SYS·CON

CFDJ CONTEST - IT'S ALL ABOUT BEING PRACTICAL PG 7

ColdFusion Developer's Journal

ColdFusionJournal.com

June 2005 Volume:7 Issue:6

ColdFusion Turns 10 Years Old

ColdFusion's 10th Birthday... and the Next 10 Years PG 32

PLUS...

One little box.
A whole lot of power.

Put it to work for you.
The shortest distance between you and powerful web applications.

Full speed ahead. The release of Macromedia ColdFusion MX 7 is changing the whole game. This groundbreaking release introduces new features to help address your daily development challenges. These features include:

› Structured business reporting? Check. Printable web content? Check.

ColdFusion MX 7 provides a structured business reporting solution that will have you creating detailed business reports for anyone who needs them. You can also dynamically transform any web content into high-quality, printable documents with ease. The days of needing a third-party application to generate dynamic reports are going, going, gone.

› Make rapid J2EE development a reality.

So, you're heavily invested in J2EE but would love to complete projects in less time? ColdFusion MX 7 is your answer: It delivers RAD productivity on top of the power and scalability of J2EE and deploys onto standard J2EE server environments.

› New mobile applications are a go.

Innovative new features enable ColdFusion MX 7 applications to reach beyond the web. So you can rapidly create new applications, or extend existing ones, to connect with mobile phones and IM clients using a variety of protocols. The new world of mobile communications is exciting, and this your invitation to the party.

To learn more or take ColdFusion MX 7 for a test drive, go to:
macromedia.com/go/cfmx7_demo

Brad Lawryk: "Before MX Kollection, I 'thought' I was doing a **pretty good job**, and now ...

Everyone tells me I´m doing a **brilliant** job!"

If you own Dreamweaver,

MX Kollection is a must!

List Database Information
Create dynamic lists
Order records within a list
Insert, Edit, Delete field in list
Automatic navigation bars
Automatic list filter
Sort column by clicking on the header
Automatic row counter
Delete multiple records directly from list
Create master/detail lists

Manage Recordsets Visually
Create, Edit recordsets visually
Visually add tables to query
Contextual menu to refresh fields
Large query management Zoom In/Out
JOINS between tables
Define and apply complex conditions
Automatically generate SQL conditions
Add GROUP BY for aggregated fields
Extract information from multiple tables
Optimized SQL generation
Smart SQL queries for list filters
Automated database introspection

Rapid HTML Form Creation
Generate Insert Record Form
Generate Update Record Form
Insert, Edit or Delete form field
Redirect to page after form submit
Table and CSS form generation

Form Validation and Error Handling
Validate form fields
Rich formats library
Allow custom validation formats
Error handling
Preserve submitted values on error

Upload Files and Images
File Upload
Image Upload
Resize and sharpen image on upload
Show Dynamic Image
Download Uploaded File
Delete file from specified folder
Show If File Exists on Server

Send E-mails
Send e-mail after form submit

Online HTML Editor
Edit HTML content visually
Use your own CSS styles
Upload and manage server images
Format tables and align images

Enhanced HTML Form Controls
Date Picker
Dependent Drop-downs
Editable Drop-down (Combo-box)
Masked Textfield
Numeric Textfield
Dynamic Search
n...1 Dependent field
Smart Date

Security & User Authentication
Login form generator
Login with User Levels
Remember me feature
Encrypted password
Restrict access to page
Generate password with fixed length
Update user password
Send password by e-mail

DREAMWEAVER PRODUCTIVITY TOOL FOR DYNAMIC WEBSITE DEVELOPMENT

Download a trial version here - **http://interaktonline.com/go/MXKollection/**

Find out how easy is to use our tools - **save 30%** by using the coupon in the right.

Caution: InterAKT extensions are known to cause a high increase in productivity. For those of you who can't deal with the free time please continue to hand-code or refer to our competitors for help.

$100 coupon - 199920002001

Interakt macromedia approved macromedia ALLIANCE PARTNER

http://www.interaktonline.com/

CFDynamics

Dedicated Server Packages Starting at $189/mo.

All dedicated servers include:

- **FREE STATS SOFTWARE!**
- **No long term commitments!**
- **FREE SQL server access!**
- **FREE MAIL SOFTWARE!**
- **Fair and simple pricing!**
- **Optional server maintenance!**

As one of the premier ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to provide **better service** to ensure your satisfaction. Not only do we offer the **finest in shared hosting plans**, but we now offer the **finest in 100% dedicated server plans**! Now you can afford the freedom of having your own dedicated server!

macromedia **ALLIANCE PARTNER**

Paper Thin Partner

BlueDragon Certified V.I.P. Hosting Partner

When your needs have outgrown shared hosting look to CFDynamics for **total freedom.** With dedicated server packages they're not offering an oxymoron; "virtually private" or "virtually dedicated" is NEITHER private nor dedicated. CFDynamics offers a solution that is **100% completely dedicated**. They don't play games with the fake stuff; CFDynamics only offers the real deal. Real Service. Real Satisfaction. Real Value.

Real Freedom.

Visit us online or call to order!

WWW.CFDYNAMICS.COM

1 - 866 - 233 - 9626
866 - CFDYNAMICS

ColdFusion Developer's Journal

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

editorial

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

technical editor
Raymond Camden raymond@sys-con.com

editor
Nancy Valentine nancy@sys-con.com

associate editor
Seta Papazian seta@sys-con.com

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

production

production consultant
Jim Morgan jim@sys-con.com

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
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com
Andrea Boden andrea@sys-con.com

contributors to this issue
Simon Horwith, Jeff Houser, Mike Nimer, Joe Rinehart, Neil Ross, Jeff Peters, Hal Helms, Andrew Simon, Steven Forehand, Phil Hulsey, Sam Farmer

editorial offices

SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
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
135 Chestnut Ridge Rd., Montvale, NJ 07645

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

For promotional reprints, contact reprint coordinator Kristin Kuhnle, kristin@sys-con.com. SYS-CON Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.

What a long, strange trip it's been. . .

By Simon Horwith

I'm writing this editorial on the exact day that ColdFusion 1.0 was released back in 1995. Looking back over the server's evolution, it's really come a long way from its start as a CGI application (anyone remember DBML?). Though the server has really grown substantially, the things that made it popular back then are still its strongest features: a rapid development environment for creating dynamic, database driven web applications. ColdFusion MX 7 reminds me a lot of the first release of CF 10 years ago – so many powerful new features that make tasks which are difficult in other environments, trivial. I don't doubt that 10 years from now, we will be looking back at CFMX 7 as one of the more pivotal releases in the server's history. Still, it's hard to believe that ColdFusion is already 10 years old. For many of us, the fact that CF is 10 years old is made more difficult to believe because of its implication that we have been doing this for a decade now!

This month marks some other milestones. It's been one year since I assumed my role as Editor in Chief here at CFDJ. I rely heavily on feedback from our readers in order to determine what I'm doing right and wrong, so please keep the emails coming. The feedback so far has indicated that everyone really likes the "focus issue" format, so we'll continue with that until people tell me otherwise. Also, the largest annual conference devoted exclusively to ColdFusion, CFUnited (formerly CFUN) is being held in Rockville, MD at the end of the month. I look forward to meeting many of our readers at the conference. You can expect full coverage of the conference in next month's issue of CFDJ.

Lastly, Macromedia had many Flash and RIA announcements this month. They unveiled the Flash Platform – which in a nutshell is the next generation player along with its accompanying tools and specs. The press release about the flash platform is available online at www.macromedia.com/macromedia/proom/pr/2005/unveiling_flashplatform.html. For more about the technical aspects of the Flash Platform, visit the Flash Platform homepage at www.macromedia.com/platform/. Macromedia also announced that they've joined the Eclipse Foundation as an add-in provider and will be releasing a plug-in (code named Zorn) that allows Flex developers to use eclipse to develop their applications. More on this initiative can be found at www.macromedia.com/macromedia/proom/pr/2005/eclipse_flashplatform.html. This is an exciting month indeed… and in that spirit we've got an exciting issue to accompany it!

To commemorate ColdFusion's birthday, I sat down with Jeremy Allaire for a one-on-one interview about the past, present, and future of ColdFusion. Jeremy is one of the founding fathers of ColdFusion and is one of the true pioneers of the web. It was a great experience discussing his views on ColdFusion and current technology trends. Also in this issue we have many articles focussing on our "deep focus" topic for the month: gathering and presenting data. The majority of our articles are to do with creating forms, reports, and other output using Flash, FlashPaper, and/or PDF as the client-side content format. Flash and PDF generation and the new reporting engine are without a doubt some of the key new features in CFMX 7, and this month's issue is an excellent introduction to the various ways to use all of these new features in your applications. I hope you enjoy reading this month's issue as much as I enjoyed putting it together for you…

About the Author

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal. *Simon is a Macromedia Certified Master Instructor and a member of Team Macromedia. 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

Seapine Software™
changing the world of software development

Complete source code and asset management in Dreamweaver MX—now possible with Surround SCM.

Dreamweaver users know a beautiful Web-based product is only skin deep. Underneath, it's a tangle of hundreds or thousands of ever changing source files. Without a good development process and strong tools, bad things happen. Surround SCM can help.

Surround SCM lets you...

Track multiple versions of your source files and easily compare and merge source code changes.

Check out files for exclusive use or work in private workspaces when collaborating on a team.

Automatically notify team members of changes to source files—push changes through your organization.

View complete audit trails of which files changed, why, and by whom.

Associate source code changes with feature requests, defects or change requests (requires additional purchase of TestTrack Pro).

Remotely access your source code repository from Dreamweaver MX.

Surround SCM adds flexible source code and digital asset control, powerful version control, and secure remote file access to Dreamweaver MX. Whether you are a team of one or one hundred, Surround SCM makes it easier to manage your source files, letting you focus on creating beautiful Web-based products.

Features:

Complete source code and digital asset control with private workspaces, automatic merging, role-based security and more.

IDE integration with Dreamweaver MX, JBuilder, Visual Studio, and other leading Web development tools.

Fast and secure remote access to your source files—work from anywhere.

Advanced branching and email notifications put you in complete control of your process.

External application triggers let you integrate Surround SCM into your Web site and product development processes.

Support for comprehensive issue management with TestTrack Pro—link changes to change requests, bug reports, feature requests and more.

Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Want to learn more?

Visit www.seapine.com/webdev to discover how Surround SCM can save you time and headaches. Be sure to download our white paper, **Change Management and Dreamweaver**, and learn why change management is the new must-have tool for web development.

www.seapine.com/webdev
1-888-683-6456

Surround SCM Source Control
TestTrack Pro Issue Management
QA Wizard Automated Testing

macromedia ALLIANCE PARTNER

software development 13th annual product excellence award

SYS-CON MEDIA

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

vp, business development
Grisha Davida grisha@sys-con.com

group publisher
Jeremy Geelan jeremy@sys-con.com

advertising

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

vp, sales & marketing
Miles Silverman miles@sys-con.com

advertising director
Robyn Forma robyn@sys-con.com

advertising manager
Megan Mussa megan@sys-con.com

sales & marketing director
Dennis Leavey dennis@sys-con.com

associate sales managers
Dorothy Gil dorothy@sys-con.com
Kim Hughes kim@sys-con.com

sys-con events

president, events
Grisha Davida grisha@sys-con.com

national sales manager
Jim Hanchrow jimh@sys-con.com

customer relations

circulation service coordinators
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com

manager, jdj store
Brunilda Staropoli bruni@sys-con.com

sys-con.com

vp, information systems
Robert Diamond robert@sys-con.com

web designers
Stephen Kilmurray stephen@sys-con.com
Percy Yip percy@sys-con.com

online editor
Roger Strukhoff roger@sys-con.com

accounting

financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

subscriptions

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

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

CFDJ Contest – It's All About Being Practical

This month's contest is about building something that we can all use

By Simon Horwith

This month, I am pleased to announce that Kevin Kazmierczak is the winner of the contest to develop a polling/survey application. His submission included a simple set-up and script for creating and populating the database (a straightforward read me file makes installation easy), a slick administrative interface that is robust and easy to use, and a front-end that allows site visitors to enter their response to the current poll(s). His solution includes not only the ability to create questions and define the answers and HTML Form Control answer format, but also the ability to view current poll results in simple chart format (including PDF format) as well.

It's also worth mentioning that his submission was created with Fusebox. I've never been a big fan or user of fusebox, and so I must admit that I was a little sceptical at first. That said, once I got it installed and running, I found it to be full-featured and pleasantly easy to use. I'm still not a fan of fusebox, and there are some non-fusebox related things about the code that I don't care for (like writing to the screen from a CFC), but it was still a terrific submission. Kevin won a free registration for himself and a guest to attend the CFUnited conference in Rockville this June. You can download Kevin's poll application from www.horwith.com/downloads/pollsystem.zip. Congratulations, Kevin!

I want to remind our readers that there is still time to enter last month's contest to build a CF Server administrator's module. I'd also like to mention that if any of our readers work for a company that is willing and interested in donating a prize(s) for this column, please let me know. Now for this month's contest.

This month, I thought I'd challenge our readers to develop something practical that others will find very useful. The contest is to develop a shopping cart. I'm not going to specify a lot of requirements – just know that for a challenge like this, the ability to integrate the cart with existing applications is the number one priority. Submissions do not have to contain any code for credit card authentication, nor do they have to include a database schema. I do expect submissions to include installation/integration instructions. I also expect clear instructions as to where in the code (and how, if applicable) to customize the cart for a person's specific database and/or chosen credit card authentication system.

The prize this month is three licenses of the soon to be released MX Kollection version 3 from InterAKT (http://www.interaktonline.com). If you haven't used it before, the MX Kollection is a very nice extension of the existing functionality in Dreamweaver MX. You can read more about MX Kollection 3 in the product review in this month's issue of CFDJ.

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

Filling out PDF Forms in ColdFusion

How to use the cfdocument tag and some Dreamweaver magic

By Jeff Houser

A few years back, when CF5 was still in the beta stages, a client wanted to be able to fill out PDF forms through their web browser and save, and print, the filled out PDF form.

After experimenting with a few free methods, such as the use of FDF files, I stumbled upon the ActivePDF toolkit. ActivePDF provides a series of tools for PDF manipulation. The toolkit is a COM object that can be used to fill out PDF forms. It did the job wonderfully. I spent more time creating the PDF form with Acrobat than I did creating the ColdFusion markup to allow people to fill out the form via the web.

Now we can jump forward to more recent times, I recently had the opportunity to revisit the project. ColdFusion MX does not deal with COM as well as CF5 did, so we decided to look into other options. CFMX7 has some built in PDF capabilities, so investigating them seemed like an obvious choice. Although CFMX7 does not deal with PDF forms, there is an alternative method that worked well for our purposes. We used the cfdocument tag and some Dreamweaver magic to implement our solution. I'm going to explain that method to you today.

Prepping the Form

You can do this with any PDF form. For example purposes, I'm going to use the US copyright form PA, downloadable from there web site, here: http://www.copyright.gov/forms/formpa.pdf. You'll want to extract the PDF as an image. I was able to open it in Adobe Photoshop and save each page as an 8.5 x 11 image. If you don't have access to Photoshop, there are also various low cost converter programs that will convert a PDF's contents to an image. Two such programs are ReaCoverter pro from ReaSoft, or VeryPDF's PDF to Image Converter. They are located at http://www.reasoft.com/products/reaconverter/ and http://www.verypdf.com/pdf2tif/index.htm respectively.

Now you have your form as an image. What next? I know this will be a tough move for many of you hard-core developers out there, but next we are going to use Dreamweaver in design mode. These are the steps:

1. Create a new page. I named mine "formip.cfm" for a "form input processing" page.
2. Expand the property panel if it is not already expanded. If it is not displayed, you can display it using "Properties" from the window menu. Click the "Page Properties…" button. If you don't see this button in the property panel, click on the body tag and it should show up.
3. You should see a window like the one shown in Figure 1. For the category select appearance. For the background image, click browse and select the form that you
4. Click OK to save your changes. Now move to the design view by clicking the design button. You should see your form in the background of the page. Depending on the size of your screen, it will probably tile but for our purposes it doesn't matter.
5. Display the insert bar in Dreamweaver if it is not already displayed. You can do this from the select "insert" from the window menu. Click the layout tab of the insert bar. The insert bar is shown in Figure 3. Roll your mouse over the third button and it should say "Draw Layer". This will allow you to draw a layer anywhere on the screen. Behind the scenes it will create a div layer, positioned using absolute position. Click this button.
6. You will now be able to draw the layer anywhere on the page just by clicking and dragging. Create a layer for each line of the form that you want to fill out.
7. Adjust the layer properties to give it a background color that is equal to the background color of your form. This will probably be white. Each layer will contain one form field, surrounded by pound signs, so that ColdFusion knows to process it. For the title of this work field, I would name it "#form.titleofwork#. You can add these specifics in either the code view of the design view, but it may be easier in the design view to tell which layer goes to which line of the form.

Here's a sample of the finished template. For brevity, I only included two fields in the sample template (Title of Work, and Previous Title):

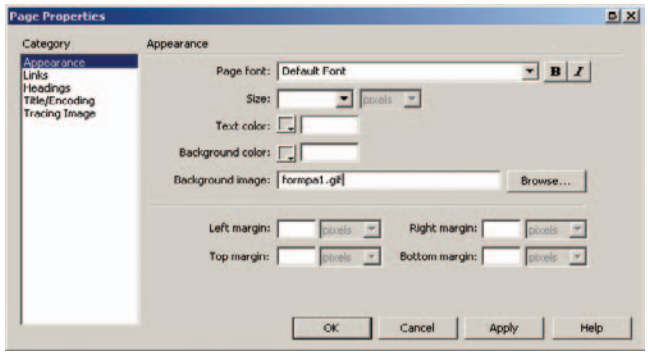


Figure 1: The Page Properties Dialog

macromedia

There are workarounds for a mediocre intranet, but they still waste time and money.

Meet the Macromedia Web Publishing System. Unlike the average CMS, this works. By allowing business users to publish content, IT bottlenecks are eliminated. Users can simply point and click to update specific pages or sections–while other areas remain protected. So everyone in your organization can share critical information quickly and easily, and you have an intranet that actually lives up to its potential.

Learn how to make your intranet work for you, not the other way around:
macromedia.com/go/webupdate

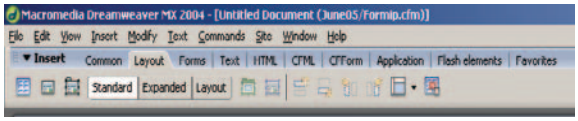


Figure 2: Menu

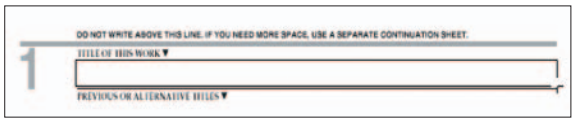


Figure 3: Insert Bar

```
<cfdocument format="PDF">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/
html; charset=iso-8859-1">
<title>Copyright Form PA PDF </title>
<style type="text/css">
<!--
 body {
 background-image: url(formpa1.gif);
 }
-->
</style>

</head>

<cfoutput>
 <body>
 <div id="Layer1" style="position:absolute;
left:86px; top:199px; width:502px; height:27px;
border:1px none ##000000; z-index:1">
  #form.TitleOfWork#
 </div>
 <div id="Layer2" style="position:absolute;
left:86px; top:243px; width:503px; height:22px;
border:1px none ##000000; z-index:2">
  #form.PreviousOrAlternate#
 </div>
</body>
</html>
</cfoutput>
</cfdocument>
```

Using the cfdocument tag

Create a new page in Dreamweaver. This one can be done all in code view if you want. I named this page, formi.cfm (for "form input"). The code is as follows:

```
<form action="formip.cfm" method="post">
 Title Of Work: <in-
put type="text"
name="TitleOfWork"><Br>
 Previous Or Alternate: <input
type="text" name="PreviousOrAlte
rnate"><Br>
 <input type="submit">
</form>
```

It contains an HTML form with two input boxes, one for the Title of the Work and one for the Previous or Alternate title. The form posts onto the formip.cfm page that you created in the previous section of this article. You probably aren't finding anything new to you in this template.

Switch back to your "formip.cfm" template. You'll notice that the whole generated html is wrapped in a cfdocument tag. This tag tells ColdFusion to generate a PDF or FlashPaper document. Here are some of the attributes to the cfdocument tag:

- ***Format:*** Format is a required attribute. The two accepted values for this are FlashPaper and PDF, depending on the type of document you want to create. In our example, we set it to PDF.
- ***FileName:*** The filename attribute is an optional attribute. If you specify a filename then ColdFusion will generate your PDF or Flashpaper document and save it as a file. If you do not specify a filename then the resulting file will be streamed to the browser. In this example, the code streamed the resulting PDF to the browser.
- ***Overwrite:*** The overwrite attribute is an optional attribute that is used in conjunction with the filename attribute. It accepts a Boolean value, and if set to yes then the file will be overwritten if it already exists. If set to no, then an error will be thrown if the file exists.
- ***Fontembed:*** FontEmbed is an optional attribute. It is a Boolean attribute, so accepts either yes or no as values. If set to yes, then all fonts are embedded in the finished document. If set to no, then fonts are not embedded.
- ***BackgroundVisible:*** The backgroundvisible attribute is an optional attribute that specifies whether the background prints when the user prints the document.
- ***Name:*** The name attribute is an optional attribute. If you want to store the PDF or FlashPaper document as a variable within ColdFusion then you'll use the name attribute. In this example, you were streaming the results to the browser and this was not used.

For a complete list of all attributes supported by the cfdocument tag, you should review the documentation located at http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000236.htm#3765835. It offers many additional formatting options that go beyond what was discussed here. You might also want to investigate the subtags to cfdocument: cfdocumentitem and cfdocumentsection. Cfdocumentitem can be used to put page breaks, headers, or footers in your documents. Documentation on the tag is located here http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000237.htm#3766460. cfdocumentsection is used in conjunction with cfdocumentitem to give different header, footer, or page numbers to different sections of the finished document. More information on this tag is found here http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000238.htm#3766659.

Wrap Up

I recently took over management of the Hartford Connecticut Macromedia User Group. The idea for this usage of cfdocument came out of a demo Ben Forta gave when he spoke at our group a few months ago. Although it doesn't make use of PDF forms, it certainly emulates the functionality available in them, and depending on the quality of your PDF image extracts should be suitable for most applications.

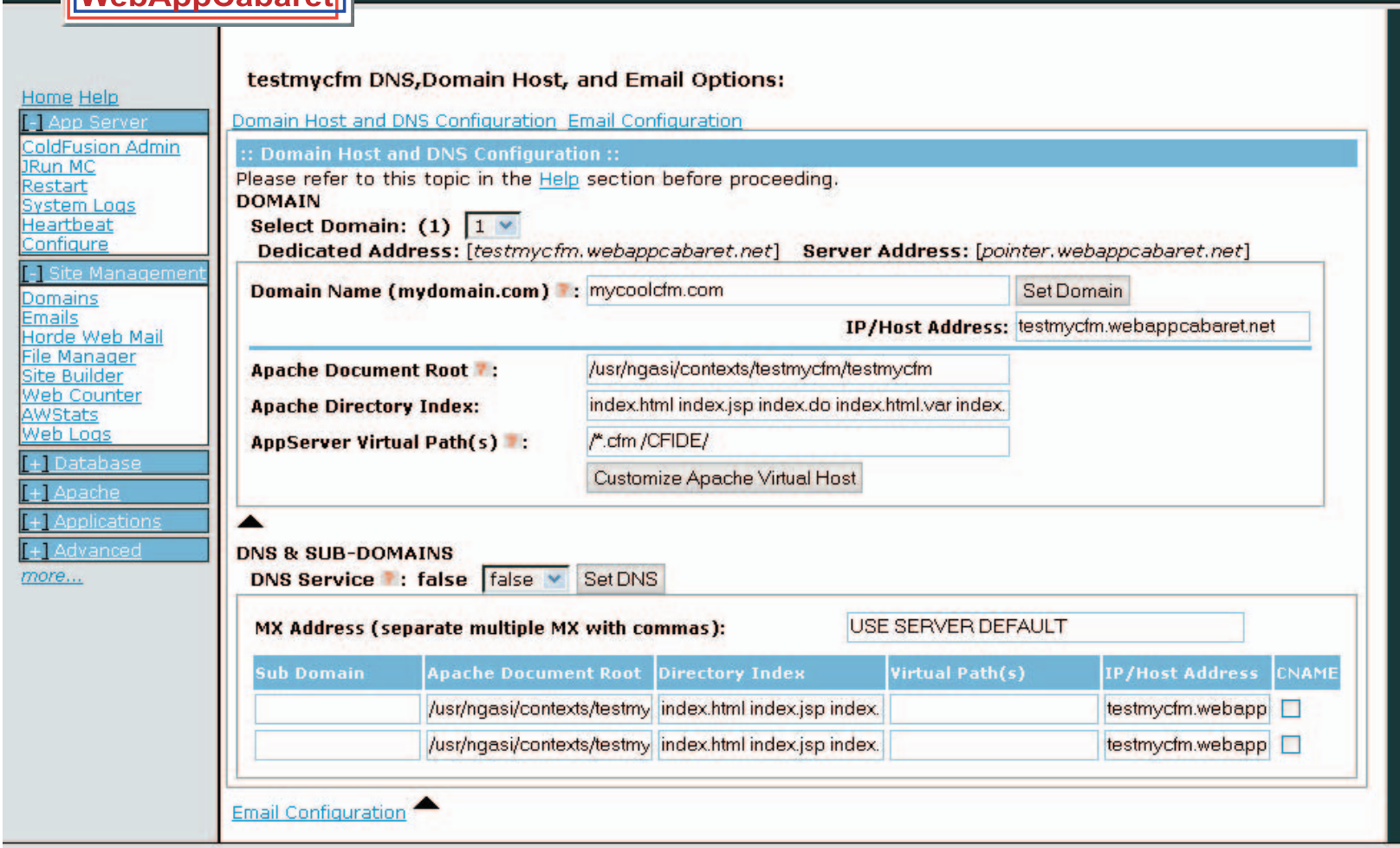
About the Author

Jeff 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

WebAppCabaret sm

ColdFusion MX 6.1 Enterprise Hosting made easy.



Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with ColdFusion MX. At WebAppCabaret our standards based process and tools make deploying ColdFusion MX applications as easy as a point-and-click. We call it **Point-and-Deploy Hosting**. Our advanced NGASI Web Hosting management Control was designed for the hosting and management of ColdFusion web sites and applications thus cutting down on maintenance time.

Backed by an experienced staff as well as a **Tier 1 Data center** and network. Our network is certified with frequent security audits by reputable security firms.

All ColdFusion hosting plans have separate and individual installation AND instances of ColdFusion MX 6.1 Enterprise with **full access to ColdFusion Administrator** and JRun Management Console; so there is virtually no restriction or customization required for your application.

Complete power and control at the tip of your fingers. We take care of the system and hosting infrastructure so you can concentrate on development and deployment of your application. That is **real ROI**.

Log on now at http://www.webappcabaret.com/cdj.jsp or call today at 1.866.256.7973

NGASI POWERED

WebAppCabaret sm

Point-and-Deploy J2EE Hosting

http://www.webappcabaret.com/cdj.jsp 1.866.256.7973

THE ASYNCHRONOUS CFML GATEWAY

A Real-World Example of a Great New Technology

By Matt Woodward

One of the most powerful new features available in ColdFusion MX 7 Enterprise that many ColdFusion developers might not yet be using is event gateways. Event gateways open up entirely new possibilities for ColdFusion and allow our ColdFusion applications to communicate with more or less any other Internet-enabled system even if the system doesn't communicate via the "traditional" HTTP protocol.

CFMX 7 Enterprise ships with a few gateways to get you started, such as an SMS gateway for communicating with mobile devices, a directory watcher gateway, a socket gateway, a Java Message Service (JMS) gateway, and Jabber/XMMP as well as Lotus Sametime IM gateways, so you have a lot of possibilities right out of the box.

If you don't need the specific functionality offered by these gateways you might not have investigated them thoroughly and because the event gateways themselves are written in Java, creating new gateways may not be a high-priority item for many ColdFusion developers. But there is another gateway that ships with CFMX 7 Enterprise that ColdFusion developers can take advantage of right away, which offers numerous benefits in your ColdFusion applications. In this article I'd like to introduce you to the Asynchronous CFML Gateway, describe some of the great potential uses of the gateway, and show how I added an asynchronous CFML gateway to vastly improve the performance and functionality in an existing ColdFusion application.

The Asynchronous CFML Gateway

The purpose of the asynchronous CFML gateway is to allow developers to offload long-running processes to the gateway. The gateway immediately passes control back to the application and does it's processing behind the scenes. This means that your users will no longer have to wait for lengthy batch processes to complete before doing other things within the application. This is great for things like logging, which if you log a lot of data can really bog down your application since the user has to wait for the logging process to complete on each request.

Even if you have a process that runs behind the scenes that is initiated by the ColdFusion scheduler as opposed to by a user action, leveraging the asynchronous gateway can allow for the process to complete more quickly. For example, if you have a loop of some sort and the work done upon each iteration is fairly lengthy, by sending a message to the asynchronous gateway on each loop iteration the overall process doesn't need to wait for each iteration to complete before proceeding.

What allows this magic to be possible is the use of ColdFusion Components (CFCs) that operate within the event gateway environment as opposed to being tied to a specific request. By creating a CFC that contains a few specific methods (actually we'll concern ourselves with one method in particular – more on this in a moment) and registering the CFC as an event gateway instance in the ColdFusion administrator, the CFC can now operate independently of the traditional request/response model of web applications.

Bear in mind, however, that not all processes are suitable for use with the asynchronous event gateway. One big caveat here is that because the asynchronous process is detached from the request/response cycle, it isn't really feasible to provide feedback to the user in a traditional way such as sending the user to a confirmation page when a process completes. Also, if your application contains processes that occur sequentially and later processes rely

on the results of preceding processes, then even if one of these processes is time-consuming this isn't a case in which use of the asynchronous gateway is possible. When you use the asynchronous gateway you essentially throw the process over the fence and give up control over when it completes and the ability to rely on the results of the asynchronous process.

The asynchronous CFC can however, do things such as log information to a file or database, send information via e-mail, or even send an instant message using the IM gateway, so by no means does the asynchronous CFC's processing disappear into a black hole.

In addition to these methods the event gateways can of course also use the cflog tag, which outputs information to a log that can be viewed from the ColdFusion administrator or directly in the log file itself. In the ColdFusion administrator's Debugging & Logging section you will also see an Event Gateway log that can provide information about what occurs inside your gateways. Finally, for some additional thoughts and methods by which to get feedback from asynchronous CFCs, please see Sean Corfield's blog entries about his Concurrency library under "Resources" at the end of this article.

If you haven't considered the power of asynchronous processing before, I'd be willing to bet the wheels are spinning and your head is filling with ideas of how you can leverage this power in your applications. Let's take a look at one real-world example that created dramatic improvements and was surprisingly easy to build.

Web-Based "E-Mail Blaster"

A few years ago I wrote a ColdFusion application for my company's marketing department that is used for generating and sending e-mail marketing campaigns. This application became affectionately known as the "E-Mail Blaster." Prior to the existence of the E-Mail Blaster our marketing team was literally cutting and pasting e-mail addresses from an Excel spreadsheet into the Lotus Notes e-mail client. Clearly not the most efficient way to do things given the fact that some of our larger batches of e-mail can top 50,000 recipients, and Notes has an upper limit of a few hundred recipients for each e-mail. (Before you declare me the Spam King of Texas, bear in mind that that our lists are entirely opt-in!)

When I saw how the marketing department was dealing with these e-mail campaigns I immediately knew that this was a perfect job for ColdFusion. ColdFusion Enterprise handles large volumes of e-mail extremely well, and it was quite simple to build a web-based interface for creating e-mail campaigns that saves countless hours of time versus the old cut-and-paste method. We use another tool to maintain contact lists, so the marketing team extracts the recipient list into an Excel spreadsheet. They then use the ColdFusion application to upload the Excel spreadsheet containing the recipients' e-mail addresses along with an HTML file containing the contents of the e-mail, and the E-mail Blaster handles the rest.

Hurry Up and Wait

The application was an immediate hit with the marketing department, but there was one drawback. Even as quickly as ColdFusion handles e-mail it still takes a bit of time to process 50,000 e-mails, and in the traditional request/response model of web applications this means the user of the application has to wait … and wait … and wait for the process to complete. This doesn't necessarily mean the user can't go do something else while it's processing, but people tend to get used to submitting a form and seeing a response relatively quickly so by force of habit the users of the E-mail Blaster tend to wait for a response from the application.

In a long-running process such as this the response can be substantially delayed, and this can cause problems ranging from simple annoyance to users resubmitting e-mail batches because they assume the process stopped or an error occurred. In the original version of the application there were problems with Internet Explorer refreshing the processing page automatically because I wasn't flushing anything to the screen during the send process. Once I discovered this issue I updated the application to flush each e-mail address to the screen as the e-mail was sent, which gave the user primitive feedback and also solved the auto-refresh issue with Internet Explorer.

This was still a less-than-ideal situation. I hope this doesn't come as a shock to any of you, but in case you were unaware, users can be impatient at times. No matter how many different ways I explained the situation, because most users are accustomed to clicking on a link and having the next page finish processing more or less instantly, some users just couldn't understand why generating 50,000 e-mails was taking so long.

Even with the addition of flushing each address to the screen, there were still occasional timeout issues that would cause the browser to throw an error of one kind or another. At a minimum seeing an error at the end of the batch would create a sense of uncertainty in the user's mind as to whether or not the e-mails were sent successfully. I thought about going to some relatively insane lengths to change this situation, but in the end I just stuck with the solution and dealt with continually educating my users on the process.

Asynchronous Processing to the Rescue

My wish for a better way of handling things in the E-Mail Blaster was answered with CFMX 7 Enterprise and the asynchronous CFML gateway. No longer do my users have to wait for each e-mail to be sent. Now I can just generate the recipient list, pass everything off to my e-mailer CFC via the asynchronous gateway, and immediately return a message back to the user informing them that their e-mail batch is in process and they will be notified via e-mail (and Lotus Sametime IM as well, but that's a story for another article!) when the process is complete. I also log each e-mail address to a text file as the e-mails are generated and attach this text log to the confirmation e-mail, so no longer do I have to respond to questions concerning whether or not a particular e-mail was sent to a particular address.

This is an amazingly powerful solution, and as usual ColdFusion makes doing powerful things such as this extremely simple. Let's walk through the steps involved with creating a CFC and registering it in the ColdFusion administrator so it can be used via the asynchronous gateway. (Just a reminder: event gateways are only available in ColdFusion Enterprise and Developer editions. Gateways are not available in ColdFusion Standard.)

Step One: Create AsynchE-mailer.cfc

This step was surprisingly easy. Since I originally wrote the E-mail Blaster when CFMX was first released I wasn't using CFCs very heavily at that point, so my e-mail sending code was in a simple CFML page. The actual code itself did its job well, however, so basically it was just a matter of some minor tweaks and pasting the CFML code into a new CFC.

In order to be able to use a CFC via the asynchronous gate-

way there is one key method that must exist in the CFC: onIncomingMessage. This function is what is called when SendGatewayMessage() is called from your ColdFusion application (more on this in a moment). The onIncomingMessage method in the CFC takes a struct as an argument, and this struct contains the information you want to use within the asynchronous CFC. In the case of the E-Mail Blaster, the struct contains the recipient list, the HTML contents of the e-mail, and a few other bits of information such as the administrator and sender's e-mail address and a path to which a log file is written. See Listing 1 for the complete AsynchE-mailer.cfc code.

Step Two: Register AsynchE-mailer.cfc in the Event Gateway Administrator

Once the CFC is created, registering it for use via the asynchronous gateway is a snap. First, log into the ColdFusion administrator and click on the Event Gateways link on the left. The Gateway Types area is where you define the various gateways to which the ColdFusion server has access, and you'll see that one of these is the asynchronous CFML gateway. You don't need to do any configuration on this page in the administrator, just be aware that this is where the asynchronous CFML gateway is registered with the ColdFusion server.

Click on Gateway Instances to be taken to the form you will use to configure the AsychMailer.cfc to run as a gateway. You'll see the following fields:

- Gateway ID: This is the handle or name by which you will refer to the gateway instance in your ColdFusion code, and can more or less be anything you want. I named this gateway "Asynch E-mailer."
- Gateway Type: This lists the gateway types that are registered on the server. In this case you'll want to choose "CFML – Asynchronous Events via CFML."
- CFC Path: This is the physical path on the server to the CFC that you want to use as the gateway.
- Configuration File: This tells ColdFusion where the configuration file for the gateway is, but we don't need one for this CFC so we leave that field blank.

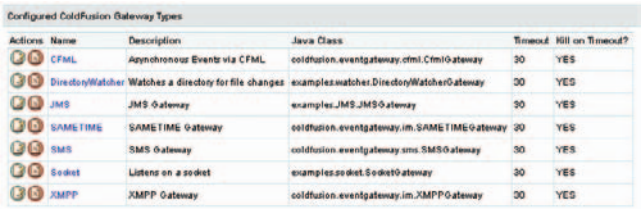


Figure 1: Gateway Types Included with CFMX 7 Enterprise

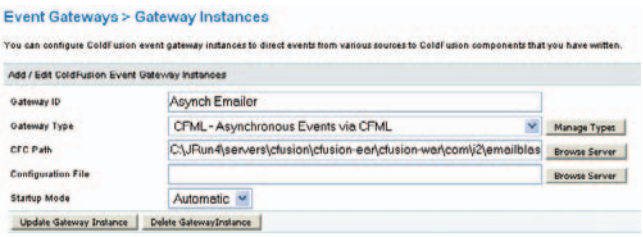


Figure 2: The Gateway Instances Form

- Startup Mode: This tells the ColdFusion server whether you want this gateway instance to start up automatically (i.e. when ColdFusion starts), manually, or that the gateway is disabled altogether. If it's a gateway you're going to use in a production environment, obviously you'll want it to start automatically.

Once you have the information entered, click on "Add Gateway Instance" and after it the gateway instance is registered and starts up it's available to use from your application. With the setup work out of the way, let's take a look at the specifics of how we send our e-mail blasts asynchronously.

Step Three: Call the Asynch E-mailer Gateway from the E-Mail Blaster Application

Calling the asynchronous gateway is as simple as calling the SendGatewayMessage() function, which takes two arguments: the handle or name we gave our CFC in the ColdFusion administrator, and a struct containing the data that we want to pass into the gateway.

First let's build a struct to hold our data. Since we're using the gateway to send e-mails, the struct we pass to the gateway contains the recipient list, the HTML contents, the sender information such as name and e-mail, and a few other items used for housekeeping and error handling. This is just a plain old ColdFusion struct, so simply populate the struct in your CFML or CFC and pass it to the gateway. Since I've converted the original application into a more object-oriented model, I'm using an E-mailBlast Session bean to store the e-mail blast data prior to beginning the send process, so the code populating this struct looks like this:

```
e-mailBlastData.recipients = Session.e-mailBlast.getRecipientList();
e-mailBlastData.fromName = Session.e-mailBlast.getFromName();
e-mailBlastData.fromE-mail = Session.e-mailBlast.getFromE-mail();
e-mailBlastData.subject = Session.e-mailBlast.getSubject();
e-mailBlastData.content = Session.e-mailBlast.getContent();
```

Once the struct is populated, we simply make a call to the asynchronous gateway by using the SendGatewayMessage() function:

```
status = SendGatewayMessage("Asynch E-mailer", e-mailBlastData);
```

Taking another look at the AsynchE-mailer.cfc code, you will see that the gateway's onIncomingMessage function takes in a struct argument called CFEvent. You can call this anything, but CFEvent is the naming convention used in the ColdFusion documentation so I retained it here. Please note that once the struct is in the gateway, rather than calling the struct's members directly as you would in your application (e.g. e-mailBlastData.content) the struct members are called using CFEvent.Data.myStructMember notation. Using the previous example, the content data would come into the gateway as CFEvent.Data.content. Just be aware of the addition of "Data" to the struct.

Although the onIncomingMessage function within the CFC doesn't return anything, the SendGatewayMessage() function returns a boolean. The gateway itself is limited in providing feedback because of its asynchronous nature, so the status boolean

that is returned is really an indication of whether or not the call to the gateway was successful; it does not indicate whether or not the gateway's process itself was successful. The success or failure of the process inside the gateway itself must be retrieved using one of the methods previously outlined.

Now here's where the asynchronous magic happens. Once you call SendGatewayMessage(), processing within your application doesn't stop and wait for a response. The application immediately proceeds past that point in the code, and no matter how lengthy the gateway process is, your application is free to proceed. For lengthy processes that would previously hang up your application this is just what the doctor ordered!

Conclusion

Event gateways open ColdFusion applications up to a whole new world of possibilities. The asynchronous CFML gateway is simple, elegant, tremendously powerful, and surprisingly simple to use. If you have long-running processes in your ColdFusion applications or want to add functionality to your application that you were afraid to attempt before due to potential issues with the traditional request/response model, fear no longer. Write a CFC to handle the processing, register it in the ColdFusion administrator, and take advantage of the power of asynchronous processing.

Resources

Corfield, Sean (2005, April 16). "Asynchronous CFML – Concurrency Library." http://corfield.org/blog/index.cfm?do=blog.entry&entry=4D121277-A559-411D-A0960C66A7E3BFCD

Corfield, Sean (2005, April 14). "Debugging and Caching in Event Gateways." http://corfield.org/blog/index.cfm?do=blog.entry&entry=3F160E0C-B987-C466-41E27E56503E3044

Forta, Ben (2005, March 16). "Understanding Asynchronous Processing." http://www.forta.com/blog/index.cfm?mode=e&entry=1541

Jordahl, Tom (2005, March 16). "Taking Advantage of ColdFusion MX 7 Event Gateways." Recorded Macrochat. http://www.macromedia.com/devnet/mx/coldfusion/articles/gateway_macrochat.html

About the Author

Matt Woodward is a web application developer for i2 Technologies in Dallas, Texas, and also works as a consultant through his company Sixth Floor Software. He is a Macromedia Certified ColdFusion Developer, a member of Team Macromedia, and has been using ColdFusion since 1996. In addition to his ColdFusion work Matt also develops in Flex, Java, and PHP.

mpwoodward@mac.com

Listing 1: AsynchE-mailer.cfc

```
<cfcomponent displayname="AsynchE-mailer" output="false" hint="Asynchronous
        CMFL gateway for sending of e-mails from the E-Mail Blaster">
    <cffunction name="onIncomingMessage" output="false">
        <cfargument name="CFEvent" type="struct" required="true" />

        <cfset logFileName = DateFormat(CFEvent.Data.sendTime, "YYYYMMDD")
                & "_" & TimeFormat(CFEvent.Data.sendTime, "HHmmss") & ".txt"
/>
        <cffile action="write" file="#CFEvent.Data.logFilePath##logFileName#"
                output="E-MAIL LOG FOR JOB RUN AT #DateFormat(CFEvent.Data.
sendTime)# #TimeFormat(CFEvent.Data.sendTime)#"
                addnewline="yes" />

        <!--- get start tick count so we can calculate total time --->
        <cfset startTick = GetTickCount() />

        <cftry>
            <cfloop index="i" from="1" to="#ArrayLen(CFEvent.Data.
               recipients)#"
                    step="1">
                <cftry>
                    <cfmail from="#CFEvent.Data.fromName# <#CFEvent.Data.
                      fromE-mail#>"
                            replyto="#CFEvent.Data.replyToE-mail#"
                            to="#CFEvent.Data.recipients[i]#"
                            subject="#CFEvent.Data.subject#"
                            type="html">
                        #CFEvent.Data.content#
                    </cfmail>
                    <cfcatch type="any">
                        <!--- just proceed on --->
                    </cfcatch>
                </cftry>

                <!--- add line to log file so if this bombs, we know what
                   send attempts
                        were already made --->
                <cffile action="append" file="#CFEvent.Data.logFilePath##log
                   FileName#"
```

```
                        output="#CFEvent.Data.recipients[i]#"
addnewline="yes" />
            </cfloop>

            <!--- calculate run time --->
            <cfset totalTime = (GetTickCount() - startTick) / 1000 />

            <!--- trap any errors --->
            <cfcatch type="any">
                <cfmail from="#CFEvent.Data.adminE-mail#"
                        to="#CFEvent.Data.adminE-mail#"
                        cc="#CFEvent.Data.userE-mail#"
                        subject="E-Mail Send Error">
An error occured during an e-mail blast send process.  The system admin-
istrator has also been notified.  The error details if available are as
follows:
#CFCATCH.Detail#

Please wait for the administrator to address the error and respond to you
before using the system again.
                </cfmail>
            </cfcatch>
        </cftry>

       <!--- send e-mail to user confirming the completion of the job --->
       <cfmail from="#CFEvent.Data.adminE-mail#" to="#CFEvent.Data.user
        E-mail#"
                subject="E-Mail Blast Process Complete"
                mimeattach="#CFEvent.Data.logFilePath##logFileName#">
Your e-mail blast was sent successfully.  The total time for the blast
was #totalTime# seconds.

Attached is a log file containing a list of all the e-mail addresses to
which the e-mail was successfully sent.  You may also view these log files
from the i2 Central administration tools.
        </cfmail>
    </cffunction>
</cfcomponent>
```

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

MX Kollection 3 by InterAKT Online

Some User Perspectives

By Nancy Gill, Joyce Evans, Barbara O'Neal and Steven Redden

Web development is a changing industry. Technologies are born, thrive, and then die, while web developers experience a great stress helping their clients get an Internet presence. Nowadays, the buzz word in web development is "dynamic." Not "dynamic" as in DHTML, but as in "using a database."

MX Kollection is designed for HTML designers that want to take a plunge into database-driven development and here four of the product's beta testers report briefly on how they feel about the product.

CSS Skins

California developer and Team Macromedia member Nancy Gill reports that it's not the many functions of MX Kollection that are her favorite as much as the simple addition of skinning the interface by site.

"I can choose Aqua for the tool's look in Site A and go on to define Site B to show the Arktic skin," she writes. "The advantage in that is that I have so many similar folders from site to site that I won't get confused about where I'm working today. It helps me sort my work flow like never before."

"The ability to fine-tune my settings on a site to site basis," Gill continues, "gives me the kind of control that I have never seen before. I can change the date format, the language for the labels and warnings and other parameters, such as mail server and such with extreme ease."

Barbara O'Neal singles out the Query Builder for praise: "I can build complex queries I would have to use Enterprise Manager to build otherwise. And this means I don't have to leave Dreamweaver MX 2004 to do it ... It's all right here." A bonus is the ability to create a query and click on the Create CFC button to store the query in an independent document, making reuse of the same functionality instantaneous, something ColdFusion gurus have come to love in their chosen server model.

Moving along to the many wizards that power the MX Kollection 3, Nancy Gill reckons that while some may have issues with feeling a wizard "dumbs them down", she thinks this approach is great for getting up to speed with the tool. "Click, click in answer to the questions and it's all created for you," Gill says. Making it easy to work with and easier to get the job done faster, she adds.

Building CMS Web sites

Steve Redden, Technical Director of 3-wise Internet Solutions (http://www.3-wise.com/), reported on MX Kollektion as follows.

"Time and time again, we demo Content Management Systems and web-based applications to potential clients only to see looks of wonder on their faces when MX Kollection 3 allows me conjure up lists that can filter, sort and re-order at the click of a mouse. NeXTensio has been our bread and butter application for the past three years in the construction of lists and create, modify and delete transactions in my developments."

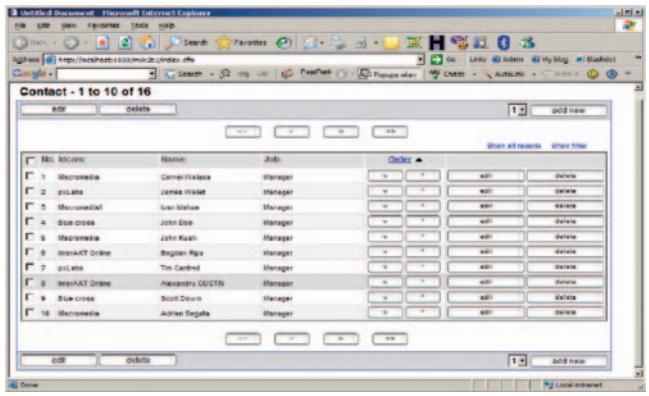


Figure 1: Contact View

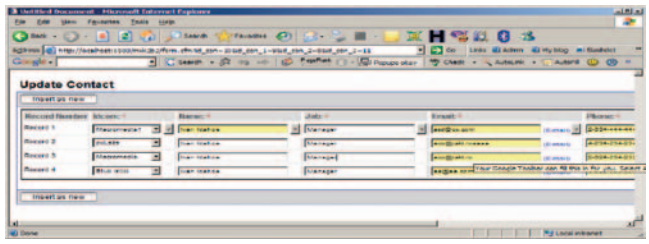


Figure 2: Update Contact

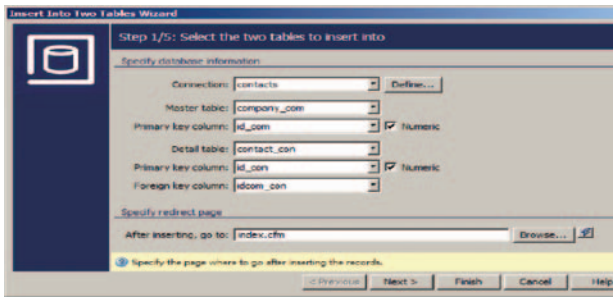


Figure 3: Two Tables Wizard

Redden's clients are delighted to find positive answers to most of their questions and requirements, he says.

Editing Multiple Records at Once

There are wizards in MX Kollection for everything from insert, update and delete forms, to the neat displaying of data. The wizards allow users to build pages for inserting and editing multiple records at the same time.

Barbara O'Neal says her favorite is the "Insert into Two Tables" wizard since this was an operation in ColdFusion or any other server model that required the functionality to be hand coded in the past. Specify your tables, bind the appropriate fields and it's all created for you like magic," she notes.

Form Validation

As for form validation, "This is the best I've seen… it's so complete," says O'Neal. "There is point and click validation for everything I can imagine." Barbara adds that her business partner is a stickler for details and in looking over her site created with MX Kollection validation, he couldn't find anything wrong with it. "That's nothing short of a miracle" she gushed. "Nothing gets by him, believe me."

Documentation and Support

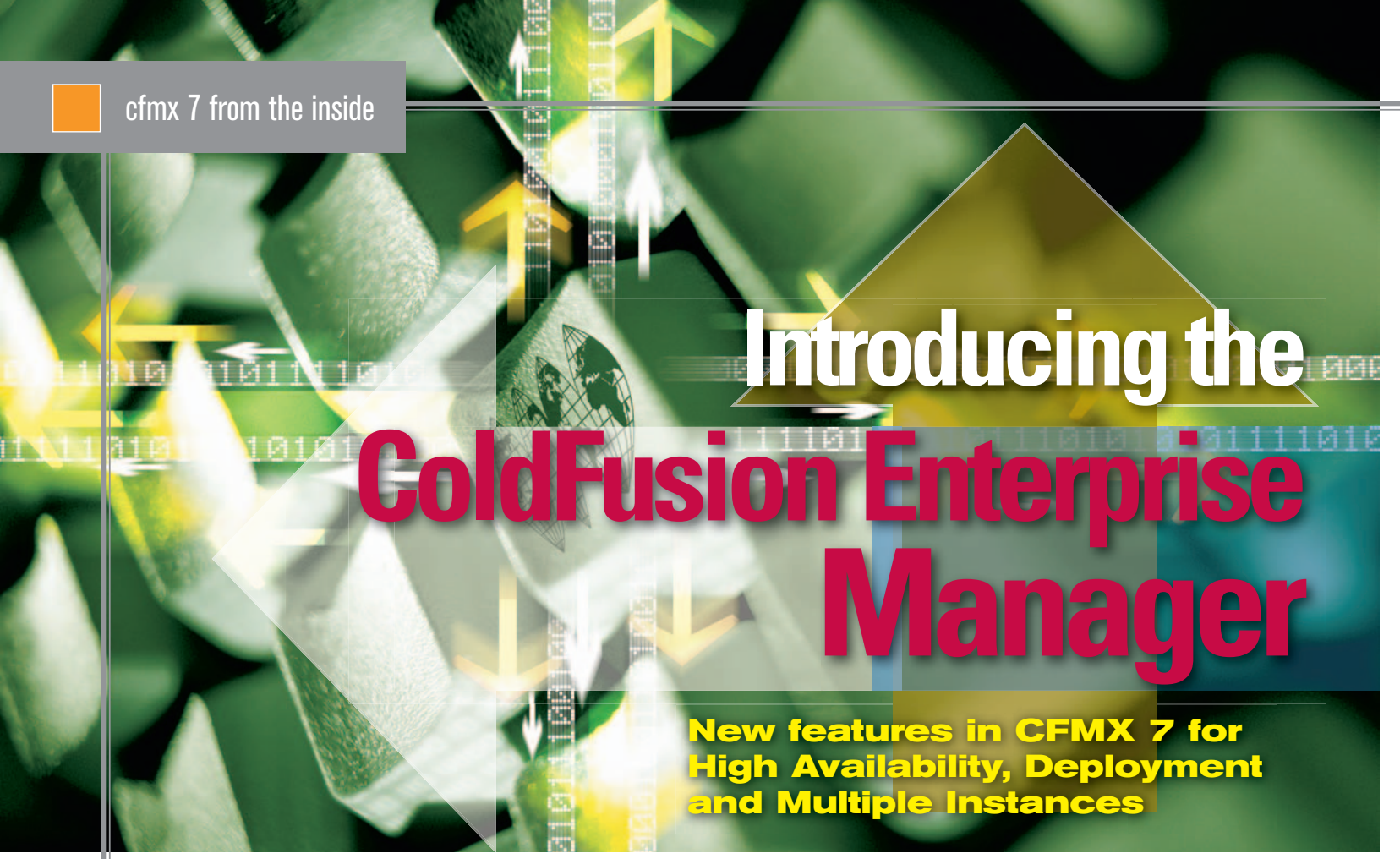
"I'm not only a Web Designer but I teach new designers and developers so I always look for tools to make their jobs easier," said noted Dreamweaver and Fireworks author and instructor Joyce Evans, adding that what she likes about the MX Kollection 3 is how the wizards "really help the beginner developer get up to speed quickly and produce complex applications in no time at all."

"With the Kollection 3, you can easily use ASP, PHP or ColdFusion, even if you're not a programmer at all," Evans notes. "My clients think I'm a genius, of course it's InterAKT that are the geniuses, but we won't tell!" she jokes.
Evans welcomed the tutorials that ship with the MX Kollection 3, calling them an "invaluable addition to any developer's toolbox."

To read more on MX Kollection, ***CFDJ*** readers can go to: http://www.interaktonline.com/

About the Authors

Nancy Gill, Joyce Evans, Barbara O'Neal and Steven Redden
mxdj@sys-con.com



Introducing the ColdFusion Enterprise Manager

New features in CFMX 7 for High Availability, Deployment and Multiple Instances

By Tim Buntel & Geoff Green

Wouldn't it be nice to get all the high-availability and scalability benefits of a cluster of multiple servers without the need to purchase multiple machines? How about being able to manage a number of isolated, performance tuned applications on one piece of hardware instead of taking up many machines in your server room?

ColdFusion MX 7 Enterprise Edition introduces the new Enterprise Manager, which will do those things and more. It makes load balancing, application isolation, and deployment management as easy as creating a datasource in the ColdFusion Administrator. In this article, we'll introduce you to the Enterprise Manager and explain how it will make managing and deploying ColdFusion applications easier and more powerful than ever before.

Background

Before ColdFusion 4.0, no high availability (HA) features were built into ColdFusion. One could use a hardware load-balancer to distribute requests to web servers, but this was an expensive option back in 1996. ClusterCats, introduced with ColdFusion 4.0, provided clustering and load-balancing right out of the box for customers who purchased ColdFusion Enterprise. With the introduction of ColdFusion MX, ColdFusion was once again without built-in HA features. The move to Java/J2EE brought many benefits, yet precluded the continued use of ClusterCats.

One of the main benefits of the move to J2EE is the ability of ColdFusion to take advantage of features provided by the underlying app server. Since ColdFusion is a certified J2EE application, it can run on any certified and many non-certified J2EE application servers. The two most prominent of these are BEA Weblogic and IBM Websphere. Both provide robust HA features and multiple instance support. They are, however, an additional expense above the price of ColdFusion itself and will require you to become familiar with their proprietary administrative functions.

As most of you reading the article already know, the underlying app server for ColdFusion is JRun, Macromedia's popular and easy-to-use J2EE server. JRun is a full-featured J2EE app server with clustering and multi-instance support. However, if you're not familiar with J2EE or the management/configuration of JRun, getting up and running with multi-instances and clustering may be a difficult task. In addition to a good understanding of JRun you will need to understand the basics of J2EE and the Java Runtime Environment (JRE). For the average ColdFusion user, there are several layers between you and a HA environment.

Our goal with ColdFusion MX 7 was to reintroduce HA features in a way that was intuitive and easy for the typical

ColdFusion customer. The Enterprise Manager, found in the CF Administrator of ColdFusion MX 7 Enterprise, provides an easy-to-use GUI for configuring multiple servers and for managing clusters. ColdFusion users can get all the benefits of clustering and multiple instances right from the comfortable confines of the CF Admin without the need for any J2EE or JRE expertise.

Multiple ColdFusion Instances

Multiple instances (sometimes known as *multiple servers*) refer to J2EE servers' ability to deploy multiple copies of an application server to a single computer. Technically, a new JVM (Java Virtual Machine) is deployed per server instance. The benefit is simply that each new application server has its own process to use and a failure of one server cannot affect other servers; this is usually referred to as *process isolation*. In ColdFusion MX 6.1 Enterprise, it was possible to configure multiple ColdFusion servers, though it was a highly manual process and required the use of the JMC and knowledge of J2EE. With ColdFusion MX 7 creating multiple ColdFusion servers on a single computer is done via a simple interface in the CF Administrator. Each new ColdFusion server has its own CF Admin and is configured separately just as if this were a copy of ColdFusion running on a completely different physical computer. A single machine can now host any number of individual ColdFusion servers, bound only by processing power and RAM. Many new deployment options are available where individual applications, which were once directories, can now be deployed as separate processes. Clustering can also be used across physical machines as well as across instances deployed on the same machine or any combination of physical and logical servers.

Deployment Scenario

Until ColdFusion MX 6.1 Enterprise, a computer could have only one instance of the ColdFusion server process. This means that the file system was the only method for logically separating applications. Moreover, this type of partitioning gave you no flexibility with respect to the runtime aspects of the ColdFusion server process. It was really just a way to keep an application's templates and other resources together in a directory structure, and there was no way to control an application's access to the ColdFusion runtime. With 6.1, it was possible to configure JRun to host multiple instance of ColdFusion, and with ColdFusion MX 7 this is even further simplified thanks to the new ColdFusion Enterprise Manager.

It's easy to imagine that many ColdFusion servers sitting on corporate intranets host a great number of applications, each with their own users and load demands. If you are running ColdFusion MX or earlier, there is no way to ensure that one busy app on a server doesn't use all of the available resources. If ColdFusion is configured for 10 simultaneous requests and one app is so busy that those threads are always servicing clients of that app, then other apps will seem slow or non-responsive. To fix this situation before the release of ColdFusion MX 7, it would have required a new piece of server hardware and a separate ColdFusion license. With multiple ColdFusion instances, this is an easy problem to solve without the added expense and maintenance of multiple physical servers. Simply create one ColdFusion instance with 10 available threads for the busy app and another ColdFusion instance to host all of the lower demand apps. Now all apps on the server will appear responsive and spikes in demand in one app won't affect the availability of others. As demand grows in particular sections of your website, you can easily create new instances to meet this demand and get a lot more value out of a single piece of server hardware and a single ColdFusion MX 7 Enterprise license.

HA in Detail

Below you'll find a high-level description of some of the HA features available in ColdFusion MX 7 Enterprise. This isn't meant to be an exhaustive analysis of how to use these features in a particular environment. However, it should help you understand the basics of HA computing. The multi-instance feature alone should be enough to justify the move to ColdFusion MX 7 Enterprise, especially for sites that host multiple applications on a single physical server. And if your server environment hosts mission-critical applications where 100 percent uptime is the goal, then the clustering features of ColdFusion MX 7 are a must-have.

Load Balancing

Load balancing is the simpler of the two HA methods. It refers to the act of distributing requests across *n* number of computers, based on an algorithm for determining load. This can be as simple as round-robin (which isn't technically *load* balanced) where requests are sent to each computer in turn or more complex methods based on, for example, CPU utilization. In this case, some method for reporting load on a CPU has to be communicated back to the load-balancing software. The clustering features in the ColdFusion Enterprise Manager can be configured to act as a simple load balancer, handing out requests to the cluster members. There are three algorithms to choose from: round robin, weighted round robin, and random round robin. Strictly speaking these are not load balanced. However you can use weighted round robin (WRR) to simulate a more complex load-balancer. WRR allows you to apply a weight to each machine in the cluster so that in a mixed hardware environment, where some machines may be more powerful than others, the cluster can be configured to send a proportionally higher volume of requests to those more robust machines. Random round robin simply routes requests randomly among the cluster participants.

Clustering

Load balancing is not usually "application aware." Its job is simply to route incoming requests. The point at which your cluster becomes application aware, and not just a simple load balancer, is when you configure your cluster to replicate session data. The ColdFusion Enterprise Manager's Cluster Manager takes care of copying session data to each node in the cluster so clients of your application will not experience any change in the event of a server failure. Sessions can move freely among members of the cluster or be configured to be "sticky" where each particular client will be routed to the same node for the duration of the session. With either sticky sessions or not, replicated data will always be up to date on every node at the moment it is requested in your app.

Another thing to note is that there are *no changes in the way you use the session object in your CFML code to take advantage*

of session replication. The Cluster Manager works under the covers and is entirely invisible to your application. An important aspect of session replication to keep in mind, however, is that since this feature is not part of the J2EE spec the behavior of session replication is not consistent across the range of app servers that support it. Some, like WebSphere, explicitly document the fact that they will only replicate instances of classes found in the JDK. This means that if you are developing a ColdFusion application using the session object that may at some point be deployed on an app server other than ColdFusion's underlying JRun, you should be conservative about the type of objects you store in session.

WARs, EARs and Sourceless Deployment

A ColdFusion Application contains an application's resources, such as ColdFusion Markup Language files (CFM), ColdFusion Components (CFC), and static resources such as HTML pages and image files. The application may also require configuration information such as details about data sources. Managing and deploying an application with all of those resources and settings, for instance moving an application from a development environment to production, can be difficult. The J2EE specification has a solution; a WebARchive (WAR) file. This is a single file used to package an application for easy deployment and it is the standard way to deploy applications in J2EE environments. The archive contains all of the application's logic, resources, and the ColdFusion runtime required to run it.

"One of the main benefits of the move to J2EE is the ability of ColdFusion to take advantage of features provided by the underlying app server

ColdFusion MX 7 automates the task of creating these J2EE archives. All versions of ColdFusion allow you - from the ColdFusion Administrator - to create a WAR or EAR file from a specified set of application directories. You can configure data sources to be included in the archive, chose to include or exclude the ColdFusion Administrator in the packaged application, include or exclude external components (such as COM support), and more. And just like the Enterprise Manager, the J2EE packaging feature of ColdFusion is as simple as creating a datasource.

ColdFusion MX 7 also supports packaging and deploying ColdFusion applications as Enterprise Archives (EAR files) – another Java 2 Enterprise Edition (J2EE) application archive format. Whether your choose EAR or WAR is entirely driven by the deployment environment (i.e. Tomcat is a servlet engine and only accepts WARs).

It's important to point out that you can *create* a J2EE package of an application with any version of ColdFusion (Developer, Standard, or Enterprise), but they must be *deployed* onto a server running ColdFusion Enterprise edition. When you create the package, you will need to enter a valid licensed Enterprise serial number for the server on which the application will be deployed. If you leave the serial number out, the application will be deployed in Developer mode (IP restricted).

Another new option we developed for ColdFusion MX 7 is the ability to write a ColdFusion application and deploy it without distributing the unencrypted, human-readable source code. This is presented as an option when you are creating your EAR or WAR file. If this box is checked, the CFML that makes up your ColdFusion application will not be included in the archive, but your application will still run. This gives your intellectual property more protection than was ever possible before in ColdFusion. The Java byte code is not human readable. Even if the byte code were decompiled, it would still only result in Java source code, which is still very far from CFML.

This is achieved by compiling the ColdFusion files to Java byte code. If you expand the archive you will still see your applications directory structure and even all of the files that make up your application - images, style sheets, etc. However, if you open any of the CFML templates you will see that they do not contain CFML, but instead contain Java byte code. Give it a try; you'll see byte code completely obscures your application logic.

Summary

ColdFusion has had features for many years that allowed you to create and deploy high performance scalable web applications in even the most demanding environments. However, with ColdFusion MX 7, we've made many of the high availability, scalability, and management features more powerful and much easier to use. If you aren't familiar with J2EE, you can still take advantage of many of its HA strengths by using the new ColdFusion Enterprise Manager. If you are in a standardized J2EE organization, you can use the new J2EE package and deployment features to make ColdFusion fit more easily alongside any other Java application. And, in either case, the job of managing and deploying these important applications is done in the kind of intuitive and highly productive ways that you've come to expect only from ColdFusion.

About the Authors

Tim Buntel is senior product manager for ColdFusion at Macromedia.

Geoff Greene is Director of Software Engineering, Bullhorn.

tbuntel@macromedia.com,

ggreene@bullhorn.com

CFDJ Advertiser Index

CONSTRUCTING AN APPLICATION WITH FLASH FORMS FROM THE GROUND UP

You can do much more with them than with simple forms

By Laura Arguello

By Nahuel Foronda

As you may know by now, ColdFusion 7 includes a new feature that lets us create forms in flash format. They work as a replacement for html forms, but give us some additional controls like the tree, grid, and calendar. Even if making a "form" doesn't sound very appealing to you, once you start using this feature you will find that you can do much more than simple forms.

But the best is that they can enhance the usability of your regular forms, even if they are simple, with built-in features such as validation, tooltips, and tabs. It doesn't take much to make a flash form; basically, we only have to specify that the form's format is flash in the cfform tag and proceed with the cfform elements.

In this article, we'll walk you through the process of creating a small application interface using only flash forms. There's a lot to discover about flash forms, although you don't need know every detail to start using them. Since going over every feature would be impossible, we'll cover some of the most commonly used features that will likely be present in your next application.

To take a look at the interface we'll construct, see Figure 1.

Step 1 – Setting the Layout

Every flash form starts with a cfform tag that encloses the whole form definition and has the format attribute defined as "flash."

```
<cfform format="flash">
```

But before we start coding our application, we must first define the layout. In our address book, we have a panel on the left for a list of contacts and three panels on the right, one for previewing the contact info, the second to edit it, and a very small panel to contain a checkbox.

Before the advent of flash forms, we would write html (tables, paragraphs, etc.) to position the form elements. We can't transfer that to flash forms; they require special layout containers. We must use the new tag cfformgroup to position and lay out all elements and controls in a flash form.

To layout our four panels, we use an enclosing <cfformgroup type="hbox"> container that lets us position its contents horizontally. In the hbox, we have a panel labeled "Contacts" and another container, a cfformgroup type="vbox." The vbox contains the other two panels "Preview" and "Edit" that are positioned one below the other. See Figure 2.

```
<cfformgroup type="hbox">
```

```
<cfformgroup type="panel" label="Contacts" width="350">
</cfformgroup>
<cfformgroup type="hbox">
<cfformgroup type="panel" label="Preview"></cfformgroup>
        <cfformgroup type="panel" label="Edit"></cfformgroup>
</cfformgroup>
</cfformgroup>
```

Step 2 – Adding Controls

Now that we have our panels positioned, we can start adding controls.

We show the list of contacts in the left panel in a grid with three columns: first name, last name, and e-mail. The grid's data comes from a query, specified in the "query" attribute of the <cfgrid> tag. firstName, lastName, and e-mail are columns of the query. Later we will need additional columns for the other fields, but we'll just assume our query has all the necessary columns. Note that we also specify rowheaders="false" because we don't want an extra column with row numbers.

```
<cfgrid name="contactList" query="contactsQuery" rowheaders="false">
   <cfgridcolumn name="firstName" header="First Name" />
<cfgridcolumn name="lastName" header="Last Name" />
   <cfgridcolumn name="email" header="Email" />
</cfgrid>
```

We want to be able to add new contacts, so let's add a button at the bottom of the grid:

```
<cfinput type="button" name="addContact" value="Add Contact" />
```

The Edit panel is a little more involved because it includes 16 inputs: 14 text inputs, one text area, and one hidden input to contain the id of the contact.

```
<cfinput type="text" name="firstName" label="First Name:" />
…the rest of the inputs…
<cfinput type="hidden" name="contactId" />
<cftextarea name="comments" height="45" label="Comments:"></cftextarea>
```

Let's also add two buttons labeled "Delete" and "Add New." These buttons are contained in a <cfformgroup type="horizontal"> tag that lays them next to the one another.

```
<cfformgroup type="horizontal">
<cfinput type="submit" name="deleteContact" value="Delete" />
   <cfinput type="submit" name="addContact" value="Add Contact" />
</cfformgroup>
```

You may have noticed that "hbox" and "vbox" types are used when we want to position containers horizontally or vertically respectively, and "horizontal" and "vertical" types when we want to position controls.

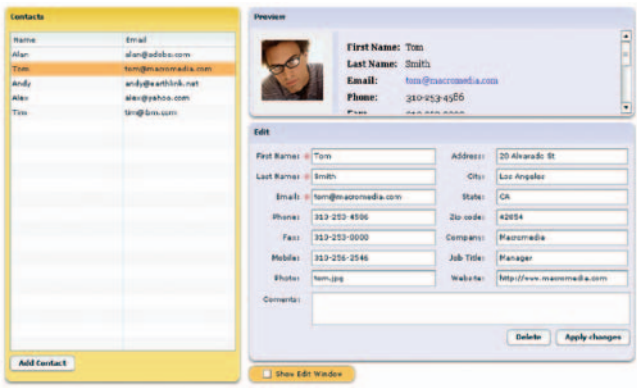


Figure 1: Edit Mode

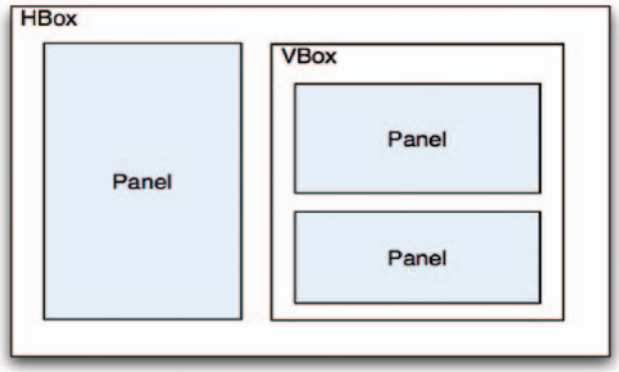


Figure 2: Layout containers

Step 3 – Binding Controls

In the Edit panel, we have what looks like a typical form: several text inputs, a textarea, and submit buttons. We added them in the last step. However, while you weren't looking, we made a little change to them. Now, every text input looks like this:

```
<cfinput type="text" name="firstName" label="First Name:" bind="{contactList.
selectedItem.firstName}" />
```

Everything looks pretty normal, except for one part – bind, a strange-looking attribute with braces around its value. The bind attribute is used to set the value of the control at the evaluation of the expression between the braces. That means that we can put any ActionScript expression that can be evaluated inside the braces and the result will become the value of the control.
In our form, we're populating the text inputs with data from the contactList grid, which contains all the columns and rows of the source query. To get the data corresponding to the selected row in the contact list, we access the cfgrid by its name, "contactList" and the special property "selectedItem," which points to the row that's been selected by the user. Then, by using dot notation, we get to the specific column we need, which is different for every input. The complete path would be "contactList.selectedItem.columnName."

It's important to note that the column name is key-sensitive and must match your database column name. The result of "contactList.selectedItem.columnName" will be the value of that specific column of the currently selected row, and that's exactly what we want to show in the input. By adding a binding to every input, we can magically populate them when the user selects a row in the contact list.

But that's not all we can do with bindings. We can use the

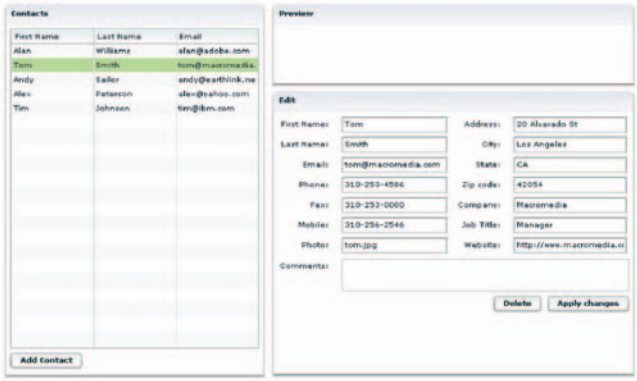


Figure 3: Binding controls

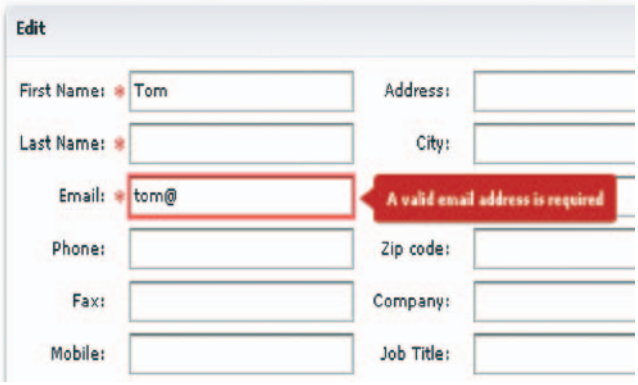


Figure 4: Validating data

binding syntax in other attributes besides the bind itself. For example, if we wanted to bind the label of a control with the data of another control, we would write:

```
label="{otherControl.text}"
```

We use that technique to change the value of the submit button to be in synch with the current operation. We show "Add new" when there's no item selected in the contact list, or "Apply changes" when an item is selected. The label of a button is given by its value attribute, so we write:

```
<cfinput type="submit" name="submit"
value="{(grid.selectedItem.id == undefined) ? 'Add new' : 'Apply changes'
}" />
```

Because what goes between braces must be a single expression, we need to use the shorthand syntax for an if-else statement.

Step 4 – Validating Data

In almost every form we create, when we get input from the user, we must verify that the data entered meets minimum requirements. Sometimes we need certain fields to be mandatory, other times we must ensure that the data complies with specific formatting guidelines. From the user's standpoint, however, there's nothing more discouraging than getting an error message at the top of the form after the form has been submitted and comes back from the server and realize that the error was simply a missing zero in the month of a date field. Fortunately, we can avoid most of these user-unfriendly forms by using the built-in validation features in flash forms.

The most basic kind of validation is ensuring that required fields have been filled out. When we add the attribute required="true" to a <cfinput type="text"> tag, a red star is added to the input label and the form won't submit until that field is properly entered. The user will also get an alert indicating the error, and even better, a red border in the problematic field with an explanatory message will appear on the mouse over.

Let's take a look at the First Name field as an example:

```
<cfinput type="text" name="firstName" required="true" validate="noblanks"
message="First Name is required" />
```

Besides making the field required, we want to ensure that at least a non-blank character is entered, so we set the validate attribute to "noblanks." The message attribute lets us write a friendly message that will be shown in the red background when the user places the mouse over the field. If we don't specify it, a default message will show up.

Our form only requires the first name, last name, and e-mail. All other fields are optional, so we can simply omit the required attribute or set it to "false."

Once the data's been entered, we can validate it against one of the built-in types: date, integer, range, telephone, zipcode (US), e-mail, URL, creditcard, and others. You can take a look at the ColdFusion documentation for a complete list of types. What's nice about them is that the user gets notified with the red border and a message if the data doesn't validate as soon as he or she leaves the field, making it easier and faster to correct. See Figure 4.

Last, we may want to get the data in a specific format such as an ISBN number or an account number that must include dashes between the numbers. We can enforce such a pattern by using the mask attribute. For example, in the phone field, we use:

```
<cfinput type="text" name="phone" mask="999-999-9999" />
```

That means only three numbers followed by a dash, three numbers, a dash, and four numbers are allowed in that field. That may sound a little too much to ask from the user, but the good news is that the user will only need to enter the numbers (no other character will be written even if he or she hits the keyboard) and the dashes will be added automatically. If the user does add dashes, that will be fine too, because it will conform to the mask. Say goodbye to the three text inputs for phone numbers!

Step 5 – Showing Text and Pictures

Not everything in a form has to be an input of some kind.

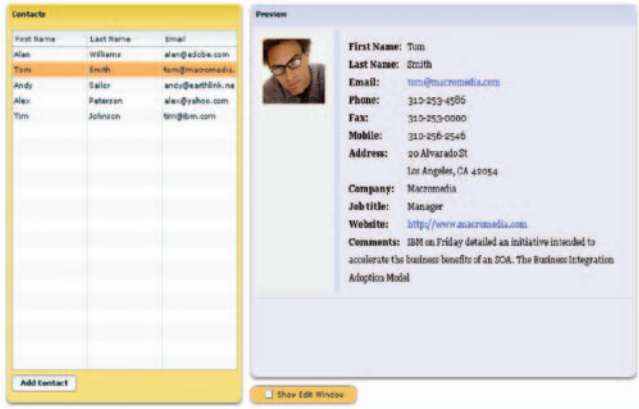


Figure 5: View mode

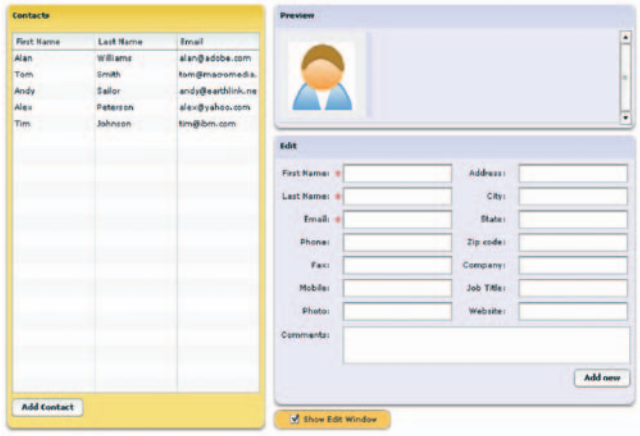


Figure 6: New mode

Sometimes we simply want to show a message or a picture. Because everything we want to have in the form has to be in the cfform tag and, unlike html forms, any text we put in between tags is ignored when we switch to flash format, we need a way to add text. That way is by using the cfformitem tag, a new tag added in ColdFusion 7. It lets us insert plain or html-formatted text and other elements such as rules or spacers.

We only want to show the contact information and a picture in the Preview panel. We'd also like the titles of the fields to be in bold. To be able to format the text like we want it, we need to use the html type of the cfformitem tag:

```
<cfformitem type="html">some html string</cfformitem>
```

If the text we want were simply a static string, we'd put it between the opening and closing tags. But in our case, we want to show the information specific to the selected contact in the contact list. How do we do that? Remember the bind attribute? Well, we can use the bind attribute to affect the value of the form item. Just like we bound the value of a text input to the selected item in the cfgrid, we can bind the value of the cfformitem to the fields in the Edit panel. We are, in fact, chaining the binding in such a way that when the selected item in the contact list changes, the text inputs get populated with the data, and that in turn populates our html cfformitem. It may sound more confusing that it actually is, so let's review the code for the picture.

First, we have an input in the Edit panel for the contact's picture called "photo" that gets its value from the cfgrid photo column.

```
<cfinput type="text" name="photo" label="Photo:" bind="{contactList.
selectedItem.photo}" />
```

Then, we have a cfformitem that will get its value from the photo field above:

```
<cfformitem type="html" height="100" width="100"
bind="<p><img src='{photo.text}'/></p>">
</cfformitem>
```

What we're trying to achieve is for the value of the cfformitem to be , a simple html paragraph with an image inside. But the name of the picture changes for each contact so we bind the path to the photo field in the Edit panel by writing the path of the picture between braces: {photo.text}.

If you look at the source code, you'll find a slightly more complex version of the picture's cfformitem bind:

```
bind="<p><img src='{(photo.text == '' ? 'man.jpg' : photo.text)}' /></p>"
```

We're using the if-else shorthand statement again to decide whether we want to show the picture of the contact or a default if the contact photo is empty.

Now that you're getting a feeling for how binding works, we'll add the complete contact information next to the picture. In this case, the binding is rather lengthy, so we'll make a variable containing the binding string and put the variable as the value of the attribute:

```
<cfsavecontent variable="previewBind">
   …the contents of the bind…
</cfsavecontent>

<cfformitem type="html"  bind="#previewBind#"></cfformitem>
```

The content of the previewBind variable is:

```
{(
((fistName.text != "") ? "<b>First Name:</b>\t" + fistName.text:"") +

   …plus all the other fields…

   ((comments.text != "") ?"<br /><b>Comments:</b>\t"+ comments.text :
"")
)}
```

As you can see, we're using the same trick we used for the

photo: if the field isn't empty, we show the value; otherwise we don't show anything. We concatenate all the fields with a "+." We also format the field label with bold and add a tab (\t) between the label and the value to align it.

In case you're wondering how we got the picture next to the text, we used layout tags inside Preview panel (bindings explained above omitted for simplicity):

```
<cfformgroup type="panel" label="Preview">
    <cfformgroup type="hbox">
        <cfformgroup type="hbox" width="116">
            <!--- the picture --->
            <cfformitem type="html" height="100" width="100">
</cfformitem>
        </cfformgroup>
        <cfformgroup type="hbox">
            <!--- the complete contact data --->
            <cfformitem type="html"></cfformitem>
        </cfformgroup>
    </cfformgroup>
</cfformgroup>
```

Step 6 – Styling the Form

Now that all the elements are in place, we'll add some color and styling to the form. In flash forms, the look-and-feel of each element must be defined in the "style" attribute. Many styles defined in outer containers also apply to inner containers and controls.

As you can see in the picture of the application (see Figure 1), we applied colors to the panels. The contact list has an orange background color and the other two are blue. The complete style for the orange (Contacts) panel is:

```
style="themeColor:##FE7E00; headerColors:##F2CB2A, ##FFE57A; backgroundC
olor:##FFE57A; panelBorderStyle:'roundCorners';"
```

In this style definition there are three properties applied:

themeColor: color;

Instead of using one of the halo themes (haloBlue, haloGreen, haloOrange, haloSilver) that can be applied with the skin attribute (skin="haloBlue"), we used our own custom color. The themeColor property applies the given color to all the form elements just like halo themes do. The advantage is that we can pick any color, not just one of the four standard themes.

headerColors: color1, color2;

The panel header is composed of a linear gradient of two colors that goes from top to bottom. We can change both of these colors with the headerColors property. The bigger the contrast between the two, the more pronounced the gradient will be.

backgroundColor: color;

The background color of the panel can be changed by using this property. Note however that the cfgrid isn't inheriting the background.

panelBorderStyle: 'roundCorners';

The upper corners of a panel are always round by default. The bottom corners aren't, so we use this property to make them round like the top corners.

You may be wondering why we need to use double # when specifying the colors. That's necessary because otherwise ColdFusion will interpret them as variables and throw a compile error. It's similar to the character escaping required when outputting html with inline font colors inside <cfoutput> tags

The other two panels have similar properties with different color values. In addition, the small panel at the bottom has no header, given by the style "headerHeight:0."

Step 7 – More User Interactivity

So far we have some functionality like populating the Preview and Edit panels on user selection of a contact in the contact list. We'll add some more interactivity by showing or hiding panels and changing the label of the buttons depending on the current activity.

When we first load the application, we see the list of contacts on the left, the Preview panel on the right, and a small panel with a checkbox at the bottom. We don't see the Edit panel. Why? Because it's hidden, we are in "view" mode. (See Figure 5.) Also note that the checkbox at the bottom "Show Edit Window" is unchecked.

```
<cfinput type="checkbox" name="editShow" checked="false" label="Show Edit
Window" />
```

If we want to edit the selected contact, we can click the checkbox and the bottom panel will appear and the top panel will shrink (). We achieve that effect by binding two attributes of the panel, the visibility and height, to the state of the checkbox:

```
<cfformgroup type="panel" label="Edit"
visible="{editShow.selected}"  height="{editShow.selected ? 332 : 0 }">
```

We can access the checked state of checkboxes by reading the "selected" property. In this case, if the checkbox is selected, it will set the visible property of the panel as true and the height of the panel will be set as 332 pixels, effectively making it bigger and shrinking the upper panel. On the other hand, if the checkbox isn't checked, the height of the panel will be 0 pixels and its visible property will be false.

But that's not the only way we can show the bottom panel. If we want to add a new contact, we need the Edit panel to be visible and all its fields empty ready for new data (see Figure 6). If you remember, there was a button at the bottom of the contact list labeled "Add contact." If the button is clicked, we want to show a clean Edit panel:

```
<cfinput type="button" name="addContact" value="Add Contact"
onClick="contactList.selectedIndex = undefined; resetForm(); editShow.
selected = true;" />
```

As you can see, the ActionScript code of the onClick attribute has three statements:

contactList.selectedIndex = undefined
which removes any current selection in the contact list,

resetForm()
a built-in function that cleans all the data in the form fields, and

editShow.selected = true
which simply checks the editShow checkbox but, as we know, that triggers a couple of events because the Edit panel properties are bound to the checkbox state.

That's not all though. In "new" mode, there's only one submit button in the bottom panel "Add New," whereas in Edit mode, there are two, "Delete" and "Apply Changes." How do we make that button appear and disappear? By now we're sure you may be guessing that we can use binding in the visible attribute of the delete button.

```
<cfinput type="submit" name="deleteContact"
visible="{contactList.selectedItem != undefined}" value="Delete" />
```

The two submit buttons "Delete" and "Add New/Apply changes" simply submit the form. What we get on the server are the values of the fields just like an html form. We're sure you're familiar with html forms so we won't go over the server-side functionality. The beauty is that if you already have a regular html form, you can easily convert it into a flash form because once the form is submitted, everything looks the same, be it a flash form or an html form.

Conclusion

We've gone a long way towards making our application interface more user-friendly. And we've done that only with flash forms! We hope you'll start exploring this new feature; you'll find there's a lot more you can do with it. We encourage you to download the source code (see this article on www.cfdj .com) and play around with it.

About the Authors

Nahuel Foronda and Laura Arguello are founders of Blue Instant (http://www.blueinstant.com), a web development firm specializing in Rich Internet Applications where they have been creating award-winning applications and offering training for the last five years. They also maintain a blog, called AS Fusion (http://www.asfusion.com), where they write about Flash, ColdFusion and other web technologies.

www.blueinstant.com

Rethinking Decorators

An alternative to the Decorator pattern

By Hall Helms

By Ben Edwards

In this article, writes Hal Helms, Ben Edwards (of Mach-II fame) and I rethink the Decorator design pattern.

Hal: So, did you see my ***CFDJ*** article on decorators?
Ben: Yeah. I saw it.
Hal: So, what did you think?
Ben: Let's just leave it at that: I saw it. Ben and I were talking about my earlier CFDJ article on the Decorator design pattern [cfdj, vol. 6 issue 11]. In the article, I explained how the Decorator pattern could be used in situations in which there exists a base class with various options. In my article, I used the example of a base Coffee object with options such as ExpressoShot, WhippedCream, and FlavoredSyrup. The Decorator is useful in avoiding the sort of "class explosion" that would occur if we were forced to create separate classes reflecting all the possible permutations of base class with options: CoffeeWithExpressoShot, CoffeeWithWhippedCream, CoffeeWithExpressoShotAndWhippedCream, CoffeeWithExpressoShotAndWhippedCreamAndFlavoredSyrup, etc.
Ben: I'm not a big fan of Decorator as it's applied here.
Hal: What are your objections to it?
Ben: In order to have the Decorator work, as you explained in your article, you have to have all the options as subtypes of the base class.
Hal: Right. In the article example, FlavoredSyrup, WhippedCream, and ExpressoShot all extend the base Coffee class.
Ben: Making for an inheritance relationship.
Hal: Right.
Ben: And what's the test for creating inheritance relationships?
Hal: That the subclass "is-a" type of the base class.
Ben: And you consider WhippedCream to be a type of Coffee?
Hal: Well…no.
Ben: And, presumably, you'd need other options like ExtraHot, LowFoam, etc?
Hal: Right.
Ben: And those things aren't Coffee subtypes, are they?
Hal: No.
Ben: Well, that's my objection. It's not the best use of inheritance.
Hal: Yes, I see your point, but…
Ben: But what?
Hal: But it's such a danged useful pattern, I guess I'm willing to forgive a little impurity. It's a little like watching a movie. Typically, there's one presupposition that you have to accept – even one that may be a little far-fetched. You suspend disbelief just a little – and then you can accept the movie on its own terms. I guess I feel that way about the Decorator pattern.
Ben: The Decorator pattern was initially introduced by the "Gang of Four" in their book, *Design Patterns*. That was also the book that laid out the very useful, very good principle: "Prefer composition to inheritance." Composition exists when one class holds as an instance variable member(s) of another class. And that same good principle can be used to solve the problem of a base class with options. Using composition, we can keep separate the coffee from the extras you can add to it. I'd have the Coffee class composite its own options. See the way it's shown in Figure 1:
Ben: In this example the extras are added to the coffee rather than creating a wrapper around it. The coffee's description and price, for instance, are derived from the coffee itself and from the extras it has had added to it.
Hal: Yes, that would certainly work. What are the benefits of this approach over using the Decorator pattern?
Ben: Well, for one thing, the model conforms to the real world more closely than does the Decorator, where you end up with the bizarre situation where WhippedCream is a type of Coffee. As a matter of fact, the other day at the local Starboard coffee shop, I tried to order a cup of whipped cream. They wouldn't let me get whipped cream by itself; I could only get it in addition to a cup of coffee. They insisted that's why the coffees are separated on the menu from the "extras" – an "extra" is *not* a coffee. Separating the two better encapsulates the intent of each, providing more flexibility.
Hal: Agreed. Any other benefits?
Ben: You can tell the difference between the base coffee and the extras. With the Decorator pattern, the base class is subsumed by the Decorator class(es). I always know that I have a cup of Coffee – that has some extras added to it.
Hal: Any more benefits?

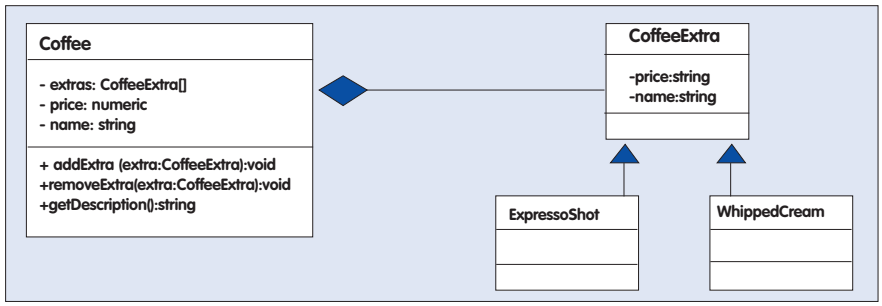


Figure 1: Replacing the Decorator with composition

Ben: You don't run the risk that Decorators will override either a method of either the base class or one of the "inner" decorators.

Hal: I suppose that's something of a double-edged sword. Decorators do provide you with the ability to override their base class's methods while composition doesn't. You actually might want that.

Ben: You could, yes. And finally, with composition, you can easily remove an extra –something you can't do (or at least can't do easily) with Decorators.

Hal: Those are good points…

Ben: But you're unconvinced.

Hal: Well, I can see the benefits you've laid out, but I fundamentally don't like the idea that Coffee has to know about its extras. It seems that Coffee should only concern itself with itself – not with what might be added to or done with it. If Coffee has that responsibility, I worry that future needs will force me to alter the code for Coffee, rather than add a Decorator. That, for me, is a great benefit of using Decorators – the base class is only responsible for itself. It provides a layer of abstraction – and you know how I love abstractions.

Now, you mentioned the case of an extra-hot coffee. In your scheme, would ExtraHot be a "CoffeeExtra" as you've defined it?

Ben: That does seem to be stretching things.

Hal: But a Decorator could handle it easily.

Ben: I agree: both patterns have their advantages and disadvantages. This example isn't a replacement for the Decorator pattern, but a different solution for this context.

Hal: Maybe that's the central point, Ben: when developers learn design patterns, they need to understand that they're not magic bullets. A pattern may be perfectly good in one context, but not well suited for another. I've seen developers force patterns into their code just for the sake of buzzword compliance.

Ben: Yeah, points will be deducted for gratuitous use of design patterns! Also, one pattern may just fit better with a developer's preferences. You're troubled by the idea that Coffee knows about its extras while it seems completely right to me that a Coffee should take on this responsibility. The point of learning design patterns isn't so that you can slavishly use them, but that you gain a wider perspective on how to solve problems in an elegant way that allows code to evolve gracefully.

Hal: So, one thing we can both agree to is that I'm right.

Ben: You said it: I'm right.

About the Authors

Hal Helms is the author of several books on programming. Hal teaches classes in Java, C#.NET, OO Programming with CFCs, Design Patterns in CFCs, ColdFusion Foundations, Mach-II, and Fusebox. He's the author of the popular Occasional Newsletter and his site is www.halhelms.com.

Ben Edwards is a Sun Certified Java Programmer and holds a degree in computer science from the Georgia Institute of Technology. He currently trains developers on software engineering practices focusing on Java, object-oriented programming, and software architectures. Ben is also cofounder of the Mach-II project.

hal@halhelms.com

ben@ben-edwards.com

"Convergence Is Now Coming Alive"

Talking with Jeremy Allaire

ColdFusion's 10th Birthday... and the Next 10 Years

By Simon Horwith

By Jeremy Allaire

ON THE PAST...

"Back in 1997 I created a simple application in CF that presented a UI via the very first Web phone (the AT&T PocketNet Phone), and allowed me to turn on and off the lights and coffee maker in my house"

ON THE PRESENT...

"For the past few years I've spent a great deal of timing working on and thinking about the application of rich media, video, and media distribution technologies"

"It's hard to think of a kind of Internet or Web application that ColdFusion couldn't be used for building"

"most…institutions – learning institutions, government institutions, religious institutions, are augmenting the ways in which they organize, communicate, and behave based on the shifts and possibilities of the Internet"

ON THE FUTURE...

"Macromedia has a huge head-start [over Microsoft] in building both the runtimes and tools needed for this unique fusion of design, programming, media and interactivity that characterizes RIAs"

"The metaphor of self-published micro-content, feeds, syndication, etc. are taking hold, and they are moving into media worlds through podcasting, vlogging, and so forth"

"I'm very positive on this [Adobe-Macromedia] merger/acquisition. I think it's a natural fit, there is very little overlap for the companies, and it will give the combined company critical mass to innovate in design, production and development tools, and also in applications for corporations"

On the eve of ColdFusion's 10th Birthday, CFDJ Editor-in-Chief Simon Horwith sits down with one of the true pioneers of the Web, the man who started it all, Jeremy Allaire, to talk about the past, present, and future of ColdFusion…

Simon: *Jeremy, ColdFusion turns 10 years old this month. Did you ever think, in the beginning, that it would last this long?*

Jeremy: I think when we were working on ColdFusion in the early days, we definitely had the sense that what we were doing was going to have very broad, and very significant impact. We were shooting to help create a mass phenomenon with the Internet, and to put the power of the Web in everyone's hands. As such, I think we all believed that what we were doing would have lasting impact. 10 years and going strong, definitely didn't expect or even think about that kind of time horizon, but it's just great.

Simon: *I agree; it is terrific. So, what have you been up to these days? Are you still involved in what goes on at Macromedia? What trends are you keeping an eye on?*

Jeremy: Over a year ago, I founded a new company called Brightcove. We're building an online service for the distribution of video and television on the Internet. It's a very exciting project, really coming at a time when the television and media industry generally is sort of in a state of siege, driven by digital distribution and convergence. As such, for the past few years I've

spent a great deal of timing working on and thinking about the application of rich media, video, and media distribution technologies, and marrying that with all the things I've learned in the past 10 years about the Internet, online services, e-commerce, and so forth.

As part of Brightcove, I'm taking advantage of a number of key Macromedia platform technologies, so am staying close to developments there, and provide feedback and input into what I'd like to see evolve.

Simon: *Since you left Macromedia, ColdFusion has moved onto the J2EE platform and ColdFusion Components have introduced exciting new ways for developers to encapsulate business logic and take advantage of Object Oriented features. The most recent release, ColdFusion MX 7, introduced many new features such as a reporting engine and report design tool, Flash Forms, support for PDF and FlashPaper generation, and an event gateway framework that allows CFML applications to fully leverage the power of Java in order to communicate with other systems via a variety of protocols including SMS and to do things like listen for changes in the file system and asynchronously execute code. What are your thoughts on how the ColdFusion server has evolved? What do you think ColdFusion will look like in another 10 years?*

Jeremy: I think it's just wonderful how the platform and product has evolved. We started the process of moving CF to Java back at Allaire, and while I was at Macromedia we released one of the most important releases in its history, ColdFusion MX. It's been really rewarding to see the ColdFusion team thrive, to innovate around features that customers have wanted for a long time, and to find unique synergies with the rest of the Macromedia platform. In many respects, it's hard to think of a kind of Internet or Web application that ColdFusion couldn't be used for building.

Another 10 years is a long time, and I have no idea how and where it will evolve. But, given the strength of the product-line, and the fact that it's experienced growth in the market right now, I suspect that, just as in the past, customers will lead the way in terms of features and innovations.

Simon: *I agree – speculating on what the Web might be in 10 years is difficult. What about in the next year or two? I know Macromedia has been focussing heavily on delivering rich content to devices. Adobe cites this as a topic of major interest to them, in their Macromedia acquisition presentation. At Brightcove, you are focussed right now on delivering video and television via the Internet. My question is whether or not you believe that sometime soon, common everyday activities are going to become another form of network activity? Does the evolution of the Web include blurring the boundries between peoples' "online activity" and "real-world activity"? So far, most attempts to cross these boundries have been successful from a technology point of view, but not popular. Why is this and will this always be the case?*

Jeremy: Well, I definitely think the biggest thing happening on the Internet right now is that we're finally seeing the promise of convergence come alive, and this is a consumer revolution – a revolution in all forms of communications and media. Many of the original promises of the Internet and convergence are finally coming together, and this will all be manifest over the next few years.

As for the connections to real-world activity, I guess I feel like that has been happening all along on the Internet, and is just getting deeper. So, explicit things like consuming media, or shopping, or communicating, are very "real-world activities," and then most other institutions – learning institutions, government institutions, religious institutions, are augmenting the ways in which they organize, communicate, and behave based on the shifts and possibilities of the Internet.

I guess one could go even deeper, and talk about monitors, sensors, and devices that are connected and embedded everywhere, and network connected, meshing with the physical world. I think that's all interesting, but probably not that practical for consumers, yet. There's a lot of this happening in industrial settings, so for example, real-time monitors of physical manufacturing infrastructure, that has dynamic feedback from software, that is modifying a manufacturing process, and so forth.

Simon: *If, at some time in the future, the Internet is "built-in" to a larger portion of daily life, will traditional personal computer Web browsers become obsolete, because our phones, televisions, cars, and other applications will be Web browsers themselves? Do you still see application servers like ColdFusion playing an important role or do you think that Web programming will be more focused on programming for devices or plug-ins?*

Jeremy: This is an accurate world-view, and it's been an idea that was important to ColdFusion very early, when we created things like CFHTTP, and then WDDX. We believed that connected applications would be device independent, and it would be helpful to have high-level abstractions for interacting with these devices, handling events from them, presenting data and so forth. Back in 1997, we were actually the first application server to support something called HDML (Handheld Device Markup Language), from Unwired Planet, in the first Web Phone. In fact, back then, I created a simple application in CF that presented a UI via the very first Web phone (the AT&T PocketNet Phone), and allowed me to turn on and off the lights and coffee maker in my house. I worked with a CF developer in Florida to build a CFX_Tag for X10 home automation control, and attached a controler to the serial port of my CF server at home, and had this all running. So these ideas were important, even back then, 8 years ago!

Simon: *Now that you view ColdFusion from more of a distance and are keeping your eye on trends in general, what are your opinions about ColdFusion versus .NET? Any thoughts about XAML and Microsoft's upcoming Longhorn platform? Microsoft obviously paid a lot of attention to Macromedia's "Rich Internet Application" vision and message that Experience Matters – they even have an answer to "Halo" called "Aero." How do you think that Avalon, Aero, and the release of all of the components of the Longhorn platform are going to impact ColdFusion developers?*

Jeremy: Well, I don't really compare ColdFusion to .NET, they're really different layers of the stack, so to speak. It's really Java vs. .NET, at least on the server-side, and clearly ColdFusion is aligned around the Java server stack, given its heritage as an open and cross-platform server environment. I wouldn't expect that to change.

I'm quite familiar with XAML and what Microsoft is attempting to do with Longhorn. It's quite clear that they've also got the

Rich Internet Application religion, which I'm proud to have been a part of pioneering at Macromedia. Clearly, Microsoft will be successful with Avalon/XAML, but it will take a very long time to play out. Macromedia has a huge head-start in building both the runtimes and tools needed for this unique fusion of design, programming, media and interactivity that characterizes RIAs. Once Longhorn is in the market in late 2006, for it to reach 50% penetration of consumer desktops might take 4 years, whereas Macromedia can upgrade the entire Internet end-user community to a new client runtime in about 12 months.

In any case, I also expect Macromedia to be responsive to and interested in integrating its products with .NET and Longhorn, so I don't think developers need to be concerned about being left behind. Macromedia has historically been a very open and agnostic software provider, and has always tried to innovate and integrate with leading platforms from Microsoft, as had Allaire.

Simon: *You were one of the early bloggers on the Web. I see that your company, Brightcove, like many other companies is taking advantage of blogging as a company tool (http://blog.brightcove.com). Microsoft has recently announced that they are adding support for RSS in the next version of Internet Explorer. What do you think of the "blog explosion" and the increasing popularity of RSS as a means for companies to syndicate information and deliver support, thoughts, and their message in general to their clients and to the public at large? Is this a trend that you think will continue or do you see RSS being replaced by something else (like video blogging for example)?*

Jeremy: Like you said, I've been an advocate for and a practitioner of blogging, both personal and corporate. It's wonderful to see it hit the mainstream, though it seems that most corporations are just beginning to think about how to apply the approach. There's clearly an opportunity to use blogging and feeds deeply within companies, and that's barely happening now, and the external facing blogging for companies is also just getting started. So, I expect the next few years to be very interesting in these areas.

Clearly, the metaphor of self-published micro-content, feeds, syndication, etc. are taking hold, and they are moving into media worlds through podcasting, vlogging, and so forth. These are some of the themes we've been working on at Brightcove, so I expect it has a bright future!

Simon: *Ben Forta has been publicly stating for years now that CFML skills alone are not enough for ColdFusion developers who want to further their careers or their applications. Once they've mastered CFML they need to also learn other programming languages. What advice do you have for ColdFusion developers?*

Jeremy: I agree with Ben here, though not universally. Clearly, many Internet applications require code to exist in different places, and with a different rigor applied to each. The two most natural adjacent languages to learn for ColdFusion developers are Java and ActionScript.

Simon: *One technology that is becoming more common for CF developers to be proficient in is XML. When XML first hit the scene it was the biggest buzzword since "World Wide Web" and "Java." Unfortunately, in the beginning there was a lot of talk but few examples of XML being put to good use. WDDX was one of the first good applications of XML to solving a specific need, followed later by SOAP Web Services. So far you and I have already mentioned RSS and XAML –- two very interesting recent applications of XML. Many ColdFusion developers are only now beginning to examine how they can leverage XML in their applications. Do you think there will be other very useful XML-driven technologies/standards in the future? If so, what might they be?*

Jeremy: Sure, it's clear that for both data and programming, tag-based languages are here to stay, and growing in popularity. I have to say, it's incredibly rewarding, as we were among the first to really advocate for this in a big way. Given that CF is a great vehicle for working with XML-based formats, I think it will continue to intersect with any emerging standard in this space.

Simon: *I've just got to ask – what are your thoughts on the proposed acquisition of Macromedia by Adobe Systems Inc.?*

Jeremy: I'm very positive on this merger/acquisition. I think it's a natural fit, there is very little overlap for the companies, and it will give the combined company critical mass to innovate in design, production and development tools, and also in applications for corporations. I know that Adobe is very excited about expanding their role as a platform company, and the entire family of Macromedia software – tools, servers and applications – will likely get even broader exposure and distribution through this new company.

Simon: *I agree – this is a very exiting time for people using ColdFusion and the other Macromedia products. Is there anything else you'd like to say to our readers?*

Jeremy: Keep developing with CF, keep telling Adobe what you need/want, and
we'll be back in touch when they release ColdFusion 10!

Simon: *I hope that in another 10 years, we can sit down and reminisce on ColdFusion's 20th birthday. Thank you very much for taking the time to share your views on the past, present, and future of ColdFusion and of the Web in general.*

About the Authors

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal. *Simon is a Macromedia Certified Master Instructor and a member of Team Macromedia. He has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com*

Jeremy Allaire as a Technologist in Residence at General Catalyst Partners, Jeremy Allaire focuses on identifying new investment opportunities for General Catalyst in the areas of software infrastructure, enterprise software applications, and Internet technology. Jeremy also works with the firm's existing early-stage portfolio companies on product development and go-to-market strategies.

simon@horwith.com

jallaire@brightcove.com

Subscribe Today!

SAVE 16%

12 Issues for **$89.99**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

- 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

We Programmers Need Etudes

Why musicians have a leg up on us programmers

By Hal Helms

Have you ever noticed the correlation between musicians and programmers? Some of the best programmers I know are or have been musicians. I recently had a class where 70% of the students were active musicians – some even with CDs of their work.

Perhaps one of the reasons for this pairing is that both musicians and programmers have their heads immersed in abstract concepts while their fingers (literally!) translate these into tangible "products." Musicians take concepts such as keys, scales, and silence and combine these into melodies that can be played; programmers take concepts such as types, variables, and encapsulation and combine these into programs that can be run.

Both endeavors, under their respective skins, are based on mathematics. (When computer science began to be taught at universities, it was an offering of the Mathematics department.) Both musicians and programmers learn a catalog of patterns, which they commonly use. And both music and programming require a high degree of both creativity and discipline: one without the other is not enough to achieve on the highest levels.

But musicians have a leg up on us programmers: they have etudes. An etude, literally a "study," is a piece of music written for purposes of practicing or displaying technique. For every instrument, study books (etudes) are available to help musicians learn specific techniques and practice in particular areas.

Etudes are helpful because they recognize that, in order for something to be truly useful, it must become natural and easy for you to use. It's paradoxical but true: to achieve such effortlessness, you must expend a great deal of effort, and to achieve mastery, there's nothing as helpful as practice.

It's all very good, for example, to know that a certain bit of information might be represented as XML, but without intimate familiarity with XML in general and ColdFusion's processing of XML in particular, you're unlikely to use XML on a real programming project.

There's surely no shortage of programming books that explain concepts (such as XML), but there's a dearth of structured practice pieces that test how well you can translate from the conceptual to the tangible. In short, we programmers need etudes.

My plan, over the next several months, is to help provide just such practice pieces. I hope you'll find them helpful and I'd love to get your ideas on subjects for etudes (hal@halhelms.com).

This month's etude deals with ColdFusion arrays. You can download the solution from www.halhelms.com/etudes/1.cfm. Ready?

Problems

1. The following table represents the average monthly prices for gold for the years, 1994-1996. Create an array to store this information. (Note: The month names and years are part of the array.)
2. Programmatically determine: what month had the highest average for all three years?
3. Programmatically determine: what year had the highest single monthly price for gold?
4. Sort the gold array by the average price of all months in ascending order. For example, suppose the average price of gold was: 1994 – 376.58; 1995 – 372.92; 1996 – 384.23. Then, the array's first element should be the prices for gold during 1995, followed by 1994 and finishing with 1996.
5. Remove the non-price information in the gold array above – that is, the abbreviations of the months in the first row and the year numbers in the first column.

Solutions

1. What's needed is a two-dimensional array. There are two options for this. To conserve space, we'll just get you started with creating the array. If you really get stuck, you can download the full program.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1994	386.88	381.91	384.13	377.27	381.26	385.64	385.49	380.35	391.58	389.77	384.39	379.29
1995	378.55	376.64	382.12	391.03	385.12	387.56	386.23	383.81	383.05	383.14	385.30	387.44
1996	400.27	404.79	396.25	392.83	391.86	385.27	383.47	387.46	383.14	381.07	377.85	369.00

1994	386.88	381.91	384.13	377.27	381.26	385.64	385.49	380.35	391.58	389.77	384.39	379.29

Option A: Create a monolithic two-dimensional array and populate it See Listing 1.

Option B: Create a series of single-dimensional array and "stack" them See Listing 2.

Whichever way you opt for, it's important to understand that you're producing *exactly the same array*. If you <cfdump> both goldA and goldB, you'll see they're identical. The reason? A two-dimensional array is nothing more than an array of arrays. Similarly, a three-dimensional array is just an array of arrays of arrays. And while you can't create more than three-dimensional arrays in a monolithic fashion, you certainly can by stacking arrays.

Understanding that multi-dimensional arrays are nothing other than arrays of arrays will also help you use ColdFusion's various array functions, most of which are meant to work with one-dimensional arrays. Does that mean that these functions can't be used with multi-dimensional arrays? They certainly can – if the argument you supply to the array function is a one-dimensional array.

For example, consider the ArrayAvg function. This function requires a one-dimensional array. Pass it a multi-dimensional array (such as our gold arrays) and the function will throw an exception.

But you can use the ArrayAvg function – if you provide it with single-dimensional arrays:

```
<cfset avg1994 = ArrayAvg(goldA[2]) />
```

In this example, **goldA[2]** points to the single-dimensional array of goldA:

Now, while the ArrayAvg function doesn't produce any errors, it also doesn't produce the correct answer because it will average the entire row, including the "1994" found in the first element.

2. Each of these problems helps reinforce the idea that multi-dimensional arrays are simply arrays of arrays. See Listing 3.
3. To determine the year with the single highest monthly price for gold. See Listing 4:
4. I'm not going to show you the code for this one, because I hope you'll work through this one on your own. I'll give you a couple of hints, though. You won't be able to use ArraySort. You'll need to come up with your own sorting algorithm. Sorting algorithms have received a lot of study (and argument). If you already have your own favorite one, by all means use it. If you're unsure of how to do a sort, look at Listing 5. It uses an insertion sort to sort a one-dimensional array. I'm asking you to sort a two-dimensional array, but by now this shouldn't create any great problems: you know a multi-dimensional array is nothing but stacked single-dimensional arrays Listing 5.

Unless you're an old pro with arrays (and sorting algorithms), this will probably cause you some grief, but that's the point of etudes: to place you in a contrived situation so that you can work on specific weak areas. Work through it and only download the answer if it's a choice between that and flinging yourself out of the window. (My personal philosophy is to prefer the admission of defeat to the act of defenestration – but that's just me.)

5. After that last problem, this one is easy:

```
<cfset ArrayDeleteAt(goldA, 1) />
<cfloop from="1" to="#ArrayLen(goldA)#" index="rowNumber">
   <cfset ArrayDeleteAt(goldA[rowNumber], 1) />
</cfloop>
```

Well, that's a good start on an etude for ColdFusion arrays. Hopefully, you'll be inspired to create your own problems to work on.

About the Author

Hal Helms is the author of several books on programming. Hal teaches classes in Java, C#, .NET, OO Programming with CFCs, Design Patterns in CFCs, ColdFusion Foundations, Mach-II, and Fusebox. He's the author of the popular "Occasional Newsletter." Hal's Web site is www.halhelms.com.

hal@halhelms.com.

Listing 1

```
<cfset goldA = ArrayNew(2) />

<cfset gold[1][1] = "" />
<cfset goldA[1][2] = "Jan" />
<cfset goldA[1][3] = "Feb" />
<cfset goldA[1][4] = "Mar" />
<cfset goldA[1][5] = "Apr" />
<cfset goldA[1][6] = "May" />
<cfset goldA[1][7] = "Jun" />
<cfset goldA[1][8] = "Jul" />
<cfset goldA[1][9] = "Aug" />
<cfset goldA[1][10] = "Sep" />
<cfset goldA[1][11] = "Oct" />
<cfset goldA[1][12] = "Nov" />
<cfset goldA[1][13] = "Dec" />

<cfset goldA[2][1] = "1994" />
<cfset goldA[2][2] = "386.88" />
<cfset goldA[2][3] = "381.91" />
<cfset goldA[2][4] = "384.13" />
<cfset goldA[2][5] = "377.27" />
<cfset goldA[2][6] = "381.26" />
<cfset goldA[2][7] = "385.65" />
<cfset goldA[2][8] = "385.49" />
<cfset goldA[2][9] = "380.35" />
<cfset goldA[2][10] = "391.58" />
<cfset goldA[2][11] = "389.77" />
<cfset goldA[2][12] = "387.39" />
<cfset goldA[2][13] = "379.29" />
```

Listing 2

```
<cfset goldB = ArrayNew(1) />

<cfset tmp = ArrayNew(1) />
<cfset tmp[1] = "" />
<cfset tmp[2] = "Jan" />
<cfset tmp[3] = "Feb" />
<cfset tmp[4] = "Mar" />
<cfset tmp[5] = "Apr" />
<cfset tmp[6] = "May" />
<cfset tmp[7] = "Jun" />
<cfset tmp[8] = "Jul" />
<cfset tmp[9] = "Aug" />
<cfset tmp[10] = "Sep" />
<cfset tmp[11] = "Oct" />
<cfset tmp[12] = "Nov" />
<cfset tmp[13] = "Dec" />

<cfset ArrayAppend(goldB, tmp) />

<cfset tmp[1] = "1994" />
<cfset tmp[2] = "386.88" />
<cfset tmp[3] = "381.91" />
<cfset tmp[4] = "384.13" />
<cfset tmp[5] = "377.27" />
<cfset tmp[6] = "381.26" />
<cfset tmp[7] = "385.65" />
<cfset tmp[8] = "385.49" />
<cfset tmp[9] = "380.35" />
<cfset tmp[10] = "391.58" />
<cfset tmp[11] = "389.77" />
<cfset tmp[12] = "387.39" />
<cfset tmp[13] = "379.29" />

<cfset ArrayAppend(goldB, tmp) />
```

Listing 3

```
<cfset jan = 0 />
<cfset feb = 0 />
<cfset mar = 0 />
<cfset apr = 0 />
<cfset may = 0 />
<cfset jun = 0 />
<cfset jul = 0 />
<cfset aug = 0 />
<cfset sep = 0 />
<cfset oct = 0 />
<cfset nov = 0 />
<cfset dec = 0 />

<!---add the values for each year into the
monthly variables--->
<cfloop from="2" to="#ArrayLen(goldA)#"
index="yearNumber">
   <cfloop from="2" to="#ArrayLen(goldA[yearNu
mber])#" index="monthNumber">
         <cfset current = Evaluate(goldA[1][
monthNumber]) />
         <cfset new = goldA[yearNumber][mont
hNumber] />
         <cfset setVariable(goldA[1][monthNu
mber], current + new) />
   </cfloop>
</cfloop>

<cfset largest = "jan">
<cfloop list="feb,mar,apr,may,jun,jul,aug,sep,o
ct,nov,dec" index="aMonth">
   <cfif Evaluate(aMonth) GT
Evaluate(largest)>
         <cfset largest = aMonth />
   </cfif>
</cfloop>

<cfoutput>
   The month with the largest average for all
three years is: #largest#
</cfoutput>
```

Listing 4

```
<cfset highest = goldA[2][1] & ":" &
goldA[2][2] />
<cfloop from="2" to="#ArrayLen(goldA)#"
index="rowNumber">
   <cfloop from="2" to="#ArrayLen(goldA[rowNum
ber])#" index="columnNumber">
         <cfif goldA[rowNumber][columnNumber]
GT GetToken(highest, 2, ':')>
               <cfset highest =
goldA[rowNumber][1] & ":" & goldA[rowNumber][c
olumnNumber] />
         </cfif>
   </cfloop>
</cfloop>

<cfoutput>
   The year with the single highest monthly
price for gold was #GetToken(highest, 1,
':')#, with a price of #GetToken(highest, 2,
':')#
</cfoutput>
```

Listing 5

```
<!--- create a sample array to work with --->
<cfset randomNumbers = ArrayNew(1) />
<cfloop from="1" to="7" index="i">
   <cfset randomNumbers[i] = RandRange(1, 10)
/>
</cfloop>

<!--- no apply the insertion sort on the array
--->
<cfloop from="1" to="#ArrayLen(randomNumbers)#"
index="x">
   <cfset indexOfMin = x />
   <cfset y = x />
   <cfloop condition="y LTE ArrayLen(randomNum
bers)">
         <cfif randomNumbers[indexOfMin] LT
randomNumbers[y]>
               <cfset indexOfMin = y />
         </cfif>
         <cfset y = y + 1 />
   </cfloop>
   <cfset temp = randomNumbers[x] />
   <cfset randomNumbers[x] = randomNumbers[in
dexOfMin] />
   <cfset randomNumbers[indexOfMin] = temp />
</cfloop>

<cfdump var="#randomNumbers#" label="Voila:
sorted" />
```

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

Handling PDF Forms with ColdFusion

A cost-efficient alternative to Adobe's enterprise PDF server

By Tim Burton

Over the past few years government agencies and businesses have converted many of their paper forms to static PDF documents, making them available on Web sites as downloadable files for users to print locally, fill out, and submit by snail mail. All Internal Revenue Service forms are now available this way.

However, it doesn't take long to realize that if the data on mailed-in PDF documents needs to be input into a database, or if the filled-in form needs to be kept as an electronic image, it's more efficient to have the users do the data entry in the first place.

One option is to deploy Adobe's enterprise PDF server. Although powerful and elegant, this solution is very costly not only because of the hardware, operating system, and application software needed for purchasing, but also because it represents yet-another complex server to maintain and patch. A simpler and inexpensive option is to deploy a ColdFusion-accessible utility that has methods for reading and writing PDF forms. ActivePDF's 'Toolkit' is one such utility and will handle most form-entry and imaging needs.

Setting Up

ActivePDF Toolkit requires a Windows environment; the component, APToolkitNET.dll, is installed like any Windows software by invoking an executable. The component has to be registered with the CF/Web server so that when instantiated by a CF template, the operating system knows where to find it. Once installed and registered, the workflow for each PDF Web form follows this general pattern: (1) create a PDF Web form with one of several PDF Writer packages available (such as Adobe Acrobat Professional or the newly released and less expensive PDF Nitro), (2) code the ColdFusion template(s) for handling the form after a user has submitted it. This workflow is illustrated in the example below, where the user-submitted PDF form is e-mailed to a business unit and the values inserted into a database.

Create the PDF Form

Forms are usually created by a word processor or graphics program and subsequently opened in one of the PDF writers mentioned above. When open, you add form fields and a submit button via the writer's wizard. By default, the wizard will name the form fields serially (Text1, Text2…), but if the form is submitted for database insertion, the form fields should be meaningfully named. When configuring the submit button in the writer's wizard, the URL of the file to which it will be submitted should be specified; in this example, the form is submitted to 'MyBusinessForm.cfm.' After adding form fields and a submit button, save the new PDF document to a directory on your Web server. The URL of the PDF document is then advertised to potential users, which in this example might be 'http://www.serverName/aDirectory/MyBusinessForm.pdf.' Figure 1 shows a screen shot of the PDF as it appears in the browser and – without the 'Submit' and 'Reset' buttons – as an e-mail.

Handle the Form: ColdFusion coding

The form's action file in the example is 'MyBusinessForm.cfm' shown in Listing 1 at the end of this article. There are two sections.

The first section loops through the form fields, assigning its values to a structure. For testing purposes, those values are also printed to the browser screen. Next, the form's values are inserted into a database. Figure 2 shows a screen shot of running the example. You'll notice that PDF forms function much like <html..> or <cfform…> forms.

The second section is for PDF image creation. The code creates an instance of the APToolkit object and, using the methods supported by the COM component, opens and reads the original PDF ('MyBusinessForm.pdf') into memory, and then creates a new PDF image file named with a uniqueID. The next section loops through the submitted form fields and copies them into the appropriate fields in memory, removing the submit button. The next section copies the in-memory data over to the new PDF image file. After closing the image file, it's attached to an e-mail sent to the business unit and, optionally, to the user. The image file, now no longer needed, is deleted.

It's important to destroy the APToolkit object, lest you fill up the server's memory and cause a crash. The code to do it in CF v4.5 is <cfset PDF = "">; the same code will probably work in CFMX, although it might be preferable to use <cfset ReleaseComObject(PDF)>.

Figure 1: Screenshot of the PDF

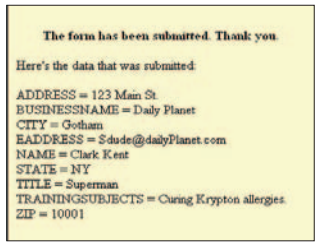


Figure 2: Running the example

Enhancing the Functionality

The APToolkit object has many more methods than those demonstrated here, giving you very granular control over the formatting of the PDF image file; they are well documented in the product's literature. (The documentation also explains the parameters seen in Listing 1.)

To eliminate the possibility of the business unit's Web editors messing around in the guts of the handler, MyBusinessForm.cfm's actions should be encapsulated in a custom tag or a CFC. Another enhancement, necessary in a world infested with malware, should be filtering out form values that might perform cross-site scripting and/or sql injection attacks.

About the Author

Tim Burton is the eGovernment Applications Architect for a large state agency in Oregon and has been writing CFML since 1998. This is his third career; he previously practiced medicine and made art (metal sculpture).

tim.burton@odot.state.or.us

Listing 1: MyBusinessForm.cfm

```
<!section 1 -->
<cfloop index="x" list="#form.fieldnames#">
    <cfif #x# is not "submit">
        <cfoutput>#x# = #form[x]#</cfoutput> <br> <!--- screen output
--->
    </cfif>
</cfloop>

<cfquery name="dbInsert" datasource="myDSN">
    insert into GMAtable
    (ADDRESS, BUSINESSNAME, CITY, EADDRESS,
    NAME, STATE, TITLE, TRAININGSUBJECTS, ZIP)
    values
    ('form.ADDRESS', 'form.BUSINESSNAME', 'form.CITY', 'form.EADDRESS',
    'form.NAME', 'form.STATE', 'form.TITLE', 'form.TRAININGSUBJECTS',
'form.ZIP')
</cfquery>


<! section 2 ------>
<!--- Instantiate the APToolkit object. --->
<CFOBJECT ACTION="Create" TYPE="COM" CLASS=APToolkit.Object NAME="PDF">

<cftry>
    <cfset u = PDF.OpenInputFile(ExpandPath("MyBusinessForm.pdf"))>
    <!--- Assign a handle to the image file, which is named with a
uniqueID --->
    <cfset strfilename = (ExpandPath(PDF.getuniquefilename(1)))>
    <cfset n = PDF.OpenOutputFile(#strfilename#)>

    <cfloop index="x" list="#form.fieldnames#">
        <cfif (#x# is not "submit")>
                <!--- calls object to assign form values: --->
                <cfset PDF.SetFormFieldData(#x#, #form[x]#, -998)>
        </cfif>
    </cfloop>

    <!--- Remove the reset and submit buttons --->
    <cfset PDF.DeleteFormField ("reset")>
    <cfset PDF.DeleteFormField ("submit")>

    <!--- Copy the values from the variables assigned above to the image
file --->
    <cfset u = PDF.CopyForm(0,0)>

    <!--- Close the image file  --->
    <cfset u = PDF.CloseOutputFile()>

    <!--- Send the submitted form (image file) via email attachment --->
    <cfmail
        to="tim.burton@odot.state.or.us"
        from="WEBMAILERR@odot.state.or.us"
        subject="New PDF form was submitted."
        mimeattach = #strfilename#>
    </cfmail>

    <!--- If the user wants a copy of the PDF by filling in a form field
with
    their own email address (the form field must be named 'EADDRESS'):   -
-->
    <cfif IsDefined("Form.EADDRESS")>
        <cfmail
                to="#Form.EADDRESS#"
                from="WEBMAILERR@odot.state.or.us"
                subject="Copy of your PDF form."
                mimeattach = #strfilename#>
        </cfmail>
    </cfif>

    <!--- Delete the constructed image file   --->
    <cffile action = "delete" file = "#strfilename#">

    <!--- Destroy the object in CF v4.5. --->
    <cfset PDF = "">

    <div align="center">
        <br><b>The form has been submitted.  Thank you.</b><br><br>
    </div>

    <cfcatch type="Any">
        <!--- error handling here--->
        <cfabort>
    </cfcatch>
</cftry>
```

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

ColdFusion U

For more information go to...

U.S.

Alabama
Huntsville
Huntsville, AL CFUG
www.nacfug.com

Alaska
Anchorage
Alaska Macromedia User Group
www.akmmug.org

Arizona
Phoenix
www.azcfug.org

Arizona
Tucson
www.tucsoncfug.org

California
San Francisco
Bay Area CFUG
www.bacfug.net

California
Riverside
Inland Empire CFUG
www.sccfug.org

California
EL Segundo
Los Amgeles CFUG
www.sccfug.org

California
Irvine
Orange County CFUG
www.sccfug.org

California
Davis
Sacramento, CA CFUG
www.saccfug.org

California
San Jose (temporary)
Silicon Valley CFUG
www.siliconvalleycfug.com

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

California
Long Beach
Southern California CFUG
www.sccfug.org

Colorado
Denver
Denver CFUG
www.denvercfug.org/

Delaware
Kennett Square
Wilmington CFUG
www.bvcfug.org/

Delaware
Laurel
Delmarva CFUG
www.delmarva-cfug.org

Florida
Jacksonville
Jacksonville, FL CFUG
www.jaxfusion.org/

Florida
Winter Springs
Gainesville, FL CFUG
www.gisfusion.com/

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

Florida
Tallahassee
Tallahassee, FL CFUG
www.tcfug.com/

Florida
Palm Harbor
Tampa, FL CFUG
www.tbmmug.org

Georgia
Atlanta
Atlanta, GA CFUG
www.acfug.org

Illinois
East Central
East Central Illinois CFUG
www.ecicfug.org/

Indiana
Avon
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana
Mishawaka
Northern Indiana CFUG
www.ninmug.org

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

Kentucky
Louisville
Louisville, KY CFUG
www.kymug.com/

Louisiana
Lafayette
Lafayette, LA MMUG
www.cflib.org/acadiana/

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

Maryland
Rockville
Maryland CFUG
www.cfug-md.org

Massachusetts
Quincy
Boston, MA CFUG
www.bostoncfug.com

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

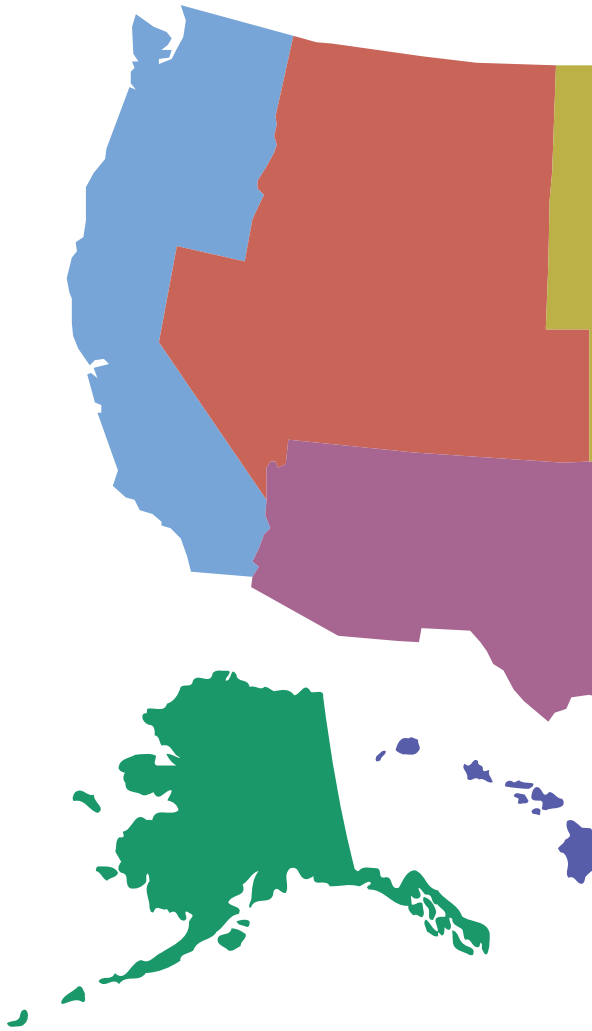
Minnesota
Brooklyn Park
Twin Cities CFUG
www.colderfusion.com

Missouri
Overland Park
Kansas City, MO CFUG
www.kcfusion.org

Missouri
O'Fallon
St. Louis, MO CFUG
www.stlmmug.com/

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

Nevada
Las Vegas
Las Vegas CFUG
www.sncfug.com/



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

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

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

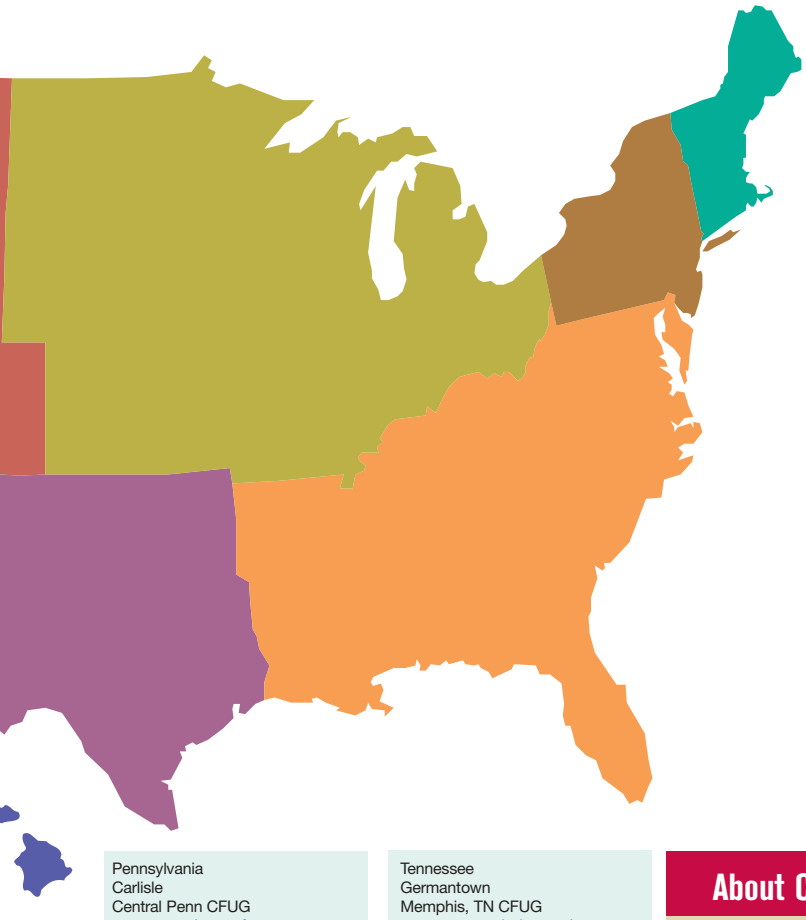
North Carolina
Raleigh
Raleigh, NC CFUG
www.ccfug.org

Ohio
Dayton
Greater Dayton CFUG
www.cfdayton.com

Oregon
Portland
Portland, OR CFUG
www.pdxcfug.org

User Groups

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



Pennsylvania
Carlisle
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Exton
Philadelphia, PA CFUG
www.phillycfug.org/

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

Rhode Island
Providence
Providence, RI CFUG
www.ricfug.com/www/meetings.cfm

Tennessee
LaVergne
Nashville, TN CFUG
www.ncfug.com

Tennessee
Germantown
Memphis, TN CFUG
www.mmug.mind-over-data.com

Texas
Austin
Austin, TX CFUG
www.cftexas.net/

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

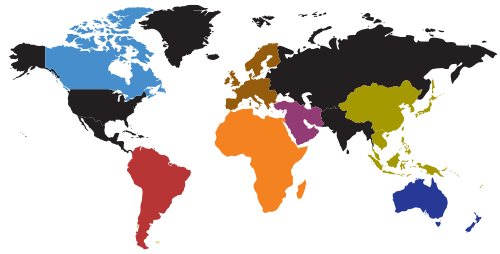
Texas
Houston
Houston Area CFUG
www.houcfug.org

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

About CFUGs

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

INTERNATIONAL



Australia
ACT CFUG
www.actcfug.com

Australia
Queensland CFUG
www.qld.cfug.org.au/

Australia
Southern Australia CFUG
www.cfug.org.au/

Australia
Victoria CFUG
www.cfcentral.com.au

Australia
Western Australia CFUG
www.cfugwa.com/

Brazil
Brasilia CFUG
www.cfugdf.com.br

Brazil
Rio de Janerio CFUG
www.cfugrio.com.br/

Brazil
Sao Paulo CFUG
www.cfugsp.com.br

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Ireland
Dublin, Ireland CFUG
www.mmug-dublin.com/

Italy
Italy CFUG
www.cfmentor.com

Japan
Japan CFUG
cfusion.itfrontier.co.jp/jcfug/jcfug.cfm

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
www.cfugspain.org

Switzerland
Swiss CFUG
www.swisscfug.org

Thailand
Bangkok, Thailand CFUG
thaicfug.tei.or.th/

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

User Configurable Reports Using Report Builder

A great tool for providing end users with well formatted, browser-independent reports

By Ryan Anklam

Delivering well-formatted reports through a web browser has always been a problem. Style sheets and cross browser compatibility are just a few of the many issues we, as developers, have to deal with when creating HTML based reports. Fortunately, with the release of CFMX 7 Macromedia has provided us with a great tool for providing end users with well formatted, browser independent reports: ColdFusion Report Builder.

Now that we have the Report Builder, life is perfect (at least when it comes to delivering reports via the web) – right? We have one tool to create a query and a perfectly formatted report in a matter of minutes. Well, let's not get too excited too fast –there are some limitations to using the report builder to build your queries and deliver your web-based reports. What if your database table or column names change after you've created a report? What if the query for your report changes, or worse yet, is wrong? What if each user wants to see specific subsets of data in a customized report? You don't want to go into the report builder and re-create your .cfr file (ColdFusion Report) every time your end users throw you a curveball.

One way to fix this problem would be to store your query for the report in a .cfm file, use the report builder to create a blank report and manually add the fields to the report, then use the report wizard to format your report. This solution would let you easily change the query that runs your report.

Now we can easily change the queries behind our reports, but what if an end user wants to change the report? What if a user would also want, for instance, last week's sales report, or a sales report for a specific customer? Writing a separate query for each report a user requests would quickly become a nightmare to maintain. Wouldn't it be convenient to let users configure their own reports?

Here's how you can do just that. By using a database to store the report's properties, a component to represent a Report object, ColdFusion Report Builder to build your report templates, and .cfm files to handle your display logic and manipulate the report object you can create reports that users can easily modify to their needs.

Building a Report

The first thing to do is to create the base reports. The base reports are the reports the user will select as the template for their user modified report. When a user customizes a report, they are actually cloning one of the base reports.

The first step in creating a base report is to use a SQL query builder tool to write a SQL statement to get the data for the report. For the first base report I'll create a Year-To-Date Sales by Salesperson report. The query is pretty straight forward, however, one thing to note is that I use table and column name aliases because I had columns from different tables that have identical names – after all, we want to take a look at a "real-life" situation. Here's my query:

```
SELECT
  C.name AS Customer
 ,P.name AS Product
 ,E.firstName + ' ' + E.lastName AS SalesPerson
```

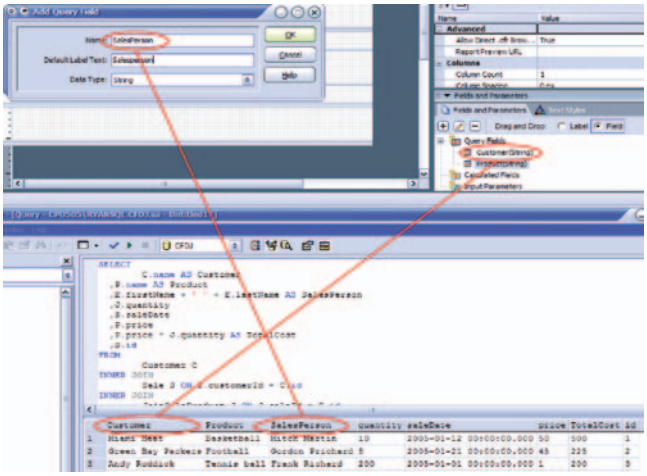


Figure 1: Adding Columns to the Blank Report

www.sys-con.com/edge2005

web services EDGE conference&expo

FALL SERIES

[Engage and Explore]

the technologies, solutions and applications that are driving today's **Web services** initiatives and strategies...

CALL FOR PAPERS NOW OPEN!*

Coming to a City Near You ►

Web Services Edge

Fall Conference Series

3 Dynamic Conference Programs

Targeting Major Industry Markets

20+ seminars within 5 tracks will address the hottest topics & issues:

- ► Web Services: The Benefits and Challenges
- ► Web Services Security
- ► SOA (Service-Oriented Architecture) and ESB (Enterprise Service Bus) Strategies
- ► Interoperability, Incremental Integration, & Open Source
- ► The Management Process in Developing a Web Services Strategy

Program Features:

- ► Keynotes
- ► Tutorials
- ► Panel Discussions

Why Attend:

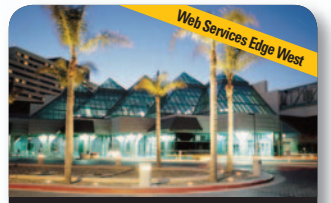
- ► Improve the return on your technology investment
- ► Develop & sharpen your strategy and identify key action steps
- ► Find new ways to reach and impress customers with Web services
- ► Maximize the power of your enterprise
- ► Protect your business from security threats
- ► Assess Web services as a viable option

Attention Exhibitors:

► An Exhibit-Forum will display leading Web services products, services, and solutions

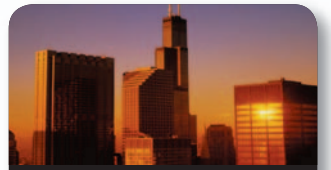
The Westin Washington, D.C.
Washington, D.C.
September 7-8, 2005

1



The Westin Santa Clara Convention Center
Santa Clara, CA
October 24-25, 2005

2



Hyatt O'Hare Airport
Chicago, IL
Oct 31-Nov 1, 2005

3

Register Today! www.SYS-CON.com/Edge2005

Sponsored by

WebServices JOURNAL | XML JOURNAL | .NET developer's JOURNAL | eclipse developer's journal | WebSphere JOURNAL | information STORAGE+SECURITY journal | wldj | JDJ | LINUXWORLD MAGAZINE | MX developer's journal | asp.netPRO | SD Times | CoDe | Software Test & Performance

*Call for Papers email: grisha@sys-con.com

For Exhibit and Sponsorship Information ► Call 201 802-3066

Produced by SYS-CON EVENTS

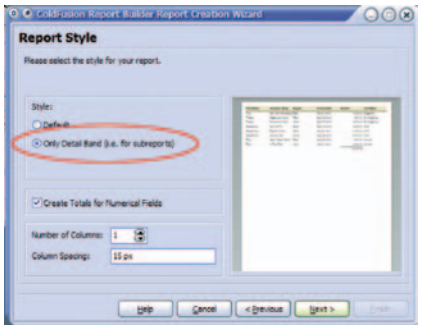


Figure 2: Report Style Screen

	Column Name	Data Type	Length
▶	id	int	4
	reportId	int	4
	name	varchar	50
	alias	varchar	50
	isString	bit	1

Figure 3: ReportColumn Table

Figure 4: Init method

```
 ,J.quantity
 ,S.saleDate
 ,P.price
 ,P.price * J.quantity AS TotalCost
 ,S.id
FROM
   Customer C
INNER JOIN
   Sale S ON S.customerId = C.id
INNER JOIN
   JoinSaleProduct J ON J.saleId = S.id
INNER JOIN
   Product P ON P.id = J.productId
INNER JOIN
   SalesPerson E ON E.id = S.salespersonId
WHERE
   DatePart(yyyy,s.saleDate) =
DatePart(yyyy,GetDate())
ORDER BY  S.id
```

Let's take a step back and look at the SQL for the base Year-To-Date Sales report. This works very nicely because there is no static date in the WHERE clause; however, it won't lend itself very easily to modification by the common user. Using a DatePart() statement isn't something many end users will be familiar. A user-modifiable version of the above query might look like this:

```
SELECT
  C.name AS Customer
 ,P.name AS Product
 ,E.firstName + ' ' + E.lastName AS SalesPerson
 ,J.quantity
 ,S.saleDate
 ,P.price
 ,P.price * J.quantity AS TotalCost
 ,S.id
FROM
   Customer C
INNER JOIN
   Sale S ON S.customerId = C.id
INNER JOIN
   JoinSaleProduct J ON J.saleId = S.id
INNER JOIN
   Product P ON P.id = J.productId
INNER JOIN
   SalesPerson E ON E.id = S.salespersonId
WHERE
   s.saleDate >= '01/01/2005'
AND
   s.saleDate <= '12/31/2005'
ORDER BY  S.id
```

Although I lose the dynamic properties of the first report by using static date values in the values within my "WHERE" clause it becomes much easier for a user to change this report to fit their needs. And, since users can modify this report, it still won't need to be changed by a developer when the next year comes around.

Now that I have my report query written I'll open the Report Builder and build my report. The first step in building my report is to create a new blank report. After I have my blank report I need to manually add my columns to the report. To do this I'll run a query analyzer application side by side with the report builder to make sure that the columns I am manually adding match the Query exactly (see Figure 1).

After I add all the columns in my query I'll use the Report Wizard to build the report layout. This can be found under the *Report -> Report Wizard for this report* menu and can be run anytime. Doing this will overwrite any layout you are currently working on. After the report is built using the wizard, I create any calculated fields for the report. In this case I'll create two calculated fields, one for the total sales for a sales person, and another for the grand total of sales. I'll also create an input parameter for the name of the report with the default value of "Year To Date Sales by Salesperson." This will allow users to specify a new name for their customized report which will be passed in at runtime. The next step is to replace the report title the wizard generated with the input parameter. The pre-defined styles come in quite handy here. Instead of choosing a font, font-size, and style, I'll set the "Report Title" pre-defined style which does all three steps in one.

After the Year To Date Sales report is finished, I'll create another baseline report: Total Sales by Product. For this report I'd like to see a summary count of all the sales for each product. In addition, I'd like to see how many of each product each salesperson is selling. In order to do this I'll have to create two queries. A red flag should have just gone off in your head. The Report Builder does not let us pass in two queries but it does let us create *subreports*.

Subreports are just what the name implies: they are a report within a report. A subreport is an entirely separate ColdFusion report file that is embedded within another report and is the best way to create a report that will need more than one query.

Before I create my subreport I still have to manually add the columns to the main report using the SQL statement that's used to get the data for the report just like I did for my Year-To-Date Sales report. Since I'm going to use a subreport, I'll need to add the product id field to the column list. This column won't be displayed on the report, but its value will be passed to the subreport.

To add a subreport go to the *Insert -> Subreport* menu. Here you have the

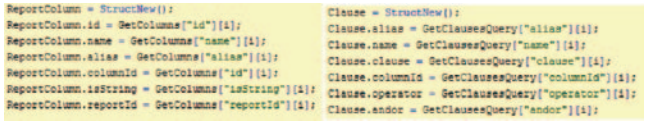


Figure 5: ReportColumn & Clause Structures

option to use an existing subreport, or to create a new one using the report wizard.

For this example I'll create a new report for the subreport. Since this is a subreport and I'm not going to let users modify this I'll use the Query Builder to create the query for this report. Taking the same approach I did before I'll write my SQL Query in a SQL Editing IDE and paste the SQL into the Query Builder.

```
SELECT
  P.firstName + ' ' + P.lastName AS SalesPerson
 ,SUM(J.quantity) AS Total
FROM
    Salesperson P
INNER JOIN
    Sale S ON S.salespersonId = P.id
INNER JOIN
    JoinSaleProduct J ON J.saleId = S.id
WHERE
    P.ProductId = 1
GROUP BY
P.firstName + ' ' + P.lastName
```

I'm not done with the query yet, I'll have to replace my static product ID with the current product from the main report. The modified SQL will look like this:

```
SELECT
  P.firstName + ' ' + P.lastName AS SalesPerson
 ,SUM(J.quantity) AS Total
FROM
    Salesperson P
INNER JOIN
    Sale S ON S.salespersonId = P.id
INNER JOIN
    JoinSaleProduct J ON J.saleId = S.id
WHERE
    P.ProductId = #param.id#
GROUP BY
P.firstName + ' ' + P.lastName
```

I'll save this query and use the default values for the Report Grouping and Report Layout screens. On the Report Style screen be sure to select the "Only Title Band" option (Figure 3). This will remove the header from the report making it fit nicely into the main report.

The next screen to pay attention to is the Subreport Binding screen (see Figure 5). This screen is where I am going to bind the #param.id# variable from our SQL query to the main report. In the Parameter Name field I enter the value "id." The value drop down will gives me a list of columns defined in the main report. Since the I am looking for is the product id, I'll select that from the dropdown list as well. The next screen is the Subreport filename screen. Since this is a subreport of the Total Sales by Product report I'll call this totalSalesByProduct_subreport.cfr. The next and final screen is the Finish subreport setup screen, click "Finish" to add the subreport to the report. Since this subreport is just a report, it is possible to make this user configurable as well, however for this article I'll stick to configuring the main reports only.

That's all the work I'll have to do in the Report Builder for this report. This report file will be the same file used for each report that is created by customizing a base report. I'll save this report to a Reports folder in my web root folder and make sure that I have the name of the file written down somewhere (I'll need it to populate the database later).

Now that I have my first base report done, I want to provide my users the means to modify the output of the report. To accomplish this I need to do a few things:

1. Give the user a list of reports that they can modify.
2. Give the user friendly names for the columns used in the report.
3. Build the WHERE clause dynamically.
4. Give the user ability to apply AND/OR operators within the WHERE clause
5. Give the user the ability to add multiple WHERE clauses to a single column.
6. Make it easy for users to create and save their own reports.

To accomplish this it's clear that I'll need to create a database to store and manage the data for the reports. A database driven approach will give me the flexibility I need to create user modifiable reports. Based on the criteria I'll need to create a Report table to store unique information for each report, a Report Column table to store the column names and a Report Column Clause table to store the data that will be used to build the dynamic WHERE clause.

Creating the Database

The Report table needs to store the name of the report, a description for each report (I like to enter a question in this field to help users identify the report), the SQL statement used to get data from the report, the location of the report template file, and whether or not this report is a baseline report. Remember, the baseline reports can only be cloned and not updated or deleted.

Another key point to make here is that when entering the SQL statement into this table *the entire WHERE clause* should be replaced with a place holder string. I'll demonstrate this a little later when I populate the tables.

The Report Column table stores the column name exactly as it was entered in the SELECT statement of the SQL Query, a friendly name for the column, and a boolean value that specifies

Figure 6: Main Report Screen

whether data type for this column is a string or not. This field determines whether to add single quotes to the data when populating the ReportColumnClause table.

The ReportColumnClause table holds the operator for the clause, the data for the clause itself, and whether the clause is an AND/OR statement. Be careful when entering data in this table because the first clause entered for each report must have a blank AND/OR field in order to parse correctly.

Creating the Report Component

After the database is done I need to create a component to control the report. I'll need to store the report id, name, description, SQL string, template file location, a list of columns, a list of clauses, a boolean value specifying if this is a baseline report, and the DSN for this report in private component level variables. I'll create and initialize them in my pseudo constructor: the Init() method (see Figure 4).

The init will have two arguments: id and DSN. The id is not required and is defaulted to 0. If we are creating an instance of an existing report an id will be passed in, if we are creating an empty instance of this component no id, or an id of 0 is passed in. If the id is not 0 I know this is an instance of a report and I'll call the GetReportInformation() function to retrieve the report data from the database.

This component needs public functions to create a new report, save the current report to the database, get a list of columns available to the report from the database, get all the clauses used to build the report, add a new column to the columns table (used to generate a new column id by cloning an existing column), add a new clause to the database, get a list of all available reports, and return the dynamic query that will be passed into the database.

The columns available to the report are stored in a structure that emulates the database structure. The clauses used to build the WHERE statement are stored the same way (see Figure 5).

The last step in building the component is to expose the private component level variables by writing getter and setter methods. Since the template file, SQL string, column array, clause array, and baseline variables are all read only, only getter methods should be written for them. The column and clause arrays are read only since they will be modified one element at a time using the AddColumn and AddClause functions.

Putting It All Together

Now that I have the entire framework complete it's time to start putting it all together. Initially, I need to manually populate the database with my base report information. The first table I'll populate is the Report table. When I insert the SQL string into the database, I'll replace the entire WHERE clause with the ~CLAUSE placeholder (Figure 8). After I enter the report, I'll need to write the id of the newly created report down.

```
SELECT
  C.name AS Customer
 ,P.name AS Product
 ,E.firstName + ' ' + E.lastName AS SalesPerson
 ,J.quantity
 ,S.saleDate
 ,P.price
 ,P.price * J.quantity AS TotalCost
```

```
<cffunction name="Init" access="public" returntype="void" hint="Constructor for the Priority class." output="no">
    <cfargument name="id" type="numeric" required="yes" default="0">
    <cfargument name="dsn" type="string" required="yes" default="#application.dsn#">
    <cfscript>
        variables.id = arguments.id;
        variables.dsn = arguments.dsn;
        variables.name = "";
        variables.description = "";
        variables.sqlString = "";
        variables.templateFile = "";
        variables.ReportColumnArray = "";
        variables.ReportClauseArray = "";
        variables.isBaseline = false;

        //if the id isn't 0 then this is an actual report, get the data from the database and set the class fields
        if(variables.id neq 0)
        {
            GetReportInformation();
        }
    </cfscript>
</cffunction>
```

Figure 7: Pairing the old column id with the new column id

```
for(x = 1; x lte ArrayLen(columnIdArray); x = x + 1)
{
    if(columnIdArray[x][1] eq form["column" & i])
    {
        newId = columnIdArray[x][2];
        break;
    }
}

//add the clause to the new report
NewReport.AddClause(andor
                    ,newId
                    ,form["operator" &i]
                    ,form["clause" & i]);
```

Figure 8: Adding clauses to the new report

```
 ,S.id
FROM
    Customer C
INNER JOIN
    Sale S ON S.customerId = C.id
INNER JOIN
    JoinSaleProduct J ON J.saleId = S.id
INNER JOIN
    Product P ON P.id = J.productId
INNER JOIN
    SalesPerson E ON E.id = S.salespersonId
~CLAUSE
ORDER BY  S.id
```

Next, I'll enter the columns for the report in the ReportColumn table keeping in mind that they must be entered exactly as they appear in the SELECT statement of my query. This is where I'll use the id I wrote down when inserting data into the Report table. For fields that are concatenated such as firstName + ' ' + lastName AS Name the entire statement must be entered. The column friendly name can be anything that will be easy for the end users to understand.

To build the dynamic WHERE clause I'll enter each part of the WHERE clause into the ReportColumnClause table with the corresponding ID value from the ReportColumn table. The order the records are entered into this table is essential. The first record entered will be the first WHERE clause and so forth. This is especially important when entering OR operators. If they are not entered in order you will not get the results you expect from the query.

The main report page is simply a listing of all the reports in the Report table with a link to customize each. It should be split up into two sections, baseline reports and customized reports. Since baseline reports can only be cloned I'll take away the edit and delete icons from their menu entirely.

When a user clicks on the clone report screen a few things need to happen behind the scenes before they are shown the customize report screen. First a report object must be created and initialized to represent the report being cloned. Next two arrays must be created, one for the columns in the report and another for the clauses of the report.

The customize report page should give the user the name of the report they are cloning, a field for the name of the new report, and a field for the report description. As I mentioned previously I like to give the user a question to describe the report so I labeled the field "Hint Question." Next I loop through the clause array and create fields for the user to modify the properties of the original report. Since these are the criteria that define the original report I'll only give the user the ability to change the values of the field. After the fields that define the original report are populated I'll give users a few more fields to customize themselves.

To add the new report to the database, two components must be created: a Report component that represents the original report and an empty Report component to store the modified report. Once that's done, since the SQL string itself is not being edited I copy the SQL string from the original report to the new report. Next, I loop through the columns of the original report and call the AddColumn() method of the new Report object to add the column to the new report. The AddColumn() method returns the id of the new column that is associated with the new report. I'll pair the new column id to the id of the column from the original report in a two dimensional array. This is important because we have not inserted the clauses for the new report yet. The clauses submitted in the form are still attached to the column id's of the original report. When I add the new clauses I want to use the column id's of the new report.

The last step is to loop through the clauses submitted and add the clauses to the database. First I need to loop through the columnIdArray to match the column id to the column submitted from the form and attach the new id to that column. Next I call the AddClause() method on the new report object making sure to insert the newly created column ids.

In Conclusion

Now every report created can be modified by each user to meet their needs. Of course a security model should be wrapped around this functionality to make sure only approved users can modify, edit, and delete reports. This can be taken a step further by implementing a role-based security model to define report data readers, report editors, or to limit report access to specific users.

After the initial time investment of teaching users to clone, edit, and delete reports, support calls for the creation of custom reports should be greatly reduced, with the added benefit that your reports will look great on screen and in print. To see this application in action visit www.innovacreative.com/cfdj/. The code for this application can be downloaded at www.innovacreative.com/cfdj/cfdj.zip.

About the Author

Ryan Anklam is Chief Information Officer and part owner of Innova Creative Media, Inc where he has been busy creating enterprise level hosted applications using an Object Orientated approach to Cold Fusion. He is a Microsoft Certified Professional and is an expert with Cold Fusion, SQL Server 2000, C#. Ryan has been developing web applications since 1996 and has been spending most of his waking hours playing with the new features in CFMX 7.

ryan@innovacreative.com

New Pre-Conference Classes
Beyond Basic SQL for CF - Nate Nelson
Intro to .NET for CFers - Geoff Snowman
Designing & Developing OO Applications w/ CFCs - Hal Helms
Maximizing Code ReUse w/ CFCs & Custom Tags - Simon Horwith
"Forms and Beans" applying simple OO principles to standard CF forms methodology - Joe Rinehart
Leveraging .Net Strengths from CFML - Charlie Arehart
Advanced Cascading Stylesheets - Sandra Clark
FLiP and Fusebox walkthrough - Jeff Peters

CFML Platform Integration
Deployment
Geoff Snowman
Simon Horwith
Tim Buntel
Charles Arehart
Christian Cantrell
Jeff Peters
Manager
Empowered Programming

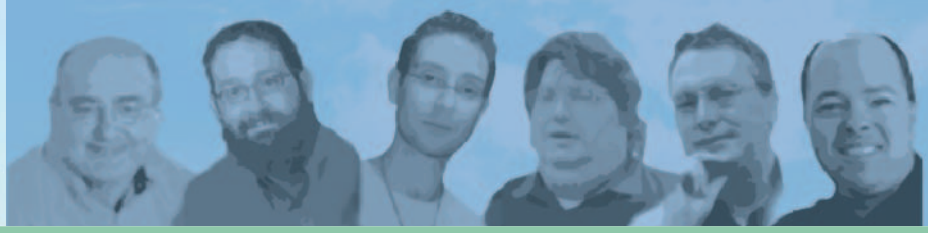
coldfusion

Raymond Camden
Steve Drucker
Michael Smith
Ben Forta
Sean Corfield
MX Integration
Advanced CF
Hal Helms
Dave Watts
Michael Dinowitz
Shlomy Gantz
Accessibility / Usability
Boot Camp

the premier coldfusion technical conference
CFUnited

June 29 - July 1, 2005
Washington DC Area
7th Annual ColdFusion Conference

3 full days!
New Venue
75% Bigger



www.cfunited.com

Produced by
TeraTech
Programming
www.teratech.com
301.424.3903
info@cfunited.com

macromedia
TeraTech
Microsoft
New Atlanta
CFDynamics
Fog Creek Software
Savvy
Interakt
Universal Mind
Paper Thin
MX developer's journal
ColdFusion Developer's Journal
HostMySite.com

CFUN has become the premier CF specific event, and Michael Smith and his team deserve all sorts of praise for their hard work in pulling it all off yet again.

Ben Forta

"...this event really is the best gathering in the world for people developing or managing CF systems. It's here that we can understand what happened, hear what's happening, and learn what's going to happen. You can't beat it."

Chuck Hoffman

"Great place to network yourself and pick up new techinques and ideas. Also to meet ones peers, and see what the future holds for all those involved with ColdFusion"

Daniel Gregorio

"Introductions to the latest developing techniques. Get inspiration in new ways to develop projects. Generally, to "re-kindle" your cf "fire" by being part of a group excited by and interested in cf development."

Kathleen Ballard


Intermedia.NET...
Now serving the hottest ColdFusion.
At Intermedia.NET we go beyond the industry standard by supporting the hottest new Coldfusion software, offering power like never before. For nearly a decade, we've been providing reliable, secure hosting to thousands of companies across the globe. We can do the same for you.
Intermedia.NET's premier hosting services include:
ColdFusion MX hosting
Competitive plans
Security sandboxes
Custom tag registration
Verity collections search engine
Guaranteed service levels
Unprecedented power, unmatched reputation...
Intermedia.NET is your hosting solution.
INTERMEDIA.NET
Call us at: 1.888.379.7729
e-mail us at: sales@intermedia.NET
Visit us at: www.intermedia.NET
© Copyright INTERMEDIA.NET, Inc 2005. All rights reserved. All other trademarks are property of their respective holders.