx.rpc.remoting.mxml.RemoteObject;
  import mx.rpc.AbstractOperation;
  import mx.rpc.events.*;
  import mx.controls.Alert;
  import mx.managers.CursorManager;

  public var destination:String=null, method : String = null;
  public var autoFill : Boolean = true;
  protected var ro:RemoteObject = null;

  public function fill(... args): void {
     if( ro==null ) {
        if( destination==null || destination.length==0 )
           throw new Error("No destination specified");
        if( method==null || method.length==0 )
           throw new Error("No retrieveMethod specified");

        ro = new RemoteObject(destination);
        ro.showBusyCursor = true;
        ro.concurrency    = "last";
        ro.addEventListener(ResultEvent.RESULT, ro_onResult);
        ro.addEventListener(FaultEvent.FAULT,   ro_onFault);
     }
     var operation:AbstractOperation = ro.getOperation(method);
     operation.arguments = args;
     operation.send();

  }
  private function ro_onFault(evt:FaultEvent):void {
     CursorManager.removeBusyCursor();
     Alert.show("Failed retrieving data: "+evt.message, "[Destination-
AwareDataGrid]" + id);
  }

  private function ro_onResult(evt:ResultEvent):void {
     CursorManager.removeBusyCursor();
     if (evt.result.length != 0)
        dataProvider = evt.result;
  }
  }
}
```

### Listing 2 LabelFunctionFormattingDemo.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- LabelFunctionFormattingDemo.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
layout="vertical" xmlns:fx="com.theriabook.controls.*">
  <fx:DataGrid  id="dg" creationComplete="dg.fill()"
     destination="com_theriabook_composition_EmployeeDAO"
method="getEmployees"
  >
     <fx:columns>
        <mx:Array>
           <mx:DataGridColumn   dataField="EMP_LNAME"   />
           <mx:DataGridColumn   dataField="EMP_FNAME"  />
           <mx:DataGridColumn   dataField="PHONE"  labelFunction="ph
oneLabelFunction" />
           <mx:DataGridColumn   dataField="SS_NUMBER" labelFunction=
"ssnLabelFunction"  />
        </mx:Array>
     </fx:columns>
  </fx:DataGridEx>
  <mx:Script>
     <![CDATA[
        import mx.formatters.SwitchSymbolFormatter;
        private var sf:SwitchSymbolFormatter;
        private function ssnLabelFunction(item:Object, column:
 DataGridColumn):String {
           if (!sf) {
              sf = new SwitchSymbolFormatter();
           }
           return sf.formatValue("###-##-####", item["SS_NUMBER"]);
        }
        private function phoneLabelFunction(item:Object, column:
DataGridColumn):String {
            if (!sf) {
                sf = new SwitchSymbolFormatter();
            }
            return sf.formatValue("(###)###-####", item[column.
dataField]);
        }
      ]]>
  </mx:Script>
</mx:Application>
```

### Listing 3 DataGridColumn.as, the first version

```
// DataGridColumn.as (first version)
package com.theriabook.controls.dataGridClasses  {
  import mx.controls.dataGridClasses.DataGridColumn;

  public class DataGridColumn  extends mx.controls.dataGridClasses.
DataGridColumn {
      public function  set formatString( fs:String ) : void{
          FormattingManager.setFormat (this, fs);
      }
  }
}
```

### Listing 4 FormattingManager.as, the first version

```
// FormattingManager.as, first version

package com.theriabook.controls.dataGridClasses
{
  public class FormattingManager
  {
          import mx.controls.dataGridClasses.DataGridColumn;
          import mx.formatters.SwitchSymbolFormatter;
          private static var sf:SwitchSymbolFormatter;


          public static function setFormat(dgc:mx.controls.dataGrid-
Classes.DataGridColumn

 formatString:String):void {
              switch (formatString.toLowerCase()) {
                  case "ssn":
                      dgc.labelFunction = ssnLabelFunction;
                  case "phone":
                      dgc.labelFunction = phoneLabelFunction;
              }
```

**Get more with Hot Banana!**

Web Content Management + Active Marketing

Hot Banana software builds efficient, affordable, fast, very easy-to-use and measurable Web sites. And they're ColdFusion-based and optimized for marketing performance. That's what makes us different and better!

**Visit us today at:** hotbanana.com/webinar or call 1-866-296-1803 to schedule a live 1-on-1 demo.

hotbanana®

***"I was totally intimidated by Java, but I knew I had to learn it. Your class taught me what I honestly thought I couldn't be taught."*** **- Sharon T**

**Java for ColdFusion Programmers?**

**Java for ColdFusion Programmers**, the five-day Hal Helms, Inc. training class is designed for ColdFusion programmers; no previous Java experience is needed. You'll learn how to think in objects and program in Java.

**For class information and registration, come to halhelms.com.**

```
        }
        private static function ssnLabelFunction(item:Object, column:
                                                    mx.controls.
dataGridClasses.DataGridColumn):String {
            if (!sf) {
                sf = new SwitchSymbolFormatter();
            }
            return sf.formatValue("###-##-####", item["SS_NUMBER"]);
            }
        private static function phoneLabelFunction(item:Object, col-
umn:DataGridColumn):String {
            if (!sf) {
                sf = new SwitchSymbolFormatter();
            }
            return sf.formatValue("(###)###-####", item[column.
dataField]);
            }

    }
}
```

## Listing 5 A simple application illustrating FormattingManager.as

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical"
  xmlns:fx="http://www.theriabook.com/2006">
  <fx:DataGrid  id="dg" creationComplete="dg.fill()"
     destination="com_theriabook_composition_EmployeeDAO"
method="getEmployees"
  >
     <fx:columns>
        <mx:Array>
           <mx:DataGridColumn    dataField="EMP_LNAME"   />
           <mx:DataGridColumn    dataField="EMP_FNAME"  />
           <fx:DataGridColumn    dataField="PHONE"
formatString="phone" />
           <fx:DataGridColumn    dataField="SS_NUMBER"
formatString="ssn"  />
        </mx:Array>
     </fx:columns>
  </fx:DataGridEx>
</mx:Application>
```

## Listing 6 MaskFormatter.as

```
//MaskFormatter.as

package com.theriabook.formatters {
  import mx.formatters.Formatter;
  import mx.formatters.SwitchSymbolFormatter;

  public class MaskFormatter extends  Formatter {
     private var formatString:String;
     private var sf:SwitchSymbolFormatter;
     public function MaskFormatter( fs:String) {
        formatString = fs;
        sf = new SwitchSymbolFormatter();
     }
     public override function format(val:Object):String {
        return sf.formatValue( formatString, val);
     }
  }
}
```

## Listing 7 FormattingManager.as

```
//com.theriabook.controls.dataGridClasses.FormattingManager.as

package com.theriabook.controls.dataGridClasses
{
  public class FormattingManager
  {
        import mx.controls.dataGridClasses.DataGridColumn;
        import mx.formatters.*;
         import com.theriabook.formatters.MaskFormatter;

         public static function setFormat(
            dgc:mx.controls.dataGridClasses.DataGridColumn,
            formatString:String):void {
            var  formatter:Formatter = null;
            switch (formatString.toLowerCase()) {
               case "ssn":
                  formatter = new MaskFormatter("###-##-####");
                  break;
               case "money":
                  formatter = new CurrencyFormatter();
                  CurrencyFormatter(formatter).precision=2;
                              break;
               case "phone":
                  formatter = new PhoneFormatter();
                  break;
               case "shortdate":
                  formatter = new DateFormatter();
                              break;
               case "zip":
                  formatter = new ZipCodeFormatter();
                              break;
               }
               if (formatter) {
                  dgc.labelFunction = function (
                     item:Object,
                     dgc:mx.controls.dataGridClasses.DataGridCol-
umn):String
                  {
                     return formatter.format(item[dgc.dataField]);
                  }
               }
         }
  }
}
```

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

ZoomFLEX
powered by Shado

## If you're new to Flex, then here's 10 things you probably don't know you don't know.

## Like, how long it will take you to:

1. Be confident your Flex application will scale and handle Enterprise level load?
2. Get to the stage where you can build ColdFusion-Flex applications faster than you can build ColdFusion only applications?
3. Develop file management controls for both the server and client including image gallery and rendering controls?
4. Build data management and database abstraction libraries that work in ColdFusion and Flex?
5. Make sure you can debug your application easily to an application specific level?
6. Be confident you have secured your Flex application and have authentication and authorization working?
7. Find, understand or develop a best practice architecture for ColdFusion-Flex applications?
8. Build a server-side MVC framework and all the related libraries for your ColdFusion server to talk to Flex?
9. Build great looking application templates that can be deployed in minutes?
10. Be confident that your Flex applications will be future proofed?

As a leading Adobe OEM Partner Straker Interactive have been developing ColdFusion Flex applications for over two years - so we think it's fair to consider ourselves in the know. We've used that time and our hands-on experience to create ZoomFlex "the fastest way to build Flex applications". Using ZoomFlex will dramatically speed up your ColdFusion Flex development, drop your learning curve and give you a head start in at least 10 ways so you can get going with Flex – fast!

With ZoomFlex you'll enjoy all the benefits of moving to a richer interface without any of the hassles.

For more information visit **www.zoomflex.com** to download a demo version now.

STRAKER

For more information

a Straker Interactive Limited
Phone (64 9) 309-5252, Fax (64 9) 309-2797
e info@straker.co.nz  w www.straker.co.nz

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

HostMySite.com

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