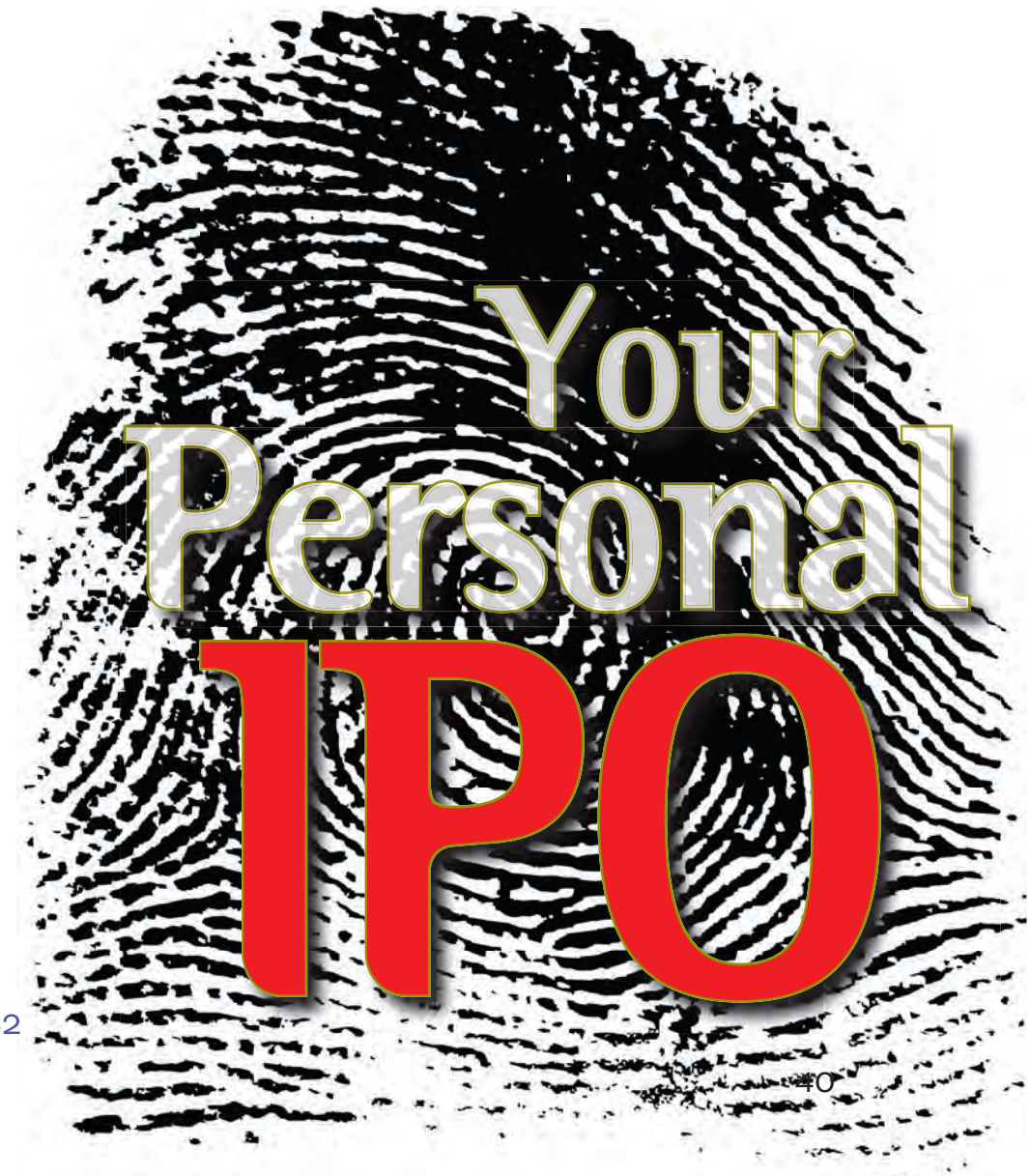
ColdFusion Developer's Journal

ColdFusionJournal.com
September 2006 Volume:8 Issue: 9



Your Personal IPO

40

ColdFusion Hosting is our Complete Focus

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

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

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

CFDynamics

866.233.9626 CFDYNAMICS.COM

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

macromedia ALLIANCE PARTNER

BlueDragon Certified V.I.P. Hosting Partner

For the greatest hits of the 70's, 80's and 90's call your web host's tech support.

For answers call us at 1-866-EDGEWEB
3 3 4 3 9 3 2

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

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

By the Numbers:

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

EDGE WEB HOSTING

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

http://edgewebhosting.net

CFDJ READERS' CHOICE AWARD — MXDJ READERS' CHOICE AWARD

2003 - 2006

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

ColdFusion Developer's Journal

editorial offices

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

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

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

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

Increase Your Productivity 100%

By Simon Horwith

Usually, when something sounds too good to be true, it probably is. We've all seen the spam – "Increase your <insert body part or bank account here> 100%" e-mails that are clearly nothing more than ridiculous claims with no validity.

Why does this month's editorial title sound like another "too good to be true" gimmick? Recently, I've noticed a trend in the growing popularity of productivity-enhancing software for ColdFusion developers. This is a trend I imagine will continue not only with third-party software but with future releases of ColdFusion as well. After all, rapid development, i.e., increased productivity, is what's made CF popular since its inception.

Before I continue, I will state for the record that I just invented the "100%" number in the title (I'd hate to think I contributed to innumeracy in any way). That said, there are some tools available that can save developers a lot of time and headaches. Some of these tools are relatively new and a few have been around for a little while; those that aren't new have recently been enhanced and all of them have recently been generating quite a buzz within the community I've used all of these utilities and thought I'd give a summary of what these utilities are as well as my experiences with them and an overview of when developers should be implementing them to help with development.

The first piece of software that ColdFusion developers should be aware of is Flex Builder 2. In addition to being a terrific IDE for Flex development, Flex Builder 2 also includes several ColdFusion enhancements for your Eclipse installation. These enhancements include:

- Support for RDS, which allows you to browse your server's file system and data sources from the IDE
- An RDS CRUD wizard that generates a CFC with Create, Read, Update, and Delete methods for working with a database table of your choosing
- A services browser that allows you to browse CFC and Web Services metadata

For those CF developers who work with Flex, Flex Builder 2 also adds wizards for creating CFCs from ActionScript classes and vice versa (used when passing Transfer Objects between the Flex 2 Gateway), a wizard for creating all of the code necessary for both the Flex UI and server-side logic (CFCs) for a Flex application based on identifying the database tables and data that the application will use, and a wizard for quickly creating CFCs to be used by the Data Management feature in Flex Data Services. These wizards are a big time-saver for developers building Flex applications that talk to ColdFusion, and having the ability to access RDS services from Eclipse is terrific for developers who want to see what

About the Author

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

CF_UNDERGROUND VIII PRE-MAX EVENT

DATE: SUNDAY, OCTOBER 22, 2006

LOCATION: THE BEACH, LAS VEGAS

365 Convention Center Drive, Las Vegas, Nevada

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

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

WWW.CF-UNDERGROUND.COM

only $99

TeraTech
www.teratech.com

ColdFusion Developer's Journal

SYS-CON MEDIA

founder & ceo
Fuat Kircaali fuat@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
Robyn Forma robyn@sys-con.com

advertising manager
Megan Mussa megan@sys-con.com

associate sales manager
Kerry Mealia kerry@sys-con.com
Lauren Orsi lauren@sys-con.com

sys-con events

president, events
Carmen Gonzalez carmen@sys-con.com

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

customer relations

circulation service coordinator
Edna Earle Russell edna@sys-con.com

sys-con.com

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

web designers
Stephen Kilmurray stephen@sys-con.com
Paula Zagari paula@sys-con.com

accounting

financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

accounts receivable
Gail Naples gailn@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

resources are available on a remote server. You can find out more about the CF Extensions in Flex Builder 2 at http://www.adobe.com/products/coldfusion/flex2/.

Another useful piece of software that's getting a lot of attention is FusionDebug, an Eclipse plug-in (it will work with CFEclipse and/or Flex Builder 2) that allows you to:

- View all the variables/scope on your pages
- Step into, over, and through your code, including custom tags and ColdFusion Components and method calls
- Set watch expressions and breakpoints
- Examine the SQL being executed in your queries as well as the data being returned
- Really see what's going on under the hood (stack traces of page execution and details of all of the variables in memory) using a "Java Detail Mode"

Other than installing the plug-in to Eclipse, you only need to turn on JVM debugging on the CF Server(s) that you want to debug – these can be local or remote servers. FusionDebug is an excellent tool for developers who are looking to review, optimize, and/or troubleshoot ColdFusion code, and is a very useful testing tool as well. You can find more information about FusionDebug at http://www.fusion-reactor.com/fusiondebug/.

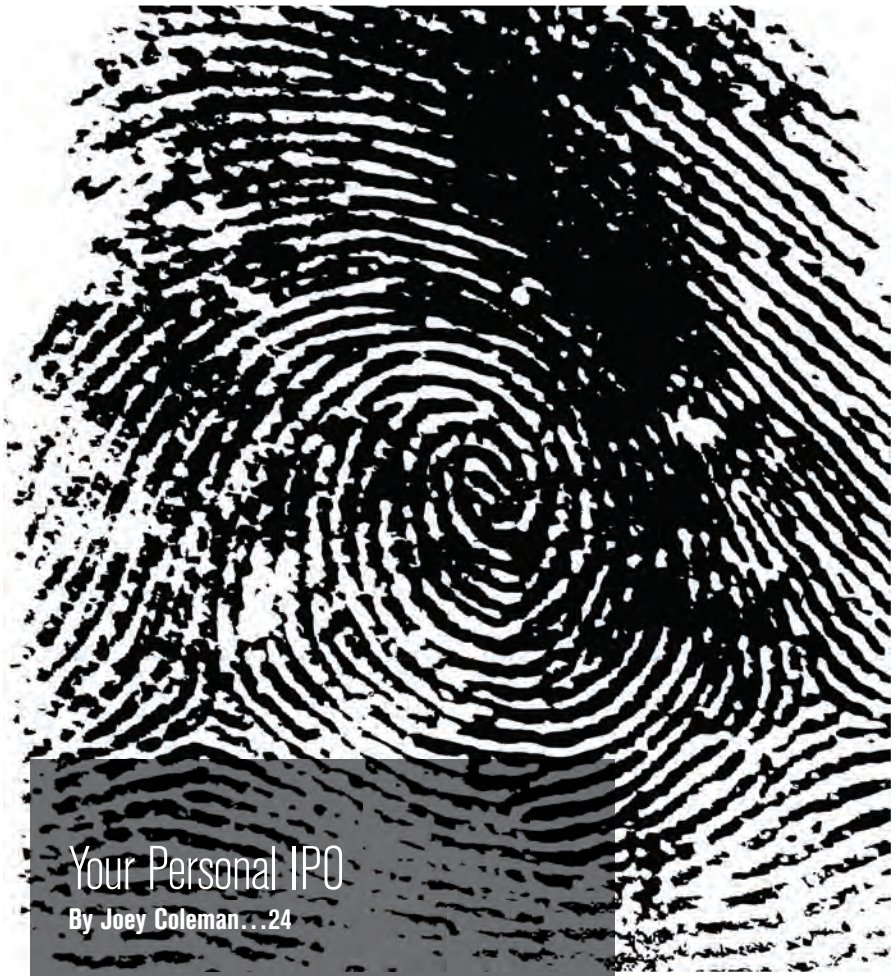
If FusionDebug is the be-all and end-all for troubleshooting and debugging ColdFusion code, SeeFusion would have to be its counterpart for debugging and troubleshooting your server. SeeFusion is a Java application that "hooks-in" to your ColdFusion server and allows developers and server administrators to view crucial information about server activity (via a Web interface), in real time.

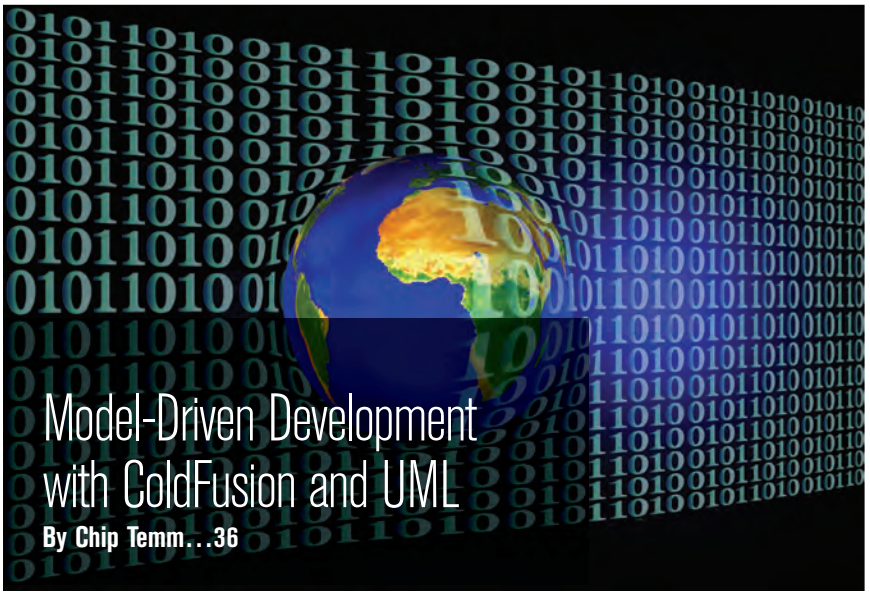
You can "kill" specific requests (threads), force garbage collection, and do a lot more. In a nutshell, SeeFusion shows you pretty much everything you'd want to see about what's happening on your server. It's useful for testing code in a development environment prior to launching (use a stress tool like OpenSTA to generate the load) and for troubleshooting a "live" production server that's experiencing problems. There are both a standard and enterprise version; among other things the enterprise version has a Flex 2 dashboard interface (with nice status bars for resource usage and datagrids for viewing request information) and comes with all of the JDBC driver wrappers for monitoring queries. Like FusionDebug, SeeFusion is easy to install – just tweak a few configuration files on your CF server in order to make SeeFusion work, namely in to "wrap" it around the CF servlet engine and JDBC drivers. Find out more about SeeFusion at www.seefusion.com/.

The bottom line in determining whether you need these products is this: if you're never going to do any Flex 2 development and you don't need or use RDS for browsing CFCs, Web services, the server file system, and the registered data sources on a server, then you don't need Flex Builder 2. If you never troubleshoot or fix ColdFusion server configurations and/or CF server performance/health, you don't need SeeFusion. If you never have performance problems with your code, never review other people's code, and never get any errors in your code (yeah, right), then you don't need FusionDebug. Even if any of that last set of criteria does apply to you, if you feel comfortable debugging CF code or you prefer not to use the typical interactive debugging features that FusionDebug offers, you don't need to use it. If, on the other hand, any of the scenarios I mentioned here do apply to you, then by all means I strongly recommend these tools. If you already have a preferred CF IDE (Dreamweaver, Homesite+, etc.) there's no critical need to make the switch to CFEclipse. That said, if you're comfortable with CFEclipse, using other Eclipse plug-ins will be a walk in the park. Flex Builder 2 and FusionDebug are only two plug-ins – there are literally hundreds of Eclipse plug-ins (many are free) for everything from working with databases to diagramming with UML to integrating with source control systems. Also, CFEclipse is free, so there's no cost involved in giving it a try, other than investing a little of your time to playing with it.

While I have always maintained that better tools will never replace better developers, there's no denying that better tools do help…especially in the hands of better developers. For the first time in a long time, ColdFusion developers can get excited about something other than the cool new features in the latest release of CF. I recommend taking a look at one or more of these software utilities and making them part of your "ColdFusion Developer Arsenal." If nothing else, they will offer a welcome distraction until the anticipated release of ColdFusion 8..

TABLE OF CONTENTS





Taking a First Look at FusionDebug

The first interactive debugger for ColdFusion MX

By Jeffry Houser

I remember a particularly long weekend sitting in a computer lab for 12 hours and trying to write an assembler program on a VAX machine that would read and write files. (A VAX is a big archaic mainframe computer.)

It took many days, but I was finally able to provide successful results. Sometimes, I think we take for granted that ColdFusion makes our lives extremely easy. (Even a novice CF developer should be able to figure out how to read and write files in under five minutes.) One tool that was integral to my success in the project was the use of an interactive debugger. In my professional career, I've used one for writing Pascal and Lotus Notes, but, moving to the Web world, debuggers were noticeably vacant. That is until now. The folks who brought us Fusion Reactor have now brought us FusionDebug, the first interactive debugger for ColdFusion MX.

Why Do You Need an Interactive Debugger?

What is an interactive debugger? It is a tool that allows you to view the results of your code, line by line. You can see the value of variables, queries, CFC Instances, and change the values of variables on the fly. If you slip back into the old days of your memory, you probably remember a product called ColdFusion Studio, right? Before CFEclipse, ColdFusion Studio was the tool that (almost) everyone used for creating ColdFusion applications. A little known (and even lesser used) feature in CF Studio was the availability of an interactive debugger. Unfortunately, the product was hard to configure and even harder to use, so it never became a staple of the ColdFusion developer's toolbox. Today, the landscape of ColdFusion development is much different than it was in the CF Studio days. In the old days, most templates did multiple things and business logic code was implemented right next to display logic. There was little thought of code encapsulation, and templates were written the same way they were processed (start at the top and work your way down). Although CFML custom tags and User-Defined Functions provided facilities for encapsulating code, it wasn't until ColdFusion Components were introduced that people started to apply advanced programming principles to their ColdFusion applications. Business logic is now put inside of CFCs. Frameworks such as Model-Glue and Mach-II help us separate business logic and presentation code. When a single ColdFusion page loads, it may be performing actions where the code is located across multiple files. It isn't always easy to find the root of your errors. FusionDebug is a tool that will help you find and correct your troublesome code.

I thought a good place to start might be to offer a definition of debugging concepts and definitions:

- ***Breakpoint:*** A breakpoint is a spot in your code where you want the debugger to stop, and can be any line with CFML code (tags, variables, and so on). After starting a debug session in FusionDebug, you'll load the page you want to debug in the browser and FusionDebug will intercept it at your first breakpoint. (It can also intercept requests from pages and CFCs called from Flex, AJAX, Flash Remoting, and Web services.) If there are no breakpoints, the page will run as normal. Breakpoints do not have to be in the page you request, they can be located in CFCs, custom tags, includes, or UDFs.
- ***Variables:*** You already know what a variable is and how to use them. (If not, read "Creating Variables in CFML" from CFDJ, Vol. 6, issue 2 [http://coldfusion.sys-con.com/read/43790.htm]). The variable list of FusionDebug shows all of the variables currently available to the template you are in, at the point where you're debugging. It will display local variables (the variables scope), URL, Form, request, CGI, and cookie variables, as well as shared-scope variables like session and application, and more.
- ***Expressions:*** In the past I've always referred to these as "watch variables." Expressions, in its simplistic form, are variables that you want to keep tabs on. However, the expression pane of FusionDebug will support any ColdFusion expression. Perhaps you want to keep tabs on what the first item of a list is, or the last element in an array? You can do that with the Expression tab.
- ***Debug:*** The debug command means to starts a debugging session. In other software I've seen this described as "run" or "execute."
- ***Step Into:*** A step is how you move from one line of code to the next. There are three variations of the Step depending on what you want to do. Step Into is probably what you'll use

most of the time. It means to step over this line of code, but jump into any functions, custom tags, or CFC method calls that are called from the line of code you're on. FusionDebug will open the file that you step into and stop at the first line of CFML code.

- ***Step Over:*** Step Over is the exact opposite of Step Into. It will execute the code in any function, custom tag, or CFC Method without opening it. It moves right onto the next line. It's a good idea to favor Step Over even on tags and expressions that wouldn't open a file, for reasons I'll explain later.
- ***Step Return:*** Step Return is used when you're inside a file that you stepped into. It will execute the rest of the file, and put you back to the place where the file was initially stepped into
- ***Resume:*** Resume means to continue execution of code until you run out of code, or find another breakpoint.
- ***Terminate:*** Terminate will end the debugging session without completing your code's execution. Usually I try to avoid using this.

These definitions should give you a good understanding of what features the debugger offers. I'll move on to the FusionDebug install, and then an example.

Installing FusionDebug

FusionDebug is created as an eclipse plug in. and you'll need Eclipse to install it. If you're already using CFEclipse or FlexBuilder, you must already have Eclipse installed. If not, you can download Eclipse from http://www.eclipse.org/. It's as simple as extracting a zip file and running the Eclipse command. There's no installer.

Speaking of CFEclipse, which is another plug-in for editing CFML code, while you don't need to use it with FusionDebug, if you want to try it, I strongly recommend reading the ACME (Apache, ColdFusion, MySQL, Eclipse) guide for detailed instructions (http://www.stephencollins.org/acme/) on the install. Even if you don't use Apache or MySQL look at just the final chapter on CFEclipse. It's the best place to find detailed installation instructions on CFEclipse.

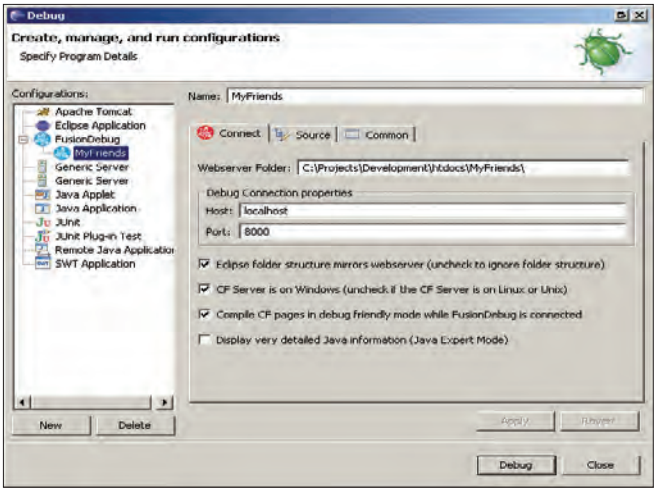


Figure 1: FusionDebug Configuration Window

There are two ways to install FusionDebug: using a manual process or an automated one. The automated process uses the "Find and Install Updates" feature built into Eclipse. This is documented in the FusionDebug user guide, which (at the time of this writing) is available at http://www.fusion-reactor.com/fusiondebug/helpDocs/FusionDebug_User_Guide.pdf. I'm going to step you through the process of the manual install, so while you're at it you may as well download the files for a manual install. The zip archive includes the PDF documentation at "plugins\com.intergral.fusionreactor.debug.core_1.0.0\FusionDebug – User manual.pdf". No matter which way you install, I recommend reading through the guide.

Plugins for Eclipse are located in a subdirectory of your Eclipse installation named plugins. You can unzip the download to the plugins directory if you wish. Restart Eclipse so the plugin is found. If you don't wish to extract into your Eclipse directory, unzip somewhere else, and copy the directories manually. There are two directories you need to put in "eclipseinstall\plugins": "com.intergral.fusionreactor.debug.core_1.0.0" and "com.intergral.fusionreactor.debug.ui_1.0.0". Restart Eclipse and you should be good to go. Yes, the install really is as easy as copying files.

Configuring ColdFusion and FusionDebug

Your next step is to set up ColdFusion to allow for debugging requests. In reality, you're setting up the Java server, which underlies your ColdFusion installation, and not making any direct changes to ColdFusion settings. You need to change the arguments that are used when launching the JVM. In most ColdFusion installations, you'll be using JRun, and the JVM settings are located in a jvm.config file. For Windows installations, this file is located in the CF install directory under "runtime/bin". For Unix, it's just in "bin" under your CF install directory. If you are using CF in any sort of multiserver configuration, your location may vary slightly. Before editing the file, you may want to save a copy of the current file, in case anything goes wrong with the one-line edit I'm about to describe.

Find the "java.args" line of the config file and remove this argument:

```
-XX:+UseParallelGC
```

Then add these options:

```
-Djava.compiler=NONE
-Xnoagent
-Xdebug
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000
```

Make sure that all the options are on the same line. For ease, you can copy these lines out of the FusionDebug documentation. Note the port 8000 that's used. If you know that port to be used is already in your system, choose a number that's not in use. I'll explain its use shortly. Save the file and restart your ColdFusion services. If it fails to start, revert to the saved copy from above and study your changes to make sure you followed the instructions carefully.

The final step in getting yourself set up is to create an

instance of FusionDebug that points to the server you set up for debugging:

1. Load up Eclipse, and select a project you want to debug with. (If you don't have any Eclipse projects, see the FusionDebug User Guide for more information on creating one for the first time.) Select "Debug" from the Run menu.
2. Select FusionDebug from the menu and Click New to create a new FusionDebug instance. This will bring up a dialog similar to Figure 1.
3. Enter a name for the FusionDebug instance. This can be anything you want it to be. Specify the Web server folder. I used the same folder as the project root for my Eclipse Project.
4. Specify the host and port. Note that the hostname can be localhost or indeed any server you have access to, which has been configured as per the steps above. Yes, FusionDebug can debug remote servers. The port is not your Web server port, but the port you specified in the Java arguments. It is 8000 by default. If you followed my instructions above, that didn't change.
5. Select your other options. In my case, my CF Server is on Windows, I want CF to compile pages in debug friendly mode, and my Eclipse folder mirrors the Web server. I have not yet experimented with the "Detailed Java Information" option, soI left that unchecked. These are explained in the User Guide.

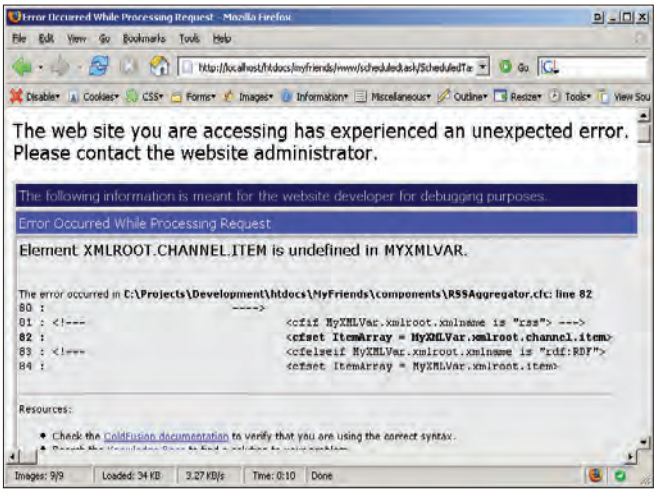


Figure 2: MyFriend error

Now you are good to go, ready to debug.

You're First Debugging Session

If you've read my two articles on creating an RSS Aggregator, you probably noticed that I found a bug between part one and part two. The code I wrote in part 1 (http://coldfusion.sys-con.com/read/235976.htm) supported only RSS 2. It turns out that the Macromedia XML News Aggregator (http://weblogs.macromedia.com/mxna/) was only providing an RSS 1.0 feed. The XML was different in each feed. In part 2 (http://coldfusion.sys-con.com/read/264745.htm), I described the error and explained the fix. I thought I'd step you through the process I used to diagnose and fix the problem (without FusionDebug) and then step you through the process using FusionDebug.

In my test database for the MyFriend RSS aggregator, I have two RSS URLs. One points to my blog and the other to MXNA. When trying to process the MXNA URL (by manually loading the scheduled task), I was seeing an error, as shown in Figure 2. I wasn't sure why I was receiving the error, since all RSS feeds should contain an item as part of the channel (right?). The first thing I did was add a cfdump tag to see why "xmlroot.channel.item" did not exist. I reloaded and got nothing [insert two head scratches and one look of utter confusion here]. Why wasn't I seeing any debug output? It turns out the CFFUNCTION had its OUTPUT attribute set to false. No output means no output and the cfdumps weren't being sent back to the browser. I changed that attribute to true and reloaded. Okay, now with the cfdump I could drill down into the XML and see the problem. Items (AKA Blog Posts) are stored differently in RSS 1 vs RSS 2. And the root elements are different.

Where would the debugger have helped? To debug this code inside a component, I'm changing code. It's one thing to add a variable output or cfdumps so you can view variables. Those are easy enough to comment out later. It's quite another to have to change code, such as the output parameter from false to true. I don't want to have different code for "production" versus "development." That defeats the purpose and allows too much margin for error. How many of these output attributes have I forgotten to change back? I have no idea. The problem gets worse when you are dealing with nested components, because you have trickle back up to the top of the tree and set all outputs to true. Still another situation occurs with code within CFSILENT, which I would have to find and remove to see any debugging output. These are situations where an interactive debugger offers benefit. To get to the root of the problem I don't have to change any code.

This is how I'll debug this in FusionDebug:

1. First, I'm going to open up the project in Eclipse, and switch to the debug perspective. You can switch perspectives using "Window --> Open Perspective and selecting "Debug." You should see something similar to Figure 3. (Again, all this is described very well in the User Guide, for those new to Eclipse.)
2. Next, start a debugging session. You can re-open the debug window (see Figure 2), select your Debug instance, and select "Debug," or you can start the debug session by clicking the "debug" button from the toolbar. You should see the session started up in the debug window (top left).
3. Open up a page in the project that you want to debug, and set a breakpoint. (Note that if you open a file from the file system rather than from a project, the debugger won't enable you to step through code.) With MyFriend, I am opening up the scheduled task file (schedultedtask/scheduledtask.cfm) and putting a breakpoint on the first CF line of the page. You can add a breakpoint by selecting the line

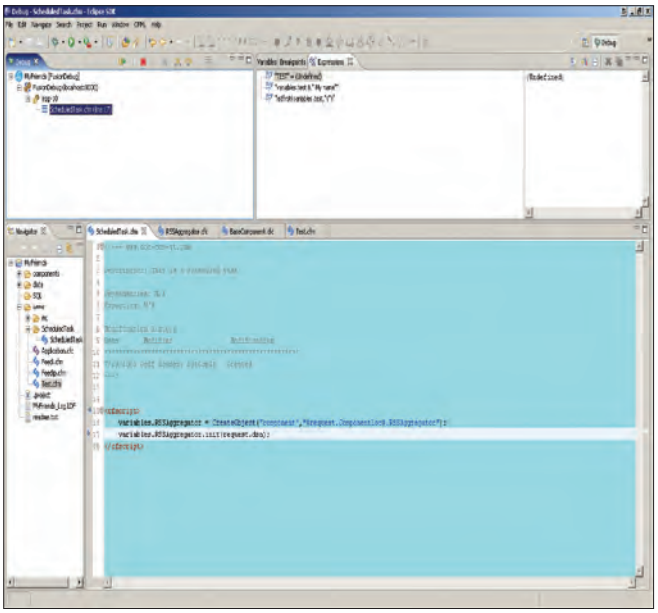


Figure 3: FusionDebug Perspective

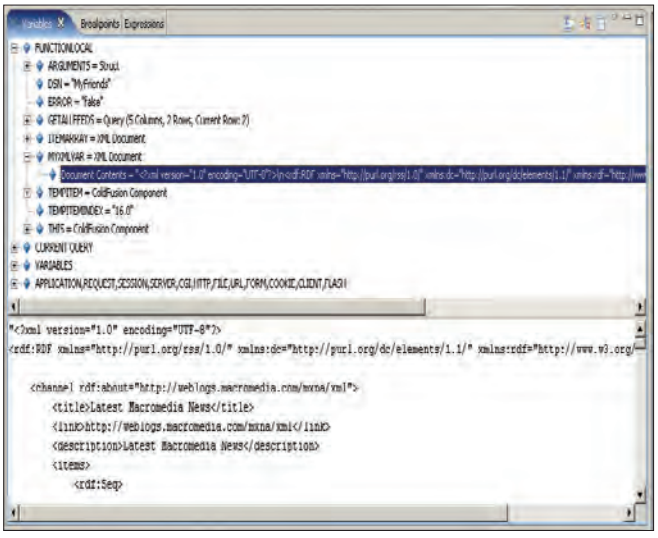


Figure 4: Variables Panel

and choosing "Window --> Toggle Line Breakpoint" or using Control + Shift + B, or right-click the line and choose "Toggle Breakpoint." You'll now see the breakpoint was set because of the blue dot in the grey bar to the left of the line number. (You can see the dot on line 15 of Figure 3).

4. Open your favorite browser and launch the page you want to debug. FusionDebug will intercept the page at the first breakpoint. Again, note that you're not opening the page "in" the debugger but in whatever browser you want. There is a blue arrow in the same grey bar to show you which template you are currently running. (This is called the "Current Instruction Pointer.")
5. Click Step Over and you can step through the initialization code to watch the component being created. Continue to click Step Over until you reach the init method of the RSSAggregator component. Around line 75 you should see code like this:

```
<cfset MyXMLVar = xmlparse(cfhttp.filecontent)>
```

6. Click past the line that parses the XML content. Now take a look at the variables tab shown in Figure 4. It should be in the upper-right corner of the screen. You can view all the variables available to the component. The one we care about most is the local function variable that holds the XML. The first time through the loop, you'll see that the XML is in proper RSS 2 format. The second time through you'll notice it uses "RDF" instead of XML. Drilling down, you'll realize that the items are different in the RDF format than they are in the RSS format. And that is the root of the problem.

The bug was easily solved by adding an if statement (explained in the previous article) to decide which RSS format to use, and how to access the item array. Both debugging methods can bring you to the same end result, but I personally find the use of FusionDebug to be much more elegant.

I mentioned near the start to favor Step Over versus Step Into, even on tags or function calls that don't open a file. The reason is that being a Java debugger, Eclipse will try to step through the underlying Java code. FusionDebug is configured to hide that by default, but the execution of the tag/function will take longer than if you'd used "Step Over."

What Next

The folks who created FusionDebug created some Captivate videos to help demonstrate what FusionDebug can do and some uses of it. It is one thing to read about how to do things, it is another to see it done, so I suggest checking them out at http://www.fusion-reactor.com/fusiondebug/gettingStarted.html. Charlie Arehart wrote a tips and tricks article for this edition of ColdFusion Developer's Journal. I've had a chance to preview it and it contains a lot of information beyond the scope of this article.

The product is available as a free 20-day trial. Pricing starts at US$299, with an 10% discount currently available with the code CFCOMMUNITY. Volume discounts are also available. The company offers free support, at support@fusion-reactor.com.

The RSSAggregator code is downloadable from my blog at www.jeffryhouser.com. I appreciate all the feedback I've received from it. Thanks for reading, and let me know what you think. I'll see you next month.

About the Author

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

jeff@instantcoldfusion.com

Meet Robert

A business executive at a popular social media site

Challenges

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

Solutions

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

Providing End-to-End Solutions for Your Business

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

VitalStream®

Digital Media Solutions for Your Business

Tools for Team Management

By Jeff Peters

At this year's CFUnited, I gave a talk titled "Supercharging Fusebox Project Management." As indicated by the title, that presentation was aimed at managers who use Fusebox. While I am a major proponent of Fusebox, this article deals with some aspects of team management, regardless of what framework is used. We'll take a look at some basic management goals, and how we can use simple tools to ease the task of managing development in a team environment.

The first point I'd like to make is that this is not an article about using version control. I only have one thing to say about version control if you're in a team environment: use it. No matter whose arm you have to twist to get a version control system set up, do it and do it right now. Your time is too valuable to waste on code lost to oversaves.

When it comes to project management, we're concerned about two things: time and money. Since, in the final analysis, money is just the way we count how much time it's taking to do the work, we'll focus on the time aspect of things in this article. When dealing with time, a manager is concerned about three major questions:

- What are we going to do?
- What have we done?
- What did we do?

The first question has to do with planning the application. Unless there is a plan on the table for what will be done, it is impossible to answer any questions about progress. Effective project management depends on planning

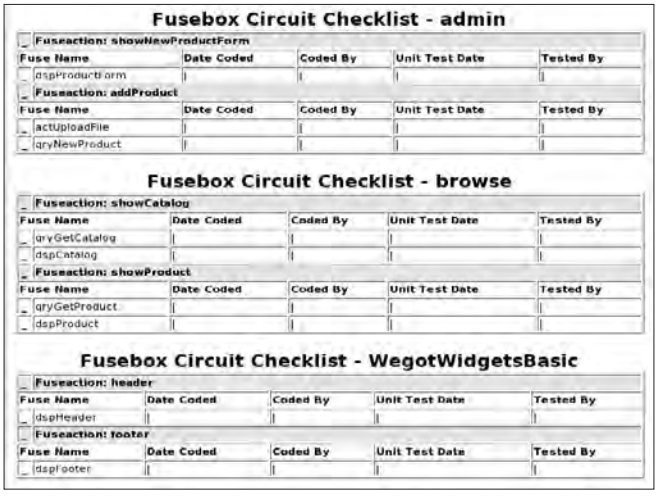
The second question addresses the progress of work performed once the coding work has begun. With a development plan in place, a manager wants to know how much of the plan has been executed, and how much remains to be done. There needs to be a way to monitor what work has been accomplished at any given time.

The third question involves documenting the project, both for technical reasons and contractual satisfaction. With strong system documentation, a customer is far more likely to be satisfied with the work that has been done. A manager would like to have a way to create strong documentation once the project has been completed.

To answer the first question, we need to have a well-designed plan for the application. This is where I will mention Fusebox, or, more accurately, the Fusebox Lifecycle Process (FLiP). FLiP uses a planning process that includes construction of a wireframe (business process), prototype (front-end), and architectural design, as steps in its methodology. The result of these three steps of the process is a framework that has a configuration file for each circuit (directory). That file is named circuit.xml, and Listing 1 shows an example. Note: If you're interested in the rest of the FLiP process, there is a free FLiP Roadmap PDF file that you can download at www.GrokFusebox.com.

It is important to note here that every template required in the circuit is expressed in the circuit.xml file as an <include> element. Like the CFML <cfinclude> tag, the <include> element specifies the filename with the template attribute. So, within these circuit.xml files, we have the entire plan for all the templates that need to be coded for the application.

This is just the way FLiP handles the process of planning and configuring an application. As mentioned previously, I'm not



Fusebox Circuit Checklist - admin

Fuseaction: showNewProductForm				
Fuse Name	Date Coded	Coded By	Unit Test Date	Tested By
dspProductForm				
Fuseaction: addProduct				
Fuse Name	Date Coded	Coded By	Unit Test Date	Tested By
actUploadFile				
qryNewProduct				

Fusebox Circuit Checklist - browse

Fuseaction: showCatalog				
Fuse Name	Date Coded	Coded By	Unit Test Date	Tested By
qryGetCatalog				
dspCatalog				
Fuseaction: showProduct				
Fuse Name	Date Coded	Coded By	Unit Test Date	Tested By
qryGetProduct				
dspProduct				

Fusebox Circuit Checklist - WegotWidgetsBasic

Fuseaction: header				
Fuse Name	Date Coded	Coded By	Unit Test Date	Tested By
dspHeader				
Fuseaction: footer				
Fuse Name	Date Coded	Coded By	Unit Test Date	Tested By
dspFooter				

Figure 1

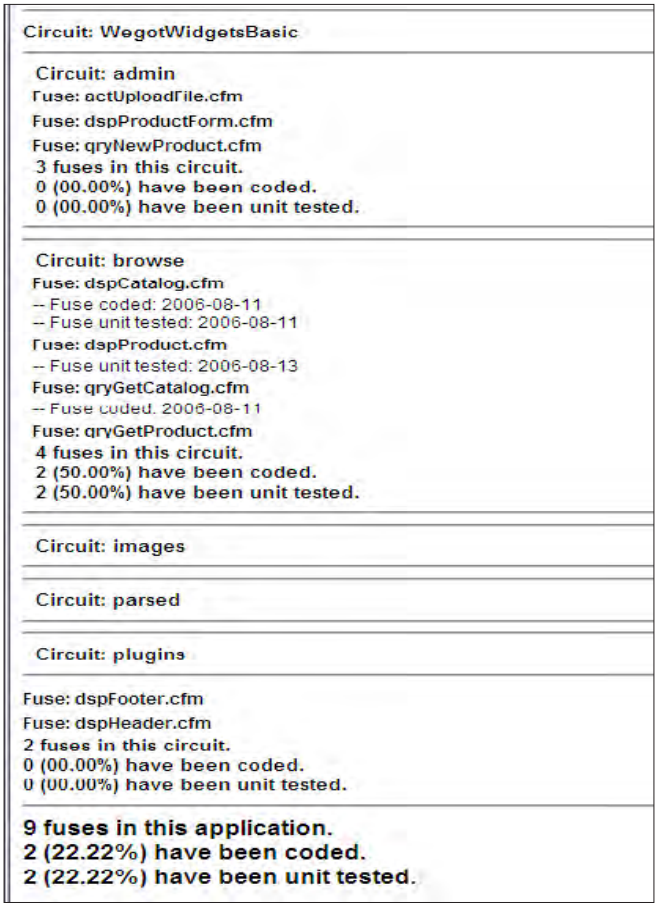


Figure 2

saying you must use Fusebox in order to manage projects well, but you must have some sort of coherent plan, and a way to inspect that plan. If you're not using FLiP, you can still infer from this example the type of thinking that will let you build your own tools to accomplish the same things we're doing in this article. At any rate, due to the well-organized manner in which the application is planned out, we can use a CF-based tool to give us a leg up on managing the development. The tool I'd like to introduce at this point is called FB4Checklister.

FB4Checklister leverages the fact that we have all this data in circuit.xml files. XML gives us the opportunity to use CF's XML manipulation capabilities to provide easy reports based on that data. All FB4Checklister does is cruise through an application's directory tree, reading each circuit.xml file and transforming it into HTML output. The name is taken from the fact that the first stylesheet produced for the tool created a coding checklist for all the fuses (templates) in the application. The output from that stylesheet is shown in Figure 1. In case you're interested, the (obviously very small) application used here is the Wegot Widgets Basic reference application from fusebox.org.

FB4Checklister (using its default XSL stylesheet) generates a checklist we can use to check off each template as it gets coded, tested, etc. We can print out the checklist, or copy and paste it into a spreadsheet; use it however it might benefit our process. Even better, we might consider writing our own XSL stylesheet to create output that is most effective for our process. If project management software is in use, perhaps we could create a format that is compatible for import into that software to save a lot of time not typing in all those tasks. Listing 2 shows the checklist XSL stylesheet. Although this article can't provide an XSL how-to, the format for this stylesheet is basic and can be used as a jumping-off point for other stylesheets.

With the first question answered, we move into the part of the project where the coding is being done, and, as managers, we become interested in the second question, "What has been done?" We might also think about this question as, "How much has been done?"

One way to answer this question at regular intervals throughout the coding phase is to use the checklist we generated with FB4Checklister. In fact, that's how I used the checklist for several projects after I first wrote the tool. However, if the project is organized, we can take advantage of another XML vocabulary to make answering the question an automated job.

The second XML vocabulary we'll look at is Fusedoc. Although it was invented as part of the Fusebox mindset, Fusedoc is not directly related to Fusebox. It is just an XML vocabulary for describing the responsibilities, properties, and I/O of a template. You can use it quite well even if you don't use Fusebox or FLiP. Details on Fusedoc and the Fusedoc DTD are available at www.HalHelms.com.

Listing 3 is a Fusedoc for a template that displays a product to the user. The responsibilities section contains a plain language, first-person description of what the template does. The io section contains elements that describe the inputs and outputs for the template, but it is the properties section that will be most useful for the purposes of this article. The properties section can contain history, note, and property elements. A property element has a name and a value, so we can store just about any properties of the template we want using this method. In the properties section in Listing 3, there are two properties: dateCoded and dateUnitTested. These are arbitrary properties I've invented to use for project management purposes. A team member, after coding the template, also adds the dateCoded property. Similarly, when the template passes unit

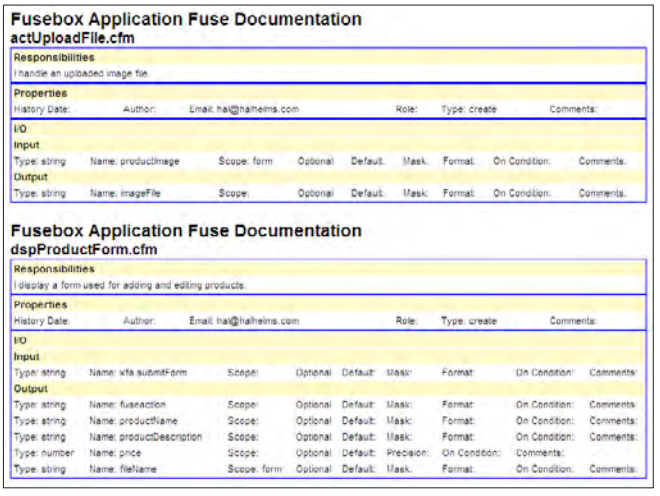


Figure 3

testing, a team member adds the dateUnitTested property.

With these properties in place, we can use another automated tool to check on the project's progress whenever we like. I've put together a simple example of such a tool, named FB4CodingProgress, which is shown in Listings 4-6. When this tool is run in the root directory, it produces an HTML file with the report shown in Figure 2.

For each circuit, we get a list of templates (fuses), and if the fuse has been coded (or coded and tested) a note is shown along with the date from the Fusedoc. Any nested directories are indented in the report. At the end of each directory's (circuit's) section, there is a set of totals for the directory showing percentages coded and tested, and a grand total section appears at the bottom of the report. We are able to grab an instant status report for the coding and testing of the project, without even bothering any of the coding staff.

So, finally, on to the last question, "What was done?" We want to answer this question with system documentation. This is typically one of the most overlooked parts of application development, usually because we coders don't put much emphasis on documentation. However, using techniques we've already seen, we can take the pain out of creating documentation for the code that exists in our finished application.

Once again, we're relying on a tool to take the pain out of the process. This time, it's a tool called FusedocMiner, which is somewhat like FB4CodingProgress, but not nearly as specialized. All it does is cruise through a directory tree, pull out all the Fusedocs, and apply an XSL stylesheet to each one. The code is shown in Listing 7. The default XSL stylesheet (Listing 8) produces HTML that, along with the default CSS stylesheet (Listing 9), renders some nice documentation ready for printing, as shown in Figure 3. When printed, each template gets its own page in the documentation, so it makes a nice bound document for delivery to the customer.

Unfortunately, there's not enough space in this article to dissect the XSL stylesheets used in FusedocMiner and FB4CodingProgress. They are both fairly simple examples of XSL, and I hope they'll serve as good starting points if you haven't already explored XSL. Both the XSL stylesheets and the CSS stylesheets are provided so you can modify them to suit your local project management needs.

This article is really just a quick overview of the sorts of things that are possible in terms of project management when a structured and consistent process is used for development. One last time, I'll repeat that, although the examples used in this article are based on Fusedoc, there's no requirement to use Fusebox or the Fusebox Lifecycle Process when using Fusedoc; I just happen to have a lot of Fusebox-based examples lying around. Even if you want to use your own XML-based template documentation standard, all the techniques would remain the same. It's important not to lose the point of the management techniques in the irrelevant fact that the samples use Fusedoc.

With a bit of thought and a few tools, it's possible to have very

good project management with minimal tedium and, even more importantly, low impact on team members' time. All the tools I've shown in this article are available for free download at www.GrokFusebox.com, in the Grok's Goodies section. As with all my tools, they're free for use and modification. Use them to feed your own ideas and let me know how you've made them better.

About the Author

Jeff Peters is a program manager and application architect for Operational Technologies Services in Vienna, Virginia. His ColdFusion-related books are available at www.protonarts.com; e-mail to jeff@grokfusebox.com.

jeff@grokfusebox.com

Listing 1:Sample circuit.xml

```
<circuit access="public">
-
 <fuseaction name="showCatalog">
<xfa name="lnkProductName" value="browse.showProduct"/>
<xfa name="lnkAdmin" value="Admin.showNewProductForm"/>
<include template="qryGetCatalog"/>
<include template="dspCatalog"/>
</fuseaction>
-
 <fuseaction name="showProduct">
<xfa name="lnkBackToCatalog" value="browse.showCatalog"/>
<include template="qryGetProduct"/>
<include template="dspProduct"/>
</fuseaction>
</circuit>
```

Listing 2: FB4CircuitChecklist.xsl

```
<xsl:stylesheet version="1.0">
<xsl:output method="html" doctype-public="-//W3C//DTD HTML 4.0 Transi-
tional//EN"/>
-
 <xsl:template match="/">
-
 <html>
-
 <head>
<title>Fusebox Project Management Checklist</title>
-
 <style>

       th { font-family: Arial, Helvetica, sans-serif;
            font-size: 8pt;
           font-weight: 700; }
      td { font-family: Arial, Helvetica, sans-serif;
            font-size: 8pt;
           font-weight: 400; }

</style>
</head>
-
 <body>
-
 <div align="center" style="font-family: Arial;
font-weight: 700;                    font-size: 14pt;">

     Fusebox Circuit Checklist - @@@[Circuit Directory]

</div>
<xsl:apply-templates select="circuit"/>
<br style="page-break-after:always;"/>
</body>
</html>
</xsl:template>
-
 <xsl:template match="circuit">
-
 <table border="1" align="center" width="100%">
<xsl:apply-templates select="fuseaction"/>
</table>
</xsl:template>
-
 <xsl:template match="fuseaction">
-
 <tr bgcolor="#e0e0e0">
<th width="10">
    –
    </th>
-
 <th align="left" colspan="5">

     Fuseaction:
<xsl:value-of select="@name"/>
</th>
</tr>
-
 <tr>
<th align="left" colspan="2">
     Fuse Name
    </th>
<th align="left">
     Date Coded
    </th>
<th align="left">
     Coded By
    </th>
<th align="left">
     Unit Test Date
    </th>
<th align="left">
     Tested By
    </th>
</tr>
<xsl:apply-templates select="include"/>
</xsl:template>
-
 <xsl:template match="include">
-
 <tr>
<th width="10">

    </th>
-
 <td align="left">
<xsl:value-of select="@template"/>
</td>
<td align="left">
     |
    </td>
<td align="left">
     |
    </td>
<td align="left">
     |
    </td>
<td align="left">
     |
    </td>
</tr>
</xsl:template>
</xsl:stylesheet>
```

Listing 3: Sample Fusedoc

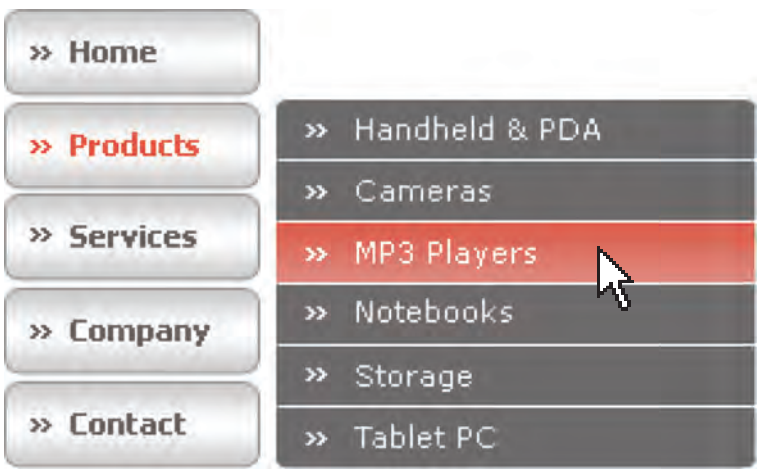
```
<fusedoc fuse="dspProduct.cfm" version="2.0" language="ColdFusion">
<responsibilities>
    I display details for the specified product.
 </responsibilities>
-
 <properties>
<history type="create" email="jeff@grokfusebox.com"/>
<property name="dateCoded" value="2006-08-12"/>
<property name="dateUnitTested" value="2006-08-13"/>
</properties>
-
 <io>
-
 <in>
```

The new MX CSS Menus2 for Dreamweaver

Create and customize CSS menus in Dreamweaver !

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

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



Buy at only $99

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

Simplifying Web Development

Interakt

```
<string name="xfa.lnkBackToCatalog"/>
-
 <recordset name="qryProductDetails">
<number name="productID" precision="integer"/>
<string name="productName"/>
<string name="productDescription"/>
<string name="imageFile"/>
<number name="price"/>
</recordset>
</in>
-
 <out>
<number name="productID" oncondition="XFA.lnkProductName"/>
</out>
</io>
</fusedoc>
```

Listing 4: FB4CodingProgress.cfm

```
<!---
<fusedoc fuse="FB4CodingProgress.cfm" language="ColdFusion" ver-
sion="2.0">
  <responsibilites>
    I read an application's fuse files, starting at the current direc-
tory,
    and produce a coding progress report in the specified output file.
  </responsibilities>
  <properties>
    <history date="11 Aug 2006" author="Jeff Peters"
email="jeff@grokfusebox.com" type="create">
      Version 1.0
    </history>
  </properties>
  <io>
    <in>
      <string name="thisDir" scope="attributes" optional="true"
default="current path" />
      <boolean name="recurse" scope="attributes" optional="true"
default="yes" />
        <string name="fusePrefixes" optional="true"
default="dsp,act,qry" />
        <string name="outputFile" optional="true"
default="#attributes.thisDir#FB4CodingProgressOutput.html" />
        <string name="indenter" optional="true" default="" />
    </in>
    <out>
    </out>
  </io>
</fusedoc>
--->
<cfparam name="attributes.thisDir" default="#GetDirectoryFromPath(GetC
urrentTemplatePath())#">
<cfparam name="attributes.xslFile" default="#attributes.thisDir#FB4Co
dingProgress.xsl">
<cfset slash = "/">
<cfif Find("\",attributes.thisDir)>
  <cfset slash = "\">
</cfif>
<cfparam name="attributes.recurse" default="yes">
<cfparam name="attributes.fusePrefixes" default="dsp,act,qry">
<cfparam name="attributes.outputFile" default="#attributes.thisDir#FB4
CodingProgressOutput.html">
<cfparam name="attributes.indenter" default="">
<cfparam name="attributes.topLevel" default="Yes">
<cfset codedCount = 0>
<cfset unitTestedCount = 0>
<cfset totalCount = 0>
<cfparam name="request.grandCodedCount" default="0">
<cfparam name="request.grandUnitTestedCount" default="0">
<cfparam name="request.grandTotalCount" default="0">

<cffile action="append"
              file="#attributes.outputFile#"
           output="<hr /><span id=""circuitName"">#attributes.
indenter#Circuit: #ListLast(attributes.thisDir,slash)#</span><br />"
     addnewline="true">

<!--- Read directory entries into array --->
<cfdirectory name="qryThisDir"
          action="LIST"
       directory="#attributes.thisDir#">

<!--- For each entry in files array: --->
<cfloop query="qryThisDir">

  <!--- Select based on entry type: --->
  <cfswitch expression="#Trim(qryThisDir.type)#">
    <cfcase value="File">
      <!--- If it's a fuse file, transform it and write it to the
output file --->
      <cfif ListFindNoCase(attributes.fusePrefixes,Left(qryThisDir.
Name,3))>
        <cfset totalCount = totalCount + 1>
        <cfset dirName = ListLast(attributes.thisDir,"\")>
        <cffile action="read"
                file="#attributes.thisDir#\#qryThisDir.Name#"
           variable="fuseFile">
          <!--- Strip out the Fusedoc --->
          <cfset inFusedoc = false>
          <cfset fusedocText = "">
          <cfloop from="1" to="#ListLen(fuseFile,Chr(10))#"
index="i">
              <cfif FindNoCase("<fusedoc",ListGetAt(fuseFile,i,Chr(10
))>
                  <cfset inFusedoc = true>
              </cfif>
              <cfif inFusedoc>
                  <cfset fusedocText = fusedocText & ListGetAt(fuseF
ile,i,Chr(10))>
              </cfif>
              <cfif FindNoCase("</fusedoc",ListGetAt(fuseFile,i,Chr(1
0))>
                  <cfset inFusedoc = false>
              </cfif>
          </cfloop>
        <cffile action="read"
                   file="#attributes.xslFile#"
              variable="xslCode">
          <cftry>
         <cfset newHTML = XMLTransform(fusedocText, xslCode)>
         <cfset newHTML = Replace(newHTML,"<hr />","<hr />#attri-
butes.indenter#","ALL")>
          <cfset newHTML = Replace(newHTML,"Fuse:","#attributes.
indenter#Fuse:","ALL")>
          <cfset newHTML = Replace(newHTML,"--","#attributes.in-
denter#--","ALL")>
              <cfif FindNoCase("coded",newHTML)>
                <cfset codedCount = codedCount + 1>
              </cfif>
              <cfif FindNoCase("unit",newHTML)>
                <cfset unitTestedCount = unitTestedCount + 1>
              </cfif>

              <cffile action="append"
                         file="#attributes.outputFile#"
                         output="#newHTML#"
                    addnewline="true">
         <cfcatch type="Expression">
           <cfoutput><li><b>Error in #qryThisDir.Name#:</b> #cfcatch.
detail#<br /></li></cfoutput>
           <cfflush>
         </cfcatch>
       </cftry>
      </cfif>
    </cfcase>

    <cfcase value="Dir">
      </ol>
      <h4>Reading <cfoutput>#qryThisDir.Name#</cfoutput></h4>
      <ol>
      <!--- Prevent recursion of the entire hard drive --->
      <cfif not ListFind("..,..",qryThisDir.Name) AND attributes.
recurse>
        <!--- If directory entry, recurse with main   --->
        <cf_FB4CodingProgress thisDir="#attributes.thisDir#\
#qryThisDir.Name#"
                              recurse="#attributes.recurse#"
                              xslFile="#attributes.xslFile#"
                          fusePrefixes="#attributes.fusePrefixes#"
                            outputFile="#attributes.outputFile#"
                              indenter="#attributes.
indenter#  "
                              topLevel="No">
      </cfif>
    </cfcase>
    <cfdefaultcase>
      No handler for #qryThisDir.type#.
    </cfdefaultcase>
  </cfswitch>
```

```
</cfloop>

<cfset outStr = "<div id=""totals"">#attributes.indenter##totalCount#
fuses in this circuit.<br />" >
<cfif totalCount GT 0>
 <cffile action="append"
                 file="#attributes.outputFile#"
               output="#outStr#"
        addnewline="true">
 <cfset outStr = "#attributes.indenter##codedCount# (#NumberFormat(co
dedCount/totalCount*100,"00.00")#%) have been coded.<br />" >
 <cffile action="append"
                 file="#attributes.outputFile#"
               output="#outStr#"
        addnewline="true">
 <cfset outStr = "#attributes.indenter##unitTestedCount# (#Numbe
rFormat(unitTestedCount/totalCount*100,"00.00")#%) have been unit
tested.<br />">
 <cffile action="append"
                 file="#attributes.outputFile#"
               output="#outStr#</div>"
        addnewline="true">
 <cfset request.grandCodedCount = request.grandCodedCount + coded-
Count>
 <cfset request.grandUnitTestedCount = request.grandUnitTestedCount +
unitTestedCount>
 <cfset request.grandTotalCount = request.grandTotalCount + totalC-
ount>
</cfif>
<cffile action="append"
                 file="#attributes.outputFile#"
            output="<hr />"
     addnewline="true">

<cfif attributes.topLevel EQ "Yes">
  <cfset outStr = "<div id=""grandTotals"">#attributes.
indenter##request.grandTotalCount# fuses in this application.<br />" >
 <cfif request.grandTotalCount GT 0>
     <cffile action="append"
                     file="#attributes.outputFile#"
                   output="#outStr#"
            addnewline="true">
     <cfset outStr = "#attributes.indenter##request.grandCoded-
Count# (#NumberFormat(request.grandCodedCount/request.grandTotalC-
ount*100,"00.00")#%) have been coded.<br />" >
     <cffile action="append"
                     file="#attributes.outputFile#"
                   output="#outStr#"
            addnewline="true">
     <cfset outStr = "#attributes.indenter##request.grandUnitTested-
Count# (#NumberFormat(request.grandUnitTestedCount/request.grandTo-
talCount*100,"00.00")#%) have been unit tested.<br />">
     <cffile action="append"
                     file="#attributes.outputFile#"
                   output="#outStr#</div>"
            addnewline="true">
 </cfif>
</cfif>
```

Listing 5: FB4CodingProgress.xsl

```
<xsl:stylesheet version="1.0">
<xsl:output method="html" doctype-public="-//W3C//DTD HTML 4.0 Tran-
sitional//EN"/>
-
 <xsl:template match="/">
-
 <html>
-
 <head>
<title>Fusebox Project Management Checklist</title>
-
 <style>

       th { font-family: Arial, Helvetica, sans-serif;
            font-size: 8pt;
          font-weight: 700; }
      td { font-family: Arial, Helvetica, sans-serif;
            font-size: 8pt;
          font-weight: 400; }

</style>
</head>
-
 <body>
-
 <div align="center" style="font-family: Arial;
font-weight: 700;                    font-size: 14pt;">

      Fusebox Circuit Checklist - @@@[Circuit Directory]

</div>
<xsl:apply-templates select="circuit"/>
<br style="page-break-after:always;"/>
</body>
</html>
</xsl:template>
-
 <xsl:template match="circuit">
-
 <table border="1" align="center" width="100%">
<xsl:apply-templates select="fuseaction"/>
</table>
</xsl:template>
-
 <xsl:template match="fuseaction">
-
 <tr bgcolor="#e0e0e0">
<th width="10">

   </th>
-
 <th align="left" colspan="5">

      Fuseaction:
<xsl:value-of select="@name"/>
</th>
</tr>
-
 <tr>
<th align="left" colspan="2">
      Fuse Name
    </th>
<th align="left">
      Date Coded
    </th>
<th align="left">
      Coded By
    </th>
<th align="left">
      Unit Test Date
    </th>
<th align="left">
      Tested By
    </th>
</tr>
<xsl:apply-templates select="include"/>
</xsl:template>
-
 <xsl:template match="include">
-
 <tr>
<th width="10">

   </th>
-
 <td align="left">
<xsl:value-of select="@template"/>
</td>
<td align="left">
      |
    </td>
<td align="left">
      |
    </td>
<td align="left">
      |
    </td>
<td align="left">
      |
    </td>
</tr>
</xsl:template>
</xsl:stylesheet>
```

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

Your Personal IPO

How to publicize and promote yourself to employers and project managers

By Joey Coleman

While the IPO market on Wall Street has cooled substantially, there is a new IPO looming on the horizon that has the potential to make you millions of dollars in the coming years.

This is not hype. This is not a pipe dream. This is not a fantasy. It is a reality that is yours for the taking.

If your stockbroker told you about this, you'd do everything you could to get in on the deal. If a friend worked for this IPO-poised venture, you would try to get hired to get your options before the big day of "going public."

The good news is that you already have an inside track on the hottest IPO of the next decade.

It's you.

Twenty years ago, the acronym IPO meant little to people outside of Wall Street and the investment banking industries. In the late 1990s, IPOs came to center stage – turning young college students into multimillionaires in a matter of minutes. Fortunes were won or lost in the blink of an eye and everyone wanted to be in on the next big IPO.

In the decade to come, people will begin to focus on their own IPOs – the initial public offering where they present themselves to industries, the marketplace, and even the world in an attempt to build long-lasting, personally gratifying careers. The first steps in developing your own personal brand and taking it public to the world are outlined below in an easy-to-accomplish, step-by-step format that promises to bring you a more fulfilling work experience in future years. Whether you work in-house for a large development company or spend your days freelancing – developing your personal brand will drastically help your career.

The Personal Brand

People are always talking about "brands." Soft drinks have brands, cars have brands, computers have brands, schools have brands, even countries have brands.

But do people have brands?

Absolutely.

Your personal brand is the image created in the minds of people when they connect with your name, your work product, or your service offering. It's your reputation. It's the experience people have when they interact with you. It's the expectation people have when they learn that you will be part of their team going forward.

Your personal life experience, personal point-of-view, and the way you are perceived by others, combine to make your personal brand.

Your personal brand is yours. It's unique to you. It's mobile – you take it with you no matter where you go. It's unique – your individual life experience made you who you are today. It's valuable – your earning power, your position in your industry, and your ability to find the next great project all hinge on your personal brand.

Much has been written about the celebrity personal brand. Images of Michael Jordan, Oprah Winfrey, Tiger Woods, Bill Clinton, David Beckham, Donald Trump, and LeBron James dominate any discussion of the value a strong personal brand can have on a career. Each of these individuals has spent countless hours developing personas and their respective career trajectories reflect this personal study and focused effort.

The Significance Of Your Personal Brand

According to the most recent findings by the U.S. Bureau of Labor and Statistics, the average American will have 10 different jobs before reaching 40 years of age. In short, that means you will be working in a new position, for a new boss or company, on an average of once every two to three years. That's a lot of changes in a short amount of time!

Given the frequency of job changes, the maintenance and development of your personal brand is vital. Gone are the days of employers helping to groom your career over time by providing a long-term employment ladder that you climb to retirement. Now, constant changes have employees shifting from project-to-project, company-to-company, and even industry-to-industry in a chaotic career dance that is only controlled by the tempo the employee sets.

A well-developed and polished personal brand will help you navigate a variety of options while at the same time positioning you for new opportunities.

While your personal brand can be many things, it must be the following:

- Honest and authentic
- Promise value
- Build trust
- Offer differentiation
- Continually grow and develop

Developing your personal brand is not as daunting as it seems, and can be broken down into three key steps: investigating, expressing, and evolving.

Investigate Your Brand

In the same way a company would decide to go public, for you to present your brand to the world you must first begin with what the experts refer to as "due diligence." This is nothing more than taking the time to thoroughly investigate your self. What types of projects have you worked on in the past? What endeavors would you like to focus on in the future? What are your best skills? What are you known for? How does your involvement with a company, project, or program make it better?

The answers to these questions help inform your personal brand offering – the attributes that are uniquely yours and for which people admire and seek you out. This searching process requires you to consider your past performance and highlight the key moments where you achieved stated goals, exceeded expectations, and contributed to the success of a project. By making a comprehensive list of these activities, you will identify your personal brand assets.

Express Your Accomplishments & Contributions

After identifying the traits and characteristics that comprise your personal brand offering, you must design a means for expressing these skills in the context of your accomplishments and contributions. In short, you must show people what you're good at doing.

The Personal Annual Report

You may have colleagues who are great at answering the question, "What do you do?" You may constantly read about a peer and their new book/award/speaking engagement/etc. While there are many ways to package you achievements, one format that is familiar in the business world and yet offers tremendous opportunities for personal adaptation is the annual report.

Companies use annual reports to provide information on their past performance, strategic goals, marketing plans, operational objectives, and financial soundness. Annual reports are often signature pieces that allow an outside "investor" to quickly ascertain a company's value and review progress in the fixed space of a limited time – one year of activities.

Individuals can develop a personal annual report to help showcase their activities over the past year. A personal annual report provides details about your professional accomplishments, your strategic goals, your past performance, and your ongoing efforts at self promotion.

Many people will quickly note, "But Joey, I've worked on the same project for two years running and we're still in development. I don't have any accomplishments to highlight!"

Really?

Have you helped keep the project on schedule? Have you made suggestions to improve the finished deliverable? Have you managed the work of other people? Have you contributed to the efforts of the team? Have you pushed the project in new and exciting directions? Have you delivered what you promised you would?

If you can answer any of these questions with a "yes," you have something to highlight. Compiling these highlights in a personal annual report offers a much more interesting and dynamic presentation of your successes than the typical, boring, one-page resume with "Experience," "Education," and "Hobbies" serving as the main headlines. A personal annual report allows for compelling design, illustrative charts and figures, and even (gasp!) photos of you at work! To a prospective employer or team leader, this type of presentation will help you stand out among the masses while at the same time offering candid insight into the type of person you are and what it will be like to work with you.

Once you develop your annual report, you can send it to friends, colleagues, potential employers, and even your current boss. A personal annual report is a great way to update people on your achievements and serves as a wonderful tool to introduce people to your personal brand.

The E-Presence

Given an audience of ColdFusion programmers, a discussion of the importance of a strong e-presence is probably like telling a jockey about the importance of a good saddle. Nonetheless, it's significant to note that with each passing year, an individual's online presence and digital identity become more critical. With people "googling" each other before a date, reading up on a seller's ratings before making a purchase, and contributing user reviews to product sites for others' consideration, a well-designed e-presence has never been more important.

My informal research of audiences around the country and the world shows that even in a room of technologically savvy professionals only about 45% have their own Web site. Approximately 25% have a blog – and about 10% of them update it regularly. And these are the statistics for those on the cutting edge of technology!

You might ask, "Since not many people seem to be focusing on their e-presence, why should I?"

There are two reasons. First, the lack of conscious focus by your peers and others means that you have a chance to make a big splash without a tremendous amount of work. Second, you're going to need to focus on your e-presence if you plan to earn a living in 2010. If you don't think you'll need to be interacting in society as a professional or an individual then, feel free to stop reading this article right now.

Given the increasing importance of an e-presence, you should ask yourself what your e-presence currently looks like. Does it represent who you truly are? Are you happy with the links that come up when you google yourself? If you tell me your name, can I find links to you in the first two pages of an online search? What are you doing to enhance you digital identity?

If you haven't considered these questions before, now is the time. Make a list of three things you can do in the next two weeks to improve your online positioning. They can be as simple as registering a personal Web site (less than $10 at GoDaddy.com), starting a blog (check out TypePad for ease of use), or just searching your name to find out what's already being said about you (you'll probably be surprised at just how much is already out there).

Evolve Your Offering

Once you've investigated your personal brand, developed a personal annual report, and enhanced your online presence, the work is done right?

Not even close.

A personal brand is constantly evolving. Consider yourself

Elements of a Personal Annual Report

- ***Letter to Investors*** – a brief narrative explaining past successes and future plans for personal growth and development
- ***Financial Highlights Summary*** – a detail of your contributions to the bottom line – i.e., how you increased revenue, found savings, or supported better financial performance in your past projects
- ***Year-in-Review*** – a timeline showing key accomplishments and milestones
- ***Testimonials*** – detailed opinions of what it's like to work with you from co-workers, bosses, mentors, and clients
- ***Contact Information*** – phone numbers, addresses, Web sites, and other ways to get in contact with you

REGISTER TODAY AND SAVE!

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

www.AjaxWorldExpo.com

AJAXWORLD™ EAST

CONFERENCE & EXPO

NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

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

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

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

BEING HELD APRIL 2-3 2007!

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

just five years ago – are you the same person? Do you have the same friends? Do you work in the same job? Do you live in the same place? Do you enjoy doing the same things? I'd be shocked to find you respond "yes" to even half those questions – and at the same time disappointed if you did.

Don't be afraid to let your personal brand grow as you grow. As you acquire new skills and develop new interests, your personal brand should meld and morph to include these new personal attributes and skill sets.

Consider the music superstar Madonna. Over the years, her personal brand has gone from "Like a Virgin," to a "Material Girl," to "Papa Don't Preach," to "Who's That Girl?" to "This Use To Be My Playground." Her career as an author has seen tiles ranging from the racy "Sex" to the children's book "The English Roses," and her movie roles have included films as diverse as "Truth or Dare," "A League of Their Own," "Die Another Day," and "Evita." Even her name has evolved from Madonna Louise Ciccone, to Madonna, to Esther, and back again to just Madonna.

While this kind of personal brand evolution is extreme, it shows that a personal brand can go through radical changes – yet continue to showcase the development of individual skills, interests, and attributes. While modeling a career on Madonna may not be the best choice for you, it's possible to find inspiration in her willingness to change directions and constantly reinvent herself.

The Time Is Now

The "IPO" of your personal brand doesn't mean "immediate pay off." The time spent developing and honing your feature will result in great dividends – both financial and personal – but they will come over time as opposed to all in one day like the IPOs of the 1990s tech boom.

The key to launching the IPO of your personal brand? *Getting started.*

You have all the assets necessary to investigate your personal brand, express your accomplishments and contributions in a well-designed and thought-out package, and consistently evolve your brand to take you to new heights. Nothing is stopping you from riding your personal IPO to newfound success and accomplishment – and having a great adventure along the way.

About the Author

Joey Coleman is chief experience composer at Design Symphony. A recovering attorney, his personal life experience has seen him serve in the Office of Counsel to the President of the United States during the Clinton Administration, juggle in front of the Taj Mahal, and establish a marketing and design firm specializing in the development of amazing experiences for clients' customers. He is a frequent presenter at conferences and was selected as the top speaker at CFUNITED '05. To receive your own Personal IPO kit, visit: www.designsymphony.com/IPO/.

joey@designsymphony.com

ColdFusion U

For more information go to...

U.S.

Alabama
Huntsville, AL CFUG
www.nacfug.com

Arizona
Phoenix, AZ CFUG
www.azcfug.com

California
Bay Area CFUG
www.bacfug.net

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

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

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

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

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

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

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

Florida
South Florida CFUG
www.cfug-sfl.org

Georgia
Atlanta, GA CFUG
www.acfug.org

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

Indiana
Indianapolis, IN CFUG
www.hoosierfusion.com

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

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

Maryland
Maryland CFUG
www.cfug-md.org

Massachusetts
Boston CFUG
http://bostoncfug.org/

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

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

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

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

Minnesota
Twin Cities CFUG
www.colderfusion.com

Missouri
Kansas City, MO CFUG
www.kcfusion.org

Nebraska
Omaha, NE CFUG
www.necfug.com

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

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

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

New York
Albany, NY CFUG
www.anycfug.org

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

New York
Syracuse, NY CFUG
www.cfugcny.org

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

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

Oregon
Portland, OR CFUG
www.pdxcfug.org

Pennsylvania
Central Penn CFUG
www.centralpenncfug.org

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

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

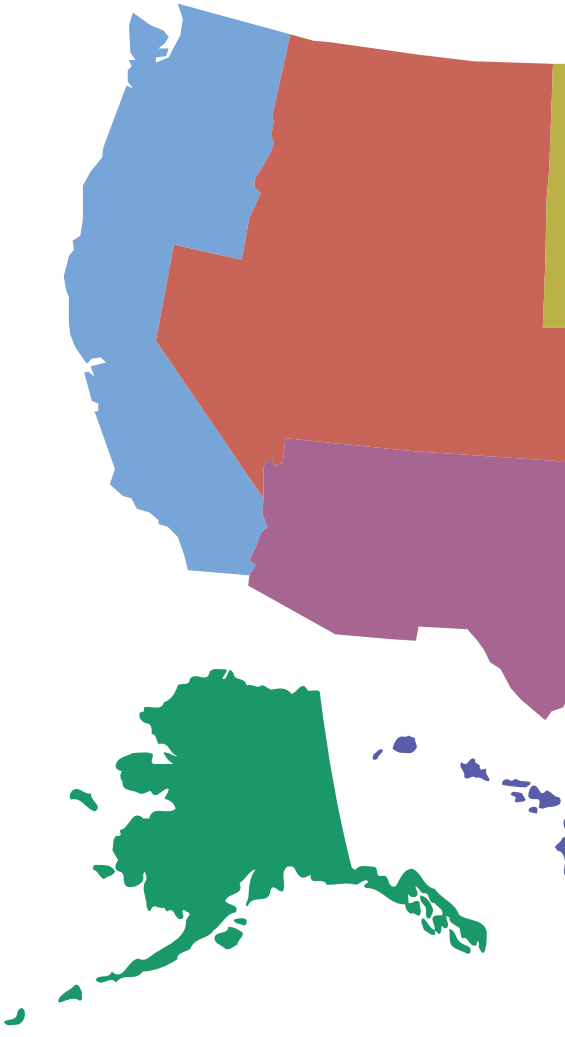
Tennessee
Nashville, TN CFUG
http://www.ncfug.com

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

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

Texas
Dallas, TX CFUG
www.dfwcfug.org/

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



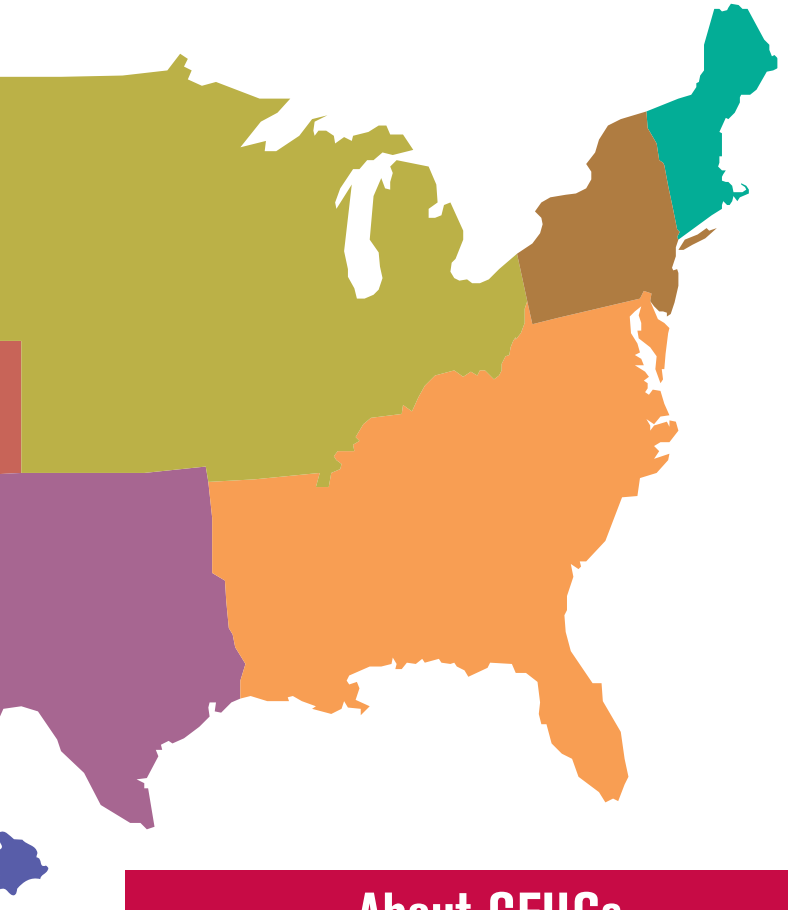
Utah
Salt Lake City, UT CFUG
www.slcfug.org

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

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

User Groups

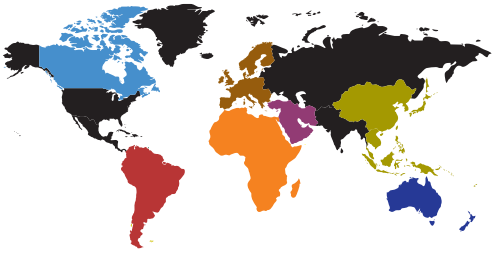
http://www.macromedia.com/cfusion/usergroups



About CFUGs

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

INTERNATIONAL



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

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

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

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

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Germany
Central Europe CFUG
www.cfug.de

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

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

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

Scotland
Scottish CFUG
www.scottishcfug.com

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

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

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

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

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

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

FusionDebug Tips, Tricks, and Traps

A cure for people who don't read documentation

By Charlie Arehart

FusionDebug is an easy straightforward tool, but if you leverage the experience of others, you can be even more productive.

Elsewhere in this issue, Jeff Houser introduces the FusionDebug interactive step debugger. I'd like to carry that discussion a bit farther and share various tips, tricks, and traps I've learned from my experience with the tool and that I've gleaned from available FusionDebug resources.

First I'd like to share some general tips that address some common questions or concerns I've observed, broken down into tips on installation and general use. These should help you appreciate the tool's use and how to make the most of its available features.

Then I'd like to share some tricks or features that you may not readily notice that will make the tool still more useful. Finally, I'll share some traps or problems that you may encounter and how to resolve them.

Tips on Getting Started

Why Bother Using a Debugger?

Perhaps the first and most important tip to share is my perspective on why I think FusionDebug is an important tool. I realize that there are some who dismiss it or may simply feel that there's nothing it can do that CFDUMP or CFOUTPUT can't. I disagree and have addressed those concerns in the first two entries in a series in my blog at http://carehart.org/blog/client/index.cfm/fusiondebug.

What If You Have Problems Using or Installing the Tool?

I've been using FusionDebug for a couple of months now and I'm obviously a fan. Still I know some have had problems using or installing it. Another important tip is that FusionDebug support is great. The folks at Intergral, who also make FusionReactor, offer free support for the tool at support@fusion-reactor.com.

How Do You Learn About Using and Installing the Tool?

The vendor's site, at www.fusiondebug.com, offers several very useful resources, including a well-done User Guide and Captivate videos showing how to use the tool. There are ample support FAQs and articles on the site to help solve common installation and troubleshooting glitches. Again, I'll discuss some of the gotchas or traps later.

I Really Don't Want to Switch to Eclipse

Here's really good news: you don't have to switch to Eclipse. Yes, the tool does run atop Eclipse, as Jeff explains, but you don't have to give up your favorite CFML editor, whether it's Dreamweaver, HomeSite+, or something else. Use those for editing, and fire up Eclipse (and FD) when you debug.

And you don't necessarily have to install Eclipse, though you may already have it installed. Are you using FlexBuilder? That, too, is an implementation of a commercial plug-in atop Eclipse. You could add FD to that. If you do, when you're debugging Flex apps the tool will switch "perspectives" when control passes from the Flex app to the CFML page it calls.

I Don't Want to Risk Hurting My Current Eclipse Implementation

You may want to think twice about installing FusionDebug on top of FlexBuilder or even the CFEclipse-based implementation of Eclipse you may have. If you have any concerns about one plug-in hurting another, remember, you can install more than one copy of Eclipse. When you download it from Eclipse.org, it's just a zip full of files that you extract to a directory. It's okay to install it more than once – provided it goes into another directory.

Along the same lines, another reason not to install FD atop a FlexBuilder install is if you're running the FlexBuilder trial. When it expires, you'll no longer be able to open that version of Eclipse.

Tips on General Use

Does It Just Debug CFM Pages? I'd Like to Debug CFCs and Calls from Flex, AJAX, Web Services, and Such

Great news: it not only debugs CFM pages but CFCs as well. Regardless of whether you instantiate the CFC in CFML or invoke it via Flex, AJAX, Web Services, Flash or Flash Remoting, or CFMX 7's gateways, FusionDebug will intercept the request.
More than that, the tool is savvy enough to recognize that there are special variables inside a CFC or method, and it will show the var (function local scope) and scopes in the "variables" pane.

Further, when debugging in a CFC method, there will be additional information in the "stack trace" pane. This is the pane (top left as typically configured in FusionDebug) that shows the filename and line number of the line where control has stopped. When control is stopped in a CFC method, this will also show the function name for the method.

Does It Show Information on All Variable Scopes? Queries?

Yes, the "variables" pane shows all scopes, not just the variables scope, form, and URL, but also session, application, and server, as well as client, cookie, cgi, and so on. It also shows the

scopes related to specific tags like cfhttp and cffile.

It shows other information when a query is viewed in any scope, including all the columns and rows, and all the data in each row, as well as information on the "current row" when processing a result set in a query loop.

Can It Debug CFSCRIPT Code?

Again, for some reason, many seem to have low expectations for the tool and often ask if it can debug code in CFSCRIPT. It absolutely can.

Does the "Step Into" Control Step into Nested Tags Like Those Inside a CFIF or CFLOOP?

No, "step into" isn't related to that, but rather to stepping into code in a new file such as a CFC method, CFINCLUDE, or custom tag.

What Does the User See While a Page Is Being Debugged?

When a browser requests a page that's debugging, it will appear to be hung until the code sends output.

I Want to Be Able to See the Output Being Generated During Debugging

Following on to the previous tip, you may not see the output being sent to the browser while in the midst of processing breakpoints because ColdFusion buffers output until the page completes or the page buffer fills, or a CFFLUSH tag is executed. While it would be nice if the tool offered a pane to see the output as it's being generated, you have a couple of other options if you really want to see the output during debugging. You could insert CFFLUSH tags periodically in the page, or use the available CFFLUSH INTERVAL attribute to cause flushing after a certain number of bytes. Just be aware that, as explained in the CF docs, the browser affects the CFFLUSH INTERVAL attribute and may not generate output if enough data hasn't been sent. The browser might not render the output until certain tags are closed (e.g., images, tables, and frames).

Tricks

Moving from tips to tricks, the following are facets of the tool that you may not have thought to expect.

You Can Debug Against More than Just Your Local Developer Edition of CFMX

Most debugging tools work against a local copy of whatever application you're debugging, but FusionDebug can be configured to debug code running on a remote CFML server too such as a central staging or production server. It could even be a remote hosted server. Of course, the server has to be configured for debugging as Jeff explains in his article so it's not a total free-for-all. Still, the feature is a two-edged sword. See my discussion of traps at the end of this article.

You Can Change Variables on-the-Fly

Don't miss the fact that the tool is for more than just watching what's happening while executing a page request. It offers the option of changing the variables' values on-the-fly. While at a breakpoint or stepping through code, highlight a variable you'd like to set and right-click "Set Variable" to give it a new value. That new value will immediately be set in the currently running request. If changing that value would cause a change in the value of an existing expression in the "expressions" tab, right-click the expression and choose "Reevaluate Watch Expression."

Available Shortcuts

Besides using the mouse-based icons and context menus, there are several keyboard shortcuts available. To set a breakpoint, use ctrl-shift-b. To effect the "step over," use F6. For the "step return" command use F7, and for "Resume" use F8.
Though not a "keyboard" shortcut, note that you can double-click on a breakpoint and FD will open the file at that specific line. Similarly, when at a breakpoint or while stepping, you can double-click on a file in the "stack trace" (showing the current template and line number on which execution has stopped) and that will jump to the appropriate line too. This can be especially helpful when you're several files deep (perhaps a file has called a CFC method that has done a CFINCLUDE).

If you want a quick shortcut for disabling all breakpoints, note the "skip breakpoints" command (an icon in the breakpoints window pane toolbar or under the main menu's Run command.) It's worth noting that the Run menu shows all the other stepping and related commands, so you don't have to use just the icons and keyboard shortcuts for stepping. This makes it clear that the debug commands used in FusionDebug are actually Eclipse commands, so you can learn more about some of these features in the general Eclipse documentation and other resources.

Tricks in Variables and Expressions Panes

When viewing the variables and expressions panes, note that if there are a large number of items, you can use the find option to locate a specific variable. With the cursor in the appropriate pane, use ctrl-f or right-click and choose "find." Note the available wildcards, ? and *.

To set a watch expression, you can either highlight it in the editor, right-click and choose "watch expression," or right-click in the expression pane and choose "add watch expression." Finally, you can right-click on an existing expression and choose "edit expression" to change the expression.

If you want to save the value of some variable as shown in the variables pane, notice the "copy variables" option.

Manipulating Breakpoints

After working with the tool for a while, you may start to have a large number of breakpoints show up in the breakpoints pane. A nifty trick is to group them (group their display) by choosing the rightmost icon (or "view action") in its toolbar, the down arrow, and "group by," and then select a value by which to group the breakpoints. Options include grouping them by the file or project in which they appear. Another option is to create "working sets" of breakpoints that are related in some way you define. They can then be grouped by those working sets and you can move the breakpoints around to appear in those working sets.
If you wanted to remove breakpoints for reuse later, you can also right-click in the breakpoints pane and export or import them.

Available Java Detail Mode

When configuring a debug setup (as Jeff explains in the initial install and setup steps), there's an option to "Display very

detailed Java information" that causes FusionDebug to interpret variables like Java objects instead. This mode is useful for Java programmers and those wanting to see the internal structure of a ColdFusion request. It's an "expert" feature, so be careful.

Traps: Challenges with Breakpoints

Finally, I want to address several things that can trip you up. Some of them are documented in the FusionDebug manual and Web site; others are ones I've found (and since reported) myself.

My Breakpoints Don't Fire as Expected

You may have problems with how you've configured your FusionDebug setup. See the support article at www.fusion-reactor.com/fusiondebug/support-settingUpDebugConfig.html. One of the features mentioned there is an option called "Compile CF page in debug-friendly mode." This is the default but if you have a problem it would solve, you have to recompile the affected templates while you're debug-enabled. That means you need to open and change the file (even if that means just entering a space and removing it) and then save it so that the next request for the page will recompile it.

Another issue is that if you're configuring debugging against a server that's not your local machine, be careful when configuring the "web server folder" in the "connect" tab since the value for the location of the Web server files should be relative to that Web server (not relative to your local machine, such as if you use a mapped drive locally).

Another possible problem is if you set a breakpoint on a line that doesn't contain CFML tags or expressions. Eclipse won't prevent that, but execution won't break on that line.
Relative to that, the breakpoint line numbers are fixed and static. If you change the file to add or remove lines, the breakpoint line number(s) may now be incorrect, which could cause the problem above (or the breakpoint may now be on a CFML line that doesn't execute).

Still other issues are explained at the support FAQ, "Why don't my breakpoints fire" at www.fusion-reactor.com/fusiondebug/support.html#breakpoints4.

My Attempts to Step Don't Show Stepping

There can be a couple of issues here. First, pay close attention to the stack trace pane. Again, this is the top left pane showing the file and line on which control has stopped. If you don't see the editor reflecting control moving to the expected next line, look to see if the stack trace pane shows the control is still "stepping." This means FusionDebug is still working to run your code.

Another instance where it may not seem to be stepping is when ColdFusion MX sometimes performs optimizations around tags. Try using step into when control is on the line before the tag.

I've found an even more pernicious problem that makes stepping seem not to work. The problem exhibits itself while control is stopped on a given breakpoint. Despite subsequent requests to step through code, no new lines seem to be reached in the file editor, but if you watch the stack trace pane you see the line numbers properly stepping through the code. The problem occurs because you opened the file using the file system rather than an Eclipse "project."

FusionDebug Is Opening the Wrong Files When I Step into New Files

The previous comment had to do with a problem created when you yourself opened a file, but FusionDebug will also open files when you're stepping through code or when it hits a breakpoint. If you find that it's opening the wrong file, see the support article at www.fusion-reactor.com/fusiondebug/support-wrongSourceFile.html.

FusionDebug Is Asking Me to Pick the Location of the File It's Opening

Related to the challenge of FusionDebug trying to open files, you have the option when configuring the source location (as Jeff describes in the setup steps) of choosing to "search for duplicates." If this feature is enabled, and FusionDebug finds more than one file of the same name in the project's folders, it will prompt you to pick which to use.

This and the previous challenge do argue for keeping your Eclipse projects as small as it can be reasonably set. As tempting as it may be to define a project at your Web site root, where you may have dozens of folders (or more), the problem is that Eclipse may find many files with the same name (such as index.cfm, application.cfm, and so on).

While at a Breakpoint or Stepping, I Find that FusionDebug Loses Its Connection to the Request

There can be a few explanations for this problem. First, if your CFML page experiences an error, then CFMX will return an error page to the browsing user, but no longer provide the debugging information that FD requires. As such, FusionDebug will just stop debugging that request with no indication of the error having occurred.

IMPLICATION OF REQUEST TIMEOUT SETTING

Another common problem, especially in CFMX 7 (though it can happen in 6.1 as well), is that while you're at the breakpoint, ColdFusion could time out the request. Many know that the CFMX admin console permits specifying a value labeled "timeout requests after x seconds." This protects the server from excessively long-running requests.

Unfortunately, FusionDebug holding a page at a breakpoint will appear to the server to be a long-running request. If CFMX terminates the request, this will cause FD to lose the connection to the request without any indication.

While CFMX 6.1 has this option, it's disabled by default in the new CFMX installation, while in CFMX 7 it's enabled by default. You can override the page request timeout on a page basis using CFSETTING RequestTimeOut.

Implication of Holding Breakpoint Across CFLOCK and CFTRANSACTION

Although this won't necessarily cause a problem for you, something to be careful of when holding a breakpoint is running code inside a CFLOCK or CFTRANSACTION. Either could negatively impact other threads or users. In the case of CFLOCK, you could hold a lock that could hold up others trying to get the same lock. (Locks are a subject of frequent confusion, but further discussion is beyond the scope of this article.) In the case of CFTRANSACTION, you could hold a lock that holds up others trying to access the database you're processing since CFTRANSACTION (driven by its ISOLATION attribute) can tell

the database to prevent other reads or writes during the life of the transaction.

Traps: Challenges on Shared Servers

If you do choose to enable FusionDebug on a shared server, such as a central development, staging, or even production server, be aware of the following potential traps.

Breakpoints Remain Alive for Any User (Not Just the Debugging Developer) While Debugging Is Active

As discussed above, if a user runs the request while debugging is enabled (imagine some developer has FusionDebug open and enabled and a breakpoint set in a page whose execution fires the breakpoint), the page will appear to hang to the user. Only the developer with FusionDebug open will see that it's stopped on the given breakpoint. For now, there isn't any option in FusionDebug to limit the IP addresses whose requests trigger breakpoints.

When multiple users make a request for a page while it's being debugged and hit a breakpoint, FusionDebug will show a "thread" for each user's request in the "stack trace" pane. If this happens and you want to quickly stop the debugging, you can choose to use the "disconnect" (not "terminate") icon in the stack trace pane. That will stop all debugging in the current debugging session.

On the other hand, another approach would be to use the available "skip all breakpoints" option (in the breakpoints pane) to temporarily disable breakpoints from firing any further for the current debugging user.

Who Can Debug Your Server?

Since debugging is enabled simply by specifying a given port, any user who can issue a request against that server and port (and is running FusionDebug) can conceivably set breakpoints. Of course, they also have to be able to open the file via an Eclipse project to set a breakpoint, so effective file security and firewalls can mitigate this concern. Again, for now, there isn't any option in FusionDebug to limit what IP addresses can initiate debugging.

What Happens When Two or More Developers Try to Debug a Single Shared Server?

Because of limitations in Java and its debugging mechanism (which FusionDebug is leveraging on our behalf), you can't have more than one debugging session open against a given server at a time. If a developer attempts to start a debugging session when another developer has already started one, the request will be denied. Again, in such cases, this calls for judiciously controlling debugging sessions using the "disconnect" feature to stop debugging when no longer needed.

Traps: Some Unexpected Challenges

ColdFusion Doesn't Start After I Configure It for FusionDebug

It could be either of two problems. The simplest is if you've made a mistake in configuring the jvm.config file (Jeff explains what to do in the installation steps). If this file has errors then CFMX won't start. So it would be wise to save a copy before editing it, but if it's too late, just read the installation instructions carefully (or contact FD support).

A second reason can easily occur for those using CFMX Enterprise and its available multiple instance deployment option, where you have multiple CFMX instances on a single box. In this case, if you configure just one jvm.config that's shared among them all, there will be problems. See the support FAQ at www.fusion-reactor.com/fusiondebug/support.html#config2.

Find That While Stepping into Some Tags and Functions It Seems to Hang

This problem happens if you use "step into" rather than "step over" on certain tags and functions and stems from CFMX's underlying implementation atop Java. Some of the CFML tags and functions themselves call on a large amount of Java code, and since Eclipse debugs Java, it will also debug the Java code underlying these tags and functions. You won't actually see the debugging. You'll just see an indication of "stepping" in the stack trace.

Related to this, some CFML tags are actually implemented in CFML so the debugger wants to debug them too. Fortunately, the tool is smart enough not to take you into such pages to debug them as explained in the support FAQ "Why don't my breakpoints fire" at www.fusion-reactor.com/fusiondebug/support.html#breakpoints4. Again, it will just show "stepping" in the stack trace, and that tag or function may take longer to run when stepping than it would when running normally.

I've seen these issues with CFDIRECTORY, CFDUMP, createobject, and findnocase to name a few. The solution for now is to simply use "step over" instead. Using "step into" really only makes sense on tags that open new files. Perhaps eventually the vendor will solve this for us.

Finally, there are various troubleshooting tips in the user manual and support pages at fusiondebug.com.

Conclusion

Well, that's a pretty deep plunge into tips, tricks, and traps. I hope they're useful for you. I've spent nearly half the article on gotchas, but to be honest, many of them can be avoided if you simply read the documentation. But I realize many people won't bother so I've highlighted them to help make you more effective in considering using the tool and getting started with it.

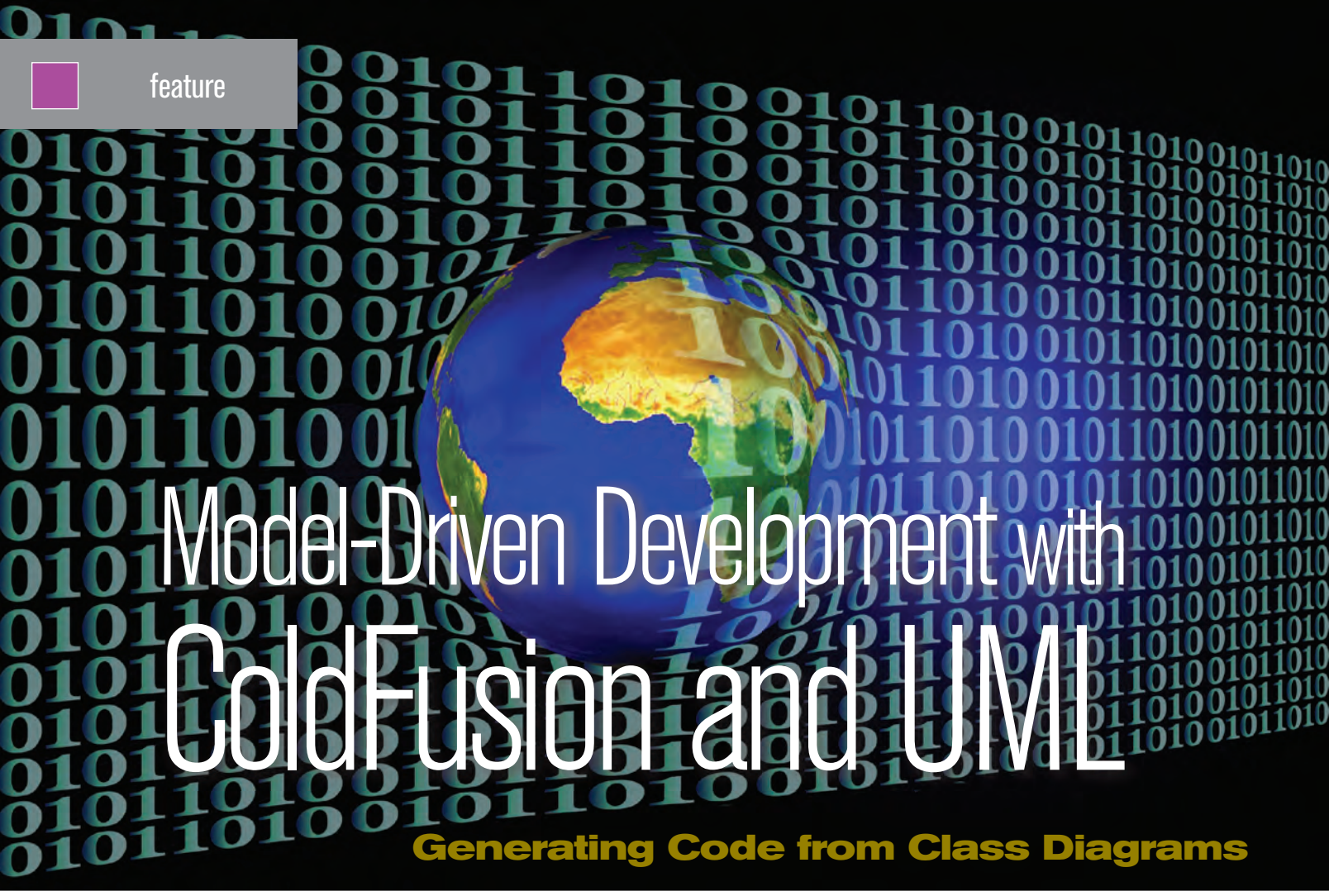
I really think it makes a great tool for the CFML developer's arsenal. And there's great support behind it — Intergral is no start-up; it's several years old. They're CFML developers committed to making tools and add-ons for CFML developers. Check out the free trial, as well as community and volume discounts.

inally, while you should share your experiences with colleagues, I'd recommend that you take any negative ones straight to the vendor. I've found it very responsive. It's a brand new product and will certainly get better with time.

About the Author

A veteran CFML developer since 1997, Charlie Arehart is a longtime contributor to the community and was recently selected to the Adobe Community Expert program. Many know he served as tech editor of the CFDJ until 2003 and was co-author of the CFMX Bible. A certified Advanced CF Developer and Instructor for CF 4/5/MX, he?s frequently invited to speak to developer conferences and user groups worldwide. Formerly CTO of New Atlanta (BlueDragon), he is now an independent contractor and still lives in Alpharetta GA where he is president of the Atlanta CFUG.

charlie@carehart.org



Model-Driven Development with ColdFusion and UML

Generating Code from Class Diagrams

By Chip Temm

The sign of an experienced developer is solid design. Novices edit examples they find on the Net, journeymen figure out how to code something as they do it, but craftsmen plan their work. Starting out, this can look like wasted time, but if your app is any good, your customer will want to expand it. Then, if you need some help, you'll have to explain all that intuitive code to ten people, all of whom you are paying by the hour. Diagrams would be nice then, right?

Many development organizations are moving toward more structured development cycles, involving formal documentation standards, to ensure that code is maintainable after the developer moves away from the project. For these organizations, the challenge is to balance the demand for rapid development with the requirement for clear, solid design and analysis documents.

For developers, it is annoying to go through the trouble of writing all of those class, property, method, and argument names only to have to do it again in the database and in the code. Wouldn't it be great to do that once and then fill in the details – kind of sketch out the application – so that the model would drive the development and get developers enthusiastic about the whole process? In this article I'll illustrate how to do this by taking a UML class diagram and using it to generate ColdFusion code. From this simple starting point, one could go on to create stubs for complex CFCs, tables, stored procedures, unit tests and even simple CRUD GUIs.

A Picture Is Worth a Thousand Words

To start, we need to find a diagramming tool that is capable of exporting its diagrams in a text format. UML tools generally seem to be using XMI (XML Metadata Interchange) to handle this, and Gentleware's Poseidon Community Edition comes to our rescue as a capable, free editor. The downside is that XMI is pretty verbose. Diagram two or three classes and you're already clocking hundreds of lines of text. The upside is that it's XML, and with ColdFusion's XML functions for parsing and searching, we'll be able to tackle this with ease... after we figure out the format of that huge file. Now that we have a tool to edit our diagrams and save them to text, we can plan our application. We're writing a tool to generate code from diagrams, so let's start by diagramming. That way we'll have something to generate at the end of this article. We know that we'll have to do at least four things: parse the XMI; create a model of our results that is easy to manipulate; generate code based on the model; and, finally,

do something with the generated code (write to a file, execute, etc.). These things are likely to be complicated enough that they will involve a set of functions, so, on our diagram, we'll create four classes to start with: XMI_Parser, Generator, Model, and Writer.

Parsing the XML

First we'll have to get our tool to export an XMI version of our diagram. In Poseidon, this is as simple as File->Export Project to XMI. Open up the diagram you created earlier and export it

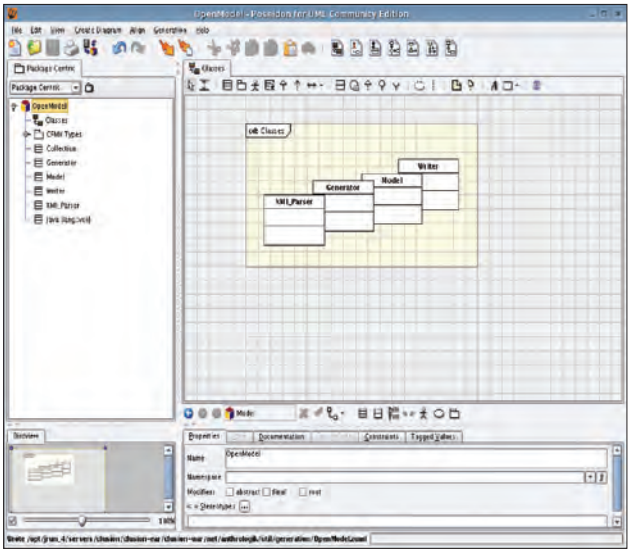


Figure 1

to XMI, then open it in a text editor or an XML editor. You will see, as you browse through the file, that there is a lot of layout markup that is irrelevant to us: We're only interested in 17 of this file's 660 lines! The XML will get more and more complex as we add functions, attributes and new datatypes. We will need to find the information that is relevant to our application design, then extract and simplify it to make it easier to work with for the rest of the project. This requires a function that wraps a series of XPath queries by way of ColdFusion's xmlSearch. We'll put that on our diagram as a public function in the XMI_Parser class as "createModelFromFile," which will take a string called fileName (the path to the XMI file) and return a Model object. Poseidon shows this as "+createModelFromFile(fileName:string):Model." For our purposes, we start by identifying classes used in the diagram:

```
   <cfscript>
          xmlDoc = xmlparse('OpenModel.xmi');
          classes = xmlsearch(xmlDoc,"//UML:Class[@xmi.id]");
          classesLen = arrayLen(classes);
          for(i=1;i lte classesLen;i=i+1){
                    thisClass = classes[i];
                    //now loop down into properties, methods and argu-
ments
          }
   </cfscript>
```

Deconstructing the XPath Query

Our XPath query for classes looks through the whole XML document for all elements of type UML:Class that have an xmi.id element ("[@xml.id]]"), regardless of their position in the document ("//"), and returns an array of ColdFusion XML objects that match. Listing 1 shows the XML element describing our XMI_Parser class at this point in the design.

We can see that we are interested in elements of type UML:Class that contain elements of type UML:Attribute and UML:Operation. UML:Operation elements also contain UML:Parameters, some of which are input (kind="in") and some of which are output (kind="return"). The subelements refer to other Classes or Datatypes for their type info: <UML:DataType xmi.idref = '1'/>. Somewhere in our diagram there is a Class or Datatype that has xmi.id = '1' to match this reference. These IDs are assigned by the tool to track the entities being diagrammed. When we want to determine an element's type (e.g, string, date, etc.), we can do an XPath query to look it up:

```
   thisMethodReturnsDataType = xmlsearch(xmlDoc,"//*[@xmi.
id='#thisMethod.returnParam_XMI_IDREF#']");
```

Creating the Model

You can see now what we have to do. We are going to loop over the results of the classes XPath query, building a ColdFusion structure as we go that simplifies and abstracts the XML we are reading. For each Class, we will loop over the nested Attributes (properties for us), then the Operations (functions for us). For each Operation, we will loop over its Parameters (arguments). At the end of the day, we have a ColdFusion structure that looks like this:

```
COMPONENTNAME e.g."XMI_Parser"[
  name [string]
  displayname [string]
  hint [string]

   PROPERTIES[
          PROPERTYNAME[
                    name [string]
                    type [string]
                    required [BOOLEAN]
                    displayname [string]
                    hint [string]
                    default [string]
          ]
   ]
   FUNCTIONS[
          FUNCTIONNAME[
                    string name [string]
                    access [string]
                    returntype [string]
                    roles [string]
                    hint [string]
                    output [BOOLEAN]
                    ARGUMENTS[
                              ARGUMENTNAME[
                                        name [string]
```

```
                                type [string]
                                required [BOOLEAN
                                default [string]
                                hint [string]
                            ]
                        ]
                    ]
                ]
            ]
```

To ensure that we have a consistent interface to this data structure, we'll add functions to our Model class to add or remove information. Let's update our diagram with: addComponent, deleteComponent, addProperty, deleteProperty, addMethod, deleteMethod, addArgument, and deleteArgument. We'll also add functions called getModel (to return our populated struct), serialize (in case we want to save our struct to a file for later use), and deserialize (to populate a Model from a previously saved state), as well as a private property called componentStruct to keep track of everything as we go.

Populating the Model

Now we are ready to create a method that reads a UML diagram and populates a model of a CF application. Again, this is a simple matter of looping over all the classes we found in the XMI document and then drilling into each (Listing 2). From there, we will loop over the class properties, extracting what we find and pushing information into an instance of the Model class (Listing 3). Next, we loop over the functions and, for each of those, down into its arguments (Listing 4). When finished, our createModelFromFile function returns a Model object that represents the information in our diagram.

Generating Code from Templates

Having all of the diagram's information distilled into a struct makes it quite easy for us to generate out any kind of template code we want, adhering to whatever standard or framework fits our fancy. In the interest of time, we'll just show generation of the CFProperty section of a webservice-oriented CFC. As you'll see, it becomes simple to generate out all the code for that boring task, and generating out classes containing CFFunction 'stubs' for all of the documented functions is not much more difficult. While some code-generation frameworks encourage modeling your database first and then generating code from Entity Relationship model, I would suggest creating a class model and then generating your Data Definition Language scripts from your classes. You could go all the way and generate out stored procedures to provide supporting functions for generated stub code. I am writing such generators for my own use.

The approach used to generate code is straightforward. We create templates to generate ColdFusion or SQL code, using CFSaveContent to keep the generated content in a string buffer. To get ColdFusion to generate ColdFusion code, we use symbols to represent where in our code we want to write three special characters: <, >, and #. We want to keep the stuff we're generating fairly readable, so we substitute the double angle bracket characters for angle brackets («») and dollar signs ($) for the pound signs. The double angle bracket character is not on your keyboard, so if you follow this method, you will have to map that character somehow. I have a macro that replaces my two typed angle brackets with one double angle bracket character. Once we have generated the whole file we want to save and have it in a string variable, we can clean up all of those placeholder symbols with the real thing with some simple string functions like replace (input,'»','>','all').

The idea is to loop over a model struct outputted from our Model.getModel() function. Below is an example of generating CFProperty code using our special templating characters. In our generation framework, this kind of code would be wrapped up in functions in the Generator class.

```
<cfset myModel= model.getModel() />
<cfloop  collection="#myModel#" item="componentName" >
    <cfset currentCFC = myModel[componentName] />
    <cfloop  collection="#currentcfc.properties#" item="propName" >
        <cfset currentProp = currentcfc.properties[propName] />
        <cfsavecontent variable="generatedProperties">
            <cfoutput>
                «cfproperty name="#propName#"
                    required="#currentProp.required#"
                    type="#currentProp.type#"
                    hint="#currentProp.hint#"
                    default="#currentProp.default#"
                    displayname="#currentProp.displayname#"
                /»
            </cfoutput>
      </cfsavecontent>
    </cfloop>
</cfloop>
```

Saving the Code

Now that we have generated all this great stuff, we need to do something with it. Our Writer class will write out the generated content to files, but it would be nice if it would also help protect us from ourselves by, for example, moving existing files to a backup location to prevent overwrites and loss of work, and logging any problems encountered. Remember that, in many cases, the output is meant to be edited; it is just an outline for the real code. Down the road, maybe this class would be smart enough to use Subversion to try to merge changes from the model into your edited code. After cleaning up our generated code above and writing it to a file, we would see something like:

```
    <cfproperty name="lastName"
        required="yes"
        type="string"
        hint="The User's family name"
        default=""
        displayname="Family Name"
    />
```

Conclusion

In this article, we have attempted to lay the foundation for a code-generation system that encourages developers to model before they code. When the model does more than fulfill docu-

mentation requirements, it becomes a more valuable tool for the developer. The blueprint for the application is generated at the start of the project, providing benefits across the development team. For organizations, model-driven development presents an opportunity to build-in standards and best practices. Senior developers and architects can focus on iteratively improving their generators to produce more code cleanly and handle mundane, repetitive tasks. Junior developers are provided a great environment for learning, and the structure they need to complete their projects rapidly. If this concept is of interest to you and your team, I encourage you to check out the OpenModel project on CFOpen and to contact me at openModel@anthrologik.net.

References

- Author's project to develop this idea: http://cfopen.org/project/openModel
- A great primer on XPath: http://w3schools.com/xpath/
- The Subversion source control system: http://subversion.tigris.org/
- Poseidon UML: http://gentlesoft.com
- CFOpen: http://cfopen.org

About the Author

Over the past decade, Chip Temm moved from North America to Europe and on to Africa where his company anthroLogik solutions provided analysis and development services to non-governmental organisations across seven timezones. He is currently back in Washington, DC where "remote development" means working from home and "wildlife" means raccoon.

chip@anthrologik.net

Listing 1

```
<UML:Class name = 'XMI_Parser'
xmi.id = '1'
visibility = 'public'
isSpecification = 'false'
isRoot = 'false'
isLeaf = 'false'
isAbstract = 'false'
isActive = 'false'>
<UML:Classifier.feature>
<UML:Operation
 name='createModelFromFile'
 xmi.id = '2'
 visibility ='public'
  isSpecification ='false'
  ownerScope ='instance'
  isQuery = 'false'
  concurrency ='sequential'
  isRoot = 'false'
  isLeaf = 'false'
  isAbstract = 'false'>
<UML:BehavioralFeature.parameter>
 <UML:Parameter name = 'fileName'
  kind = 'in'
  xmi.id = '3'
  isSpecification ='false' >
   <UML2:TypedElement.type>
     <UML:DataType
      xmi.idref='4'/>
   </UML2:TypedElement.type>
  </UML:Parameter>
  <UML:Parameter name = 'return'
```

```
      kind = 'return'
      xmi.id = '5'
      isSpecification ='false'>
      <UML2:TypedElement.type>
       <UML:Class xmi.idref = '6'/>
      </UML2:TypedElement.type>
     </UML:Parameter>
   </UML:BehavioralFeature.parameter>
  </UML:Operation>
  <UML:Method xmi.id = '7'
   isSpecification='false'
   isQuery = 'false'>
  <UML:Method.body>
   <UML:ProcedureExpression
    xmi.id='8'
    language = 'java'
    body =''/>
  </UML:Method.body>
  <UML:Method.specification>
   <UML:Operation xmi.idref = '9'/>
   </UML:Method.specification>
  </UML:Method>
 </UML:Classifier.feature>
</UML:Class>
```

Listing 2

```
<cffunction
 name="XMI_to_model"
 access="public"
 returntype="model"
 hint="reads in an XMI file and
    returns a fully populated model"
 output="no"
>

<cfargument
 name="filePath"
 type="string"
 required="yes"
 hint="fully qualified path on the
    server containing a valid
    XMI document"
/>

<cffile action="read"
    file="#arguments.filePath#"
    variable="myfile"
/>
<cfscript>
//setup an empty model
var myModel=createObject
   ('component','Model').init();

var xmlDoc = xmlparse(myfile);

//extract the data from the XML
var classes =xmlsearch(xmlDoc,
  "//UML:Class[@xmi.id]"
);
var cLen=arraylen(classes);
var i=1;

//loop over the classes----------
for(i=1; i lte cLen;i=i+1){
  thisClass = classes[i];
  //add class to model
  myModel.addComponent(
    name=thisClass.XmlAttributes.name
  );
  //see sample3
  myModel=populateProps(
    myModel,
    thisClass,
    xmlDoc
  );

  //see sample4
  myModel=populateFuncs(
    myModel,
    thisClass,
    xmlDoc
  );
}
return myModel;
</cfscript>
</cffunction>
```

Listing 3

```
<cffunction
  name="populateProps"
  access="private"
  returntype="model"
  hint="inserts property information
        into a Model object"
  output="no"
>

<cfargument
  name="model"
  type="Model"
  required="yes"
  hint="the Model to populate"
/>
<cfargument
  name="class"
  type="xml"
  required="yes"
  hint="the xml element representing
        the class containing properties"
/>
<cfargument
  name="xmlDoc"
  type="xml"
  required="yes"
  hint="xml object containing all info
        on the packages being modelled"
/>

<cfscript>
var thisProperty = '';
var propType  ='';
var myModel   = arguments.model;
var thisClass = arguments.class;
var xmlDoc    = arguments.xmlDoc;

var properties =xmlsearch(thisClass,
  'UML:Classifier.feature/UML:Attribute'
);

var len_props=arraylen(properties);
var i =1;

for(i=1;i lte len_props;i=i+1){
  thisProperty =properties[i];

  //this function uses xmi.idref to
  //lookup the name of the CF type
  propType=
  lookupType(thisProperty,xmlDoc);

  //add property to class' model
  myModel.addProperty(
    componentname=
    thisClass.XmlAttributes.name,

    name=
    thisProperty.XmlAttributes.name,

    displayname='',
    type='#propType#',
    required='no',
    default='',
    hint='',
  );
  //some of these attributes may be
  //hard to support in the diagram
}
return myModel;
</cfscript>
</cffunction>
```

Listing 4

```
<cffunction
  name="populateFuncs"
  access="private"
  returntype="model"
  hint="inserts function information
        into a Model object"
  output="no"
>

<cfargument
  name="model"
  type="Model"
  required="yes"
  hint="the Model to populate"
/>
<cfargument
  name="class"
  type="xml"
  required="yes"
  hint="the xml element representing
        the class containing properties"
/>
<cfargument
  name="xmlDoc"
  type="xml"
  required="yes"
  hint="xml object containing all info
        on the packages being modelled"
/>

<cfscript>
var methods='';
var thisMethod='';
var len_methods =0;
var returntype='';
var returntypeElement='';
```

```
var arguments='';
var thisArg='';
var len_arguments=0;
var thisArgType='';

var i=1;
var j=1;

methods =xmlsearch(thisClass,
  'UML:Classifier.feature/UML:Operation'
);

len_methods = arraylen(methods);

for(i=1;i lte len_methods;i=i+1){
  thisMethod= methods[i];
  returntypeElement=
    xmlsearch(
      thisMethod,
     "UML:BehavioralFeature.parameter
      /UML:Parameter[@kind= 'return']"
    );

  //use a helper method to lookup the
  //related xmi.idref
  returntype=
    lookupType(thisMethod,xmlDoc);

//add methods to class' model
myModel.addMethod(
  componentname=
    thisClass.XmlAttributes.name,
  name=
    thisMethod.XmlAttributes.name,
  access='public',
  returnType=returntype,
  roles='',
  hint='',
  output='no',
);

//get arguments for method
arguments =
  xmlsearch(
    thisMethod,
   'UML:BehavioralFeature.parameter
    /UML:Parameter'
  );
//loop over arg list
len_arguments= arraylen(arguments);
for(j=1;j lte len_arguments;j=j+1){
  thisArg = arguments[j];
  if(thisArg.XmlAttributes.kind
     neq 'return'
  ){
      thisArgType=
      lookupType(thisArg,xmlDoc);

      //add arg to method
      myModel.addArgument(
        componentname=
          thisClass.XmlAttributes.name,
        methodname=
          thisMethod.XmlAttributes.name,
        name=
          thisArg.XmlAttributes.name,
        ordinality=j,
        type=thisArgType,
        required='no',
        default='',
        hint='',
      );
    }
  }
}
</cfscript>
</cffunction>
```

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

CF/Flex Super Wizarad

Building cool applications

By Mary McDonald

ColdFusion Extensions for Flex, included in ColdFusion MX 7.0.2, enable Flex Builder 2.0 developers to automate the more mundane tasks like CRUD (creating, reading, updating and deleting records) by creating ColdFusion components, ActionScript class files, and code to invoke a Web Service.

In this article we will use the ColdFusion/Flex Application wizard to automatically generate Master, Detail and Master/Detail pages that will let the user create, read, update and delete artist information supplied from a database. The application will include ColdFusion Components, Flash ActionScript classes, as well as Flex MXML pages. Special thanks to Damon Cooper and Dean Harmon for helping me better understand the CF/Flex Super Wizard.

First we have to make sure we have all the software we need to make our software writing jobs easier.

The following software will be used in this example:

1. Windows XP Home Edition (the operating system I am using)
2. Java2Runtime Environment Std Edition 1.4.2.12 (needed to run Eclipse)
3. Eclipse SDK 32win32 (the tool we will be using to develop in)
4. Flex Builder 2.0 with Charting (this contains the CFFlex Wizard)
   Please see my previous article "My First Flex App" in September CFDJ (Vol. 8, Issue 9) for instructions on how to install Flex Builder and Eclipse.
5. ColdFusion MX 7.0.2 (contains the ColdFusion Extensions for Flex Builder)
   To install the ColdFusion Extenstions for Flex Builder (in Flex Builder):
   A) Select Help > Software Updates > Find and Install.
   B) Select the Search for New Features and Install option, click Next.
   C) Click New Archive Site.
   D) Select the ColdFusion_Flexbuilder_Feature.zip file, click Open.
   (If you installed CF MX 7.0.2 using the default values, look in the Extras Folder.)
   E) When the Edit Local Site dialog box appears, click OK.
   F) Ensure the ColdFusion Flex Builder feature is selected, then click Finish.
   G) Select the check box next to ColdFusion_Flexbuilder_Feature.zip, then click Next.
   H) Accept the terms of the license, then click Next.
   I) Click Finish.
   J) Click Install All.
   K) When the installation is complete, click to restart Flex Builder.
6. Lastly, make sure RDS is set up correctly to allow ColdFusion to access the database you will be using. To configure RDS in Flex Builder:
   A) Select Window > Preferences > RDS Configuration
   B) I will be using the default localhost server in this example:
   - Select localhost
   - Enter Description
   - Host Name 127.0.0.1
   - Port Number 8500 (if you are using the built in web server)
   - I won't be using the Context Root.
   - RDS password

   C) To Test the connection, select the server, then click Test Connection.
   D) To view files in Flex Builder, select Window > Other Views, RDS, RDS Fileview to select files or RDS Dataview to select data sources.
7. Please reference the livedocs for more information on setting up the ColdFusion extensions and RDS in Eclipse to use ColdFusion with Flex: http://download.macromedia.com/pub/documentation/en/flex/2/using_cf_with_flex2.pdf

Once everything is installed, we are ready to use the CFFlex Wizard to create an application. This application demonstrates how you can use the CFFlex Wizard and Flex Builder 2.0 to "automatically" create a Master, Detail, and Master/Detail page to create, read, update, and delete artist information along with their specific pieces of art. As you will see, the CFFlex Wizard will generate Flex, Flash and ColdFusion pages for you. Read on and be amazed!

To Create the application while in Eclipse, select CTRL + N

and select the ColdFusion/Flex application wizard (see Figure 1).

Press Next and read through text displayed. Press Next again (see Figure 2).

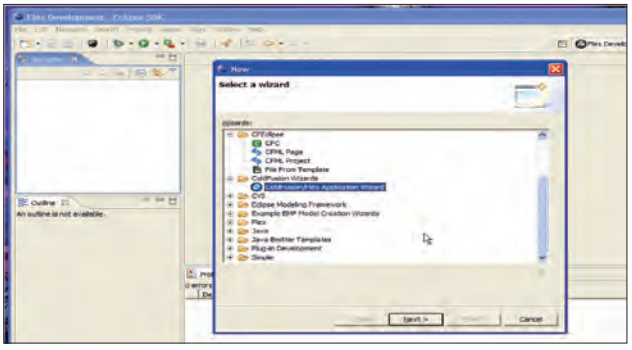


Figure 1

Figure 2

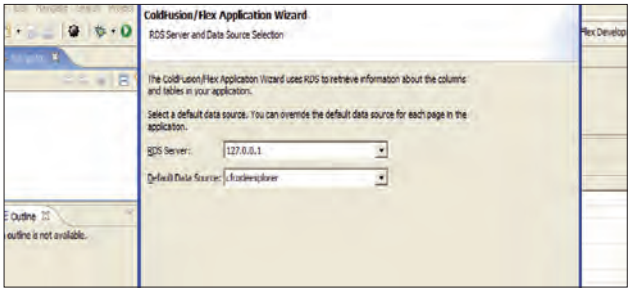


Figure 3

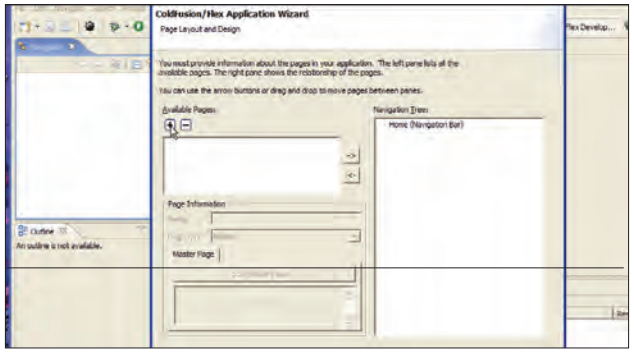


Figure 4

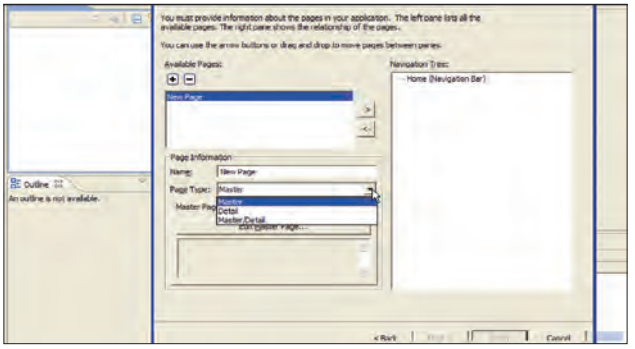


Figure 5

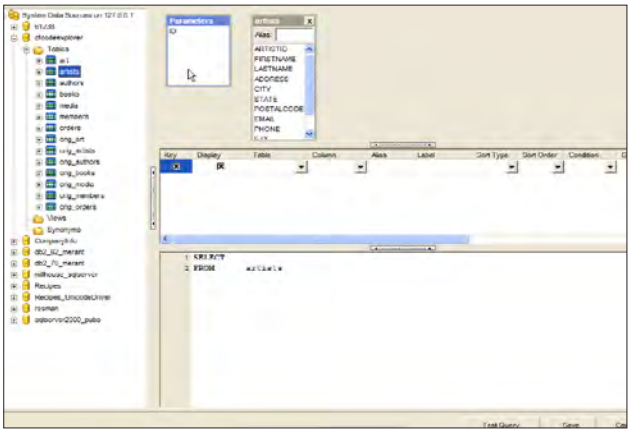


Figure 6

Since this is a new application we will click Next to specify our settings. (If we wanted to load settings from a previous application, we could just select the "Load ColdFusion/Flex Application Wizard Settings" button instead) (see Figure 3).

Select your RDS Server where your Data Source is located, then select the Data Source being used in your application. Press Next. (ColdFusion uses RDS to get the metadata from your Data Source here) (see Figure 4).

This is where we will define the Page Layout and Design for our first page (see Figure 5).

We will select "Master" for the Master Page Type. This will generate a page that just lists information. Then select the "Edit Master Page" button (see Figure 6).

Select the artists table from the list of tables on the left (see Figure 7).

Select from the artists table the ARTISTID, FIRSTNAME and LASTNAME. Click Save (see Figure 8).

Enter "List of Artists" next to the Name: label to name the page. Select + to Add the next page (see Figure 9).

Name the page Artist Detail and select "Detail" as a Page Type then select the "Edit Detail Page" button (see Figure 10).

Double-click the artists table. This time the wizard automatically ties the artists to the ARTISTID as indicated by the "ID" parameter (see Figure 11).

Next we add the two pages to the Navagation Tree to the left. Select Next (see Figure 12).

Here we can change settings that could affect the Flash Re-

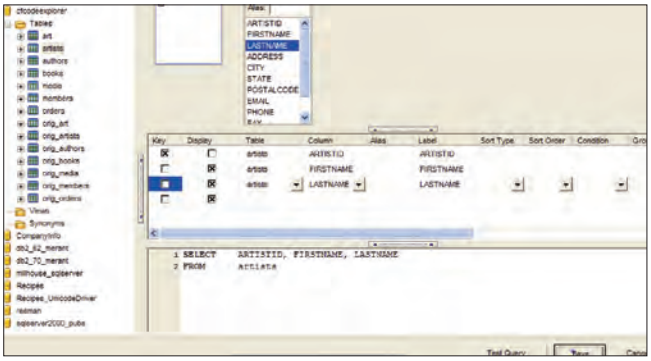


Figure 7

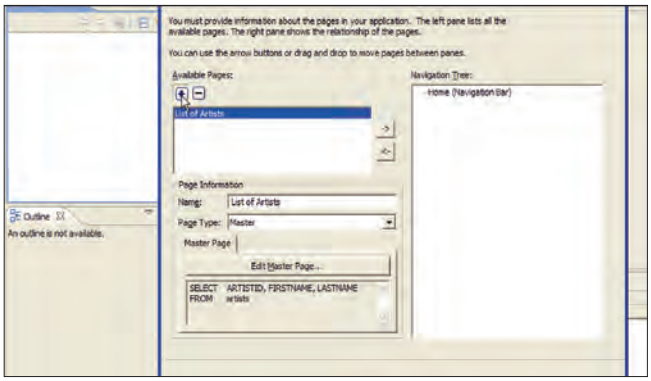


Figure 8

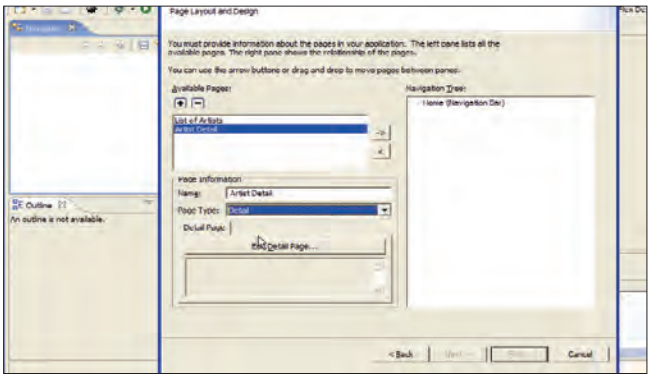


Figure 9

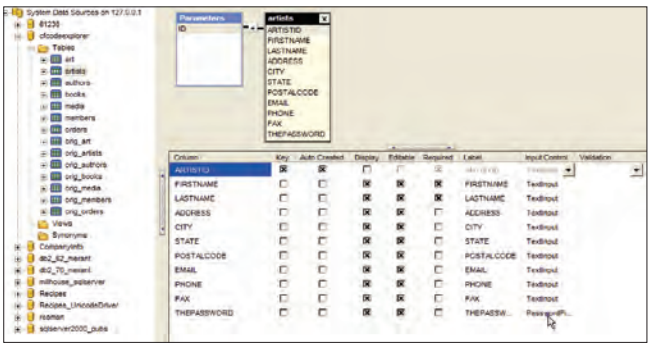


Figure 10

moting settings; the Web Root URL settings; the Project Name and Web Root Folder, where the Project will be placed; as well as the Wizard Settings. This time we will accept the default settings, so click Finish (see Figure 13).

The wizard makes the ArtExample project, then compiles it! Automatically created is the ColdFusion code (as seen by the ColdFusion components on the left) as well as the Flex code (as seen by the Main.mxml Source code on the right). WOW! That's what I thought when I saw this! (see Figure 14).

Here, we just double-clicked on the List_of_Artists.cfc to look inside at the query (see Figure 15).

Notice that in the generated folder, there are model, view, and controller folders. Click on the Artist_Detailartists.as file to look at the ActionScript model (see Figure 16).

This ActionScript file was automatically generated. Are you blown away yet? Next, we will run the application (see Figure 17).

Select the Run > button, Flex Application, New button (see Figure 18).

Use the defaults here and press the Run button (see Figure 19).

By pressing on the List of Artists tab above, the user sees a list of artists. Users at this point can "automatically" create, read, update and delete artists from the database in real time. With Taylor Webb Frazier highlighted, select the update (pencil icon) button. (see Figure 20).

Notice a list of artists. We want to display a detail list of their artwork on this page as well. To do that we must re-run the wizard. Don't worry, you don't have to start again from scratch. Close your browser where the application was running (see Figure 21).

Select Ctrl-N, ColdFusion/Flex Application Wizard, Next (see Figure 22).

This time select the Lode ColdFusion/Flex Application Wizard Settings. Press Next (see Figure 23).

Keep the RDS settings the same and press Next (see Figure 24).

The wizard remembered our settings. To add the artist detail grid, press the Edit Detail Page button (see Figure 25).

Click on the art table twice to add the art table detail information. Notice the ARTISTID in the art table was used to tie the art table to the ARTISTID in the artists table. Uncheck the MEDIAID and ISSOLD checkboxes under the Display column to not display those fields. Press Save. Press Finish. Run the application (see Figure 26).

Taylor Webb's Detail information is displayed in edit mode. Notice the bottom of the page displays Taylor Webb's artwork, allowing the user to add and delete items.

The last page we are going to add will contain a Master/Detail list of Art for Sale. Select CTRL-N to start the wizard, Coldfusion/Flex Application, Next, Load ColdFusion/Flex Application Settings, Next, Next (see Figure 27).

This time we are going to add the Master/Detail page to display and edit the Art for Sale.

Enter Art For Sale in the Name field and select Master/Detail page type, then select Edit Master Section (see Figure 28).

To list the names of the artists, double-click the art table and select ARTID and ARTName. Uncheck the ARTID Display column so it won't be displayed. Press Save (see Figure 29).

The World's Leading Java Resource Is Just a >Click< Away!

JDJ is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of Internet technology professionals who use Java.

SYS-CON MEDIA

ONLY $69.99

ONE YEAR
12 ISSUES

Subscription Price Includes FREE JDJ Digital Edition!

www.JDJ.SYS-CON.com

or **1-888-303-5282**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

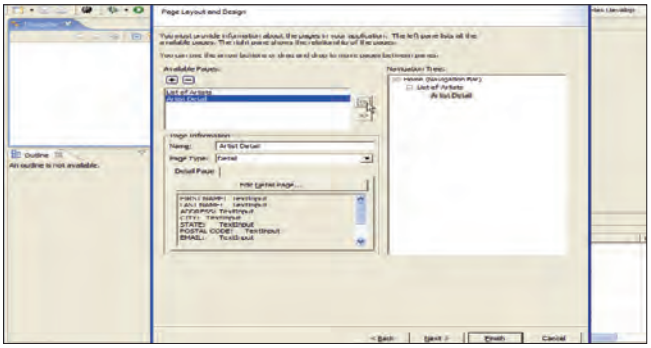


Figure 11

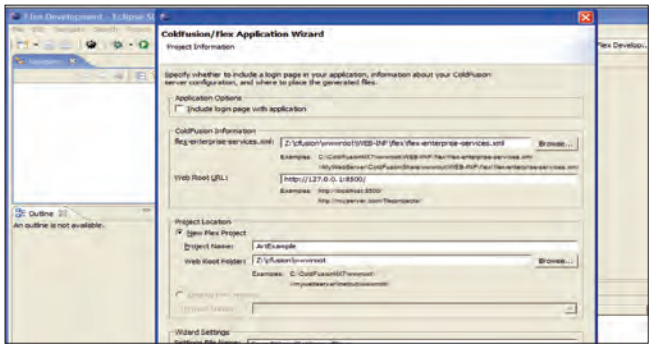


Figure 12

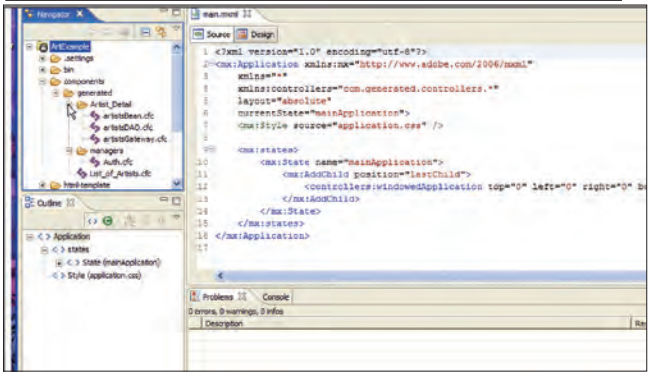


Figure 13

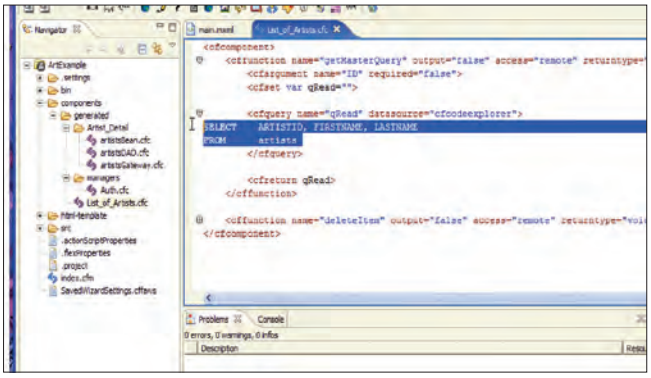


Figure 14

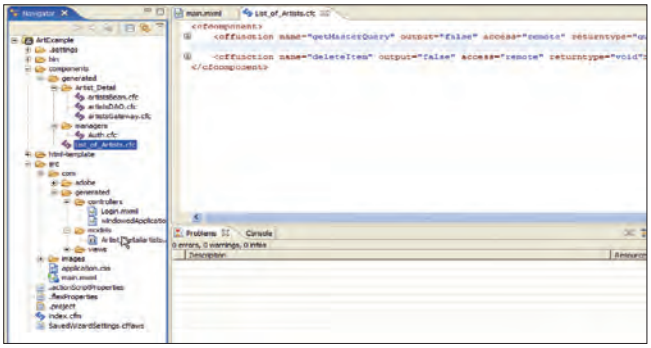


Figure 15

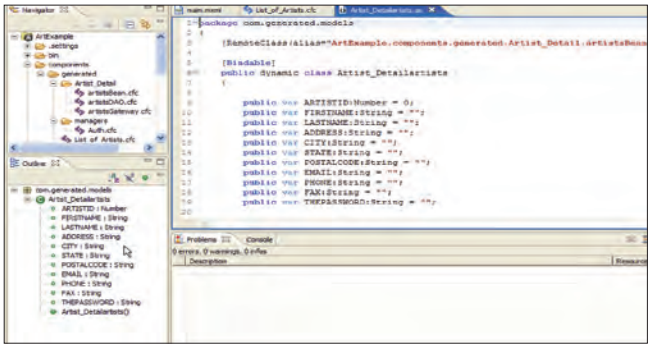


Figure 16

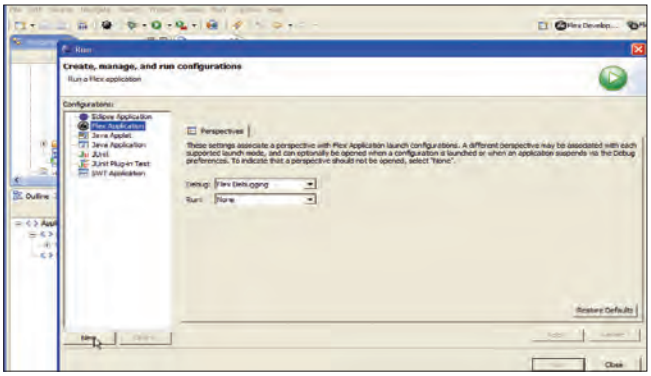


Figure 17

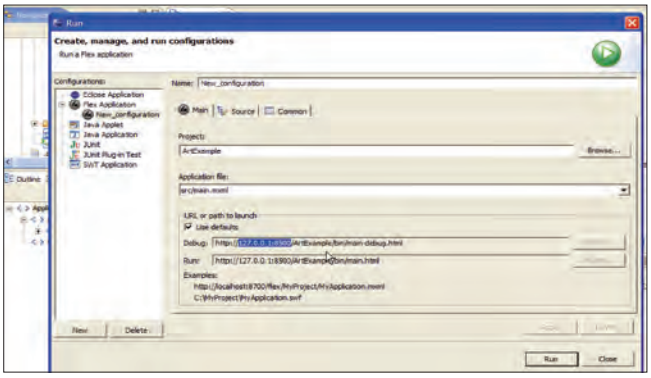


Figure 18

Figure 19

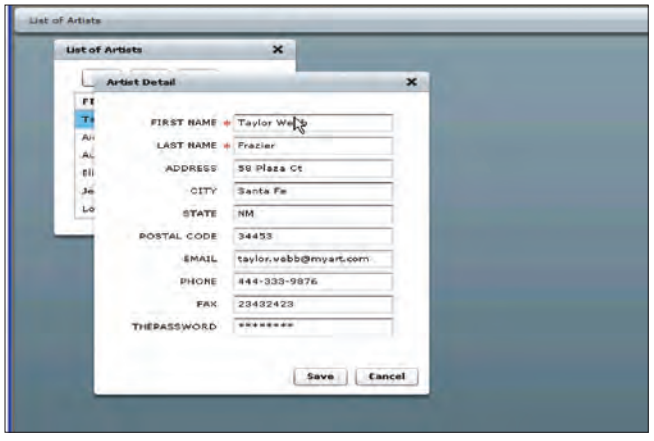


Figure 20

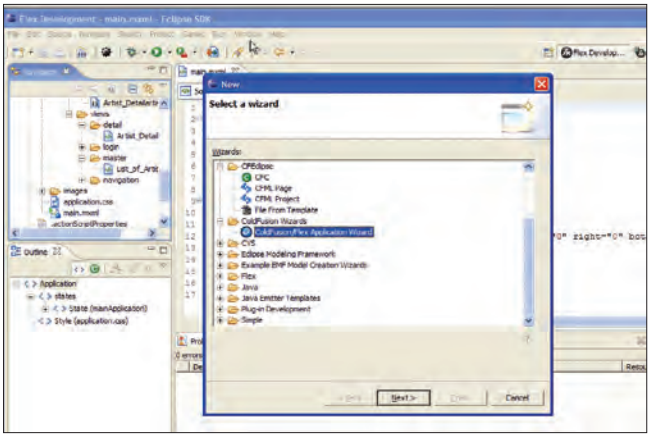


Figure 21

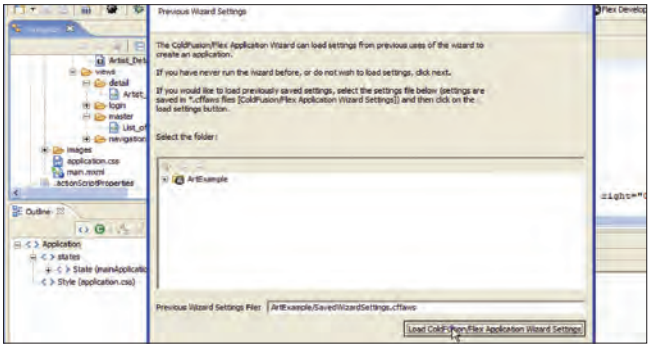


Figure 22

Press the Detail Section tab and the Edit_Detail Section Button (see Figure 30).

For the detail page we again select the art table. Under the Input Control column for ARTISTID we are going to select a ComboBox that will list the artist names. To get the names we use a sub-select by pressing the sub-select button (see Figure 31).

Here we select the artists table. We select the ARTISTID, FIRSTNAME and LASTNAME fields, but uncheck the ARTISTID

Figure 23

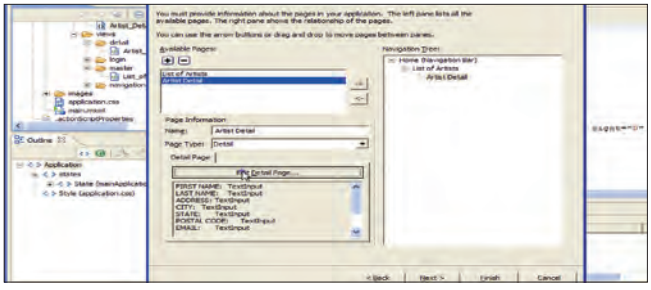


Figure 24

Figure 25

Figure 26

under the Display column to only display the artists first and last name. Press Save (see Figure 32).

Change MEDIAID Input Control to ComboBox and select the sub-select button on the right (see Figure 33).

Double-click the media table. Select the MEDIAID and MEDIATYPE. Uncheck the MEDIAID Display checkbox. Press Save.

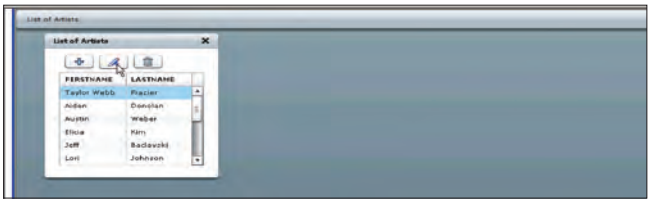


Figure 27

Figure 28

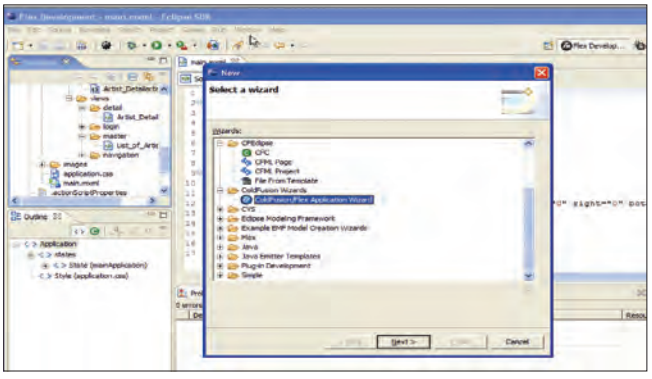


Figure 29

Figure 30

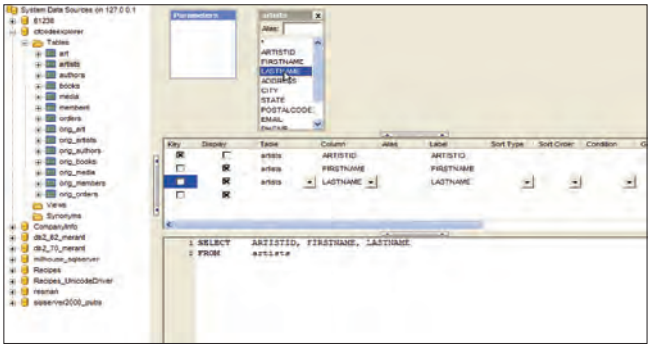


Figure 31

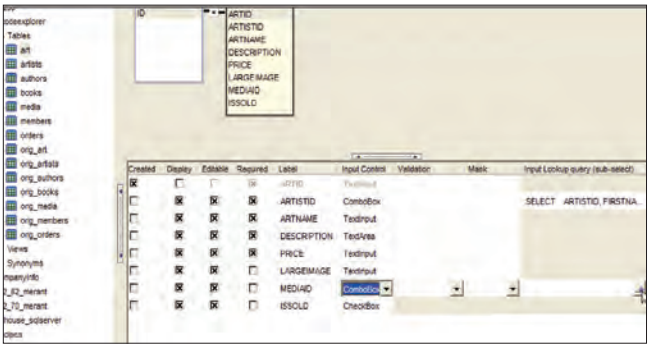


Figure 32

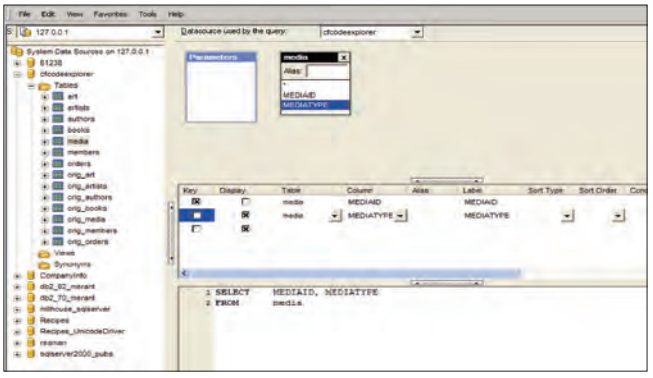


Figure 33

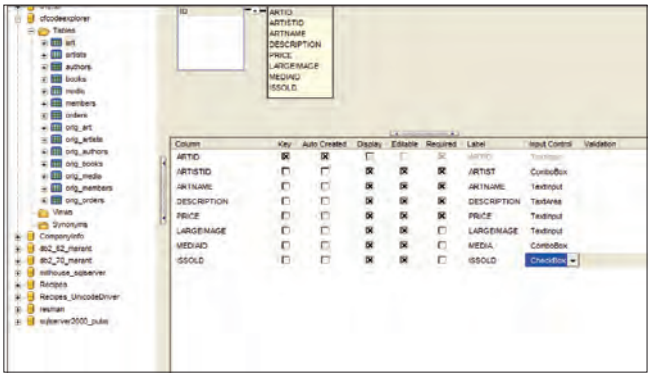


Figure 34

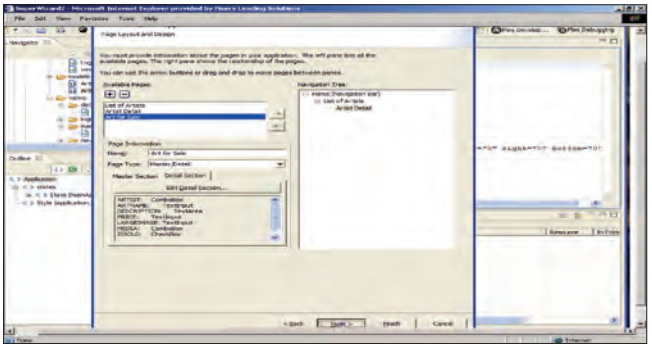


Figure 35

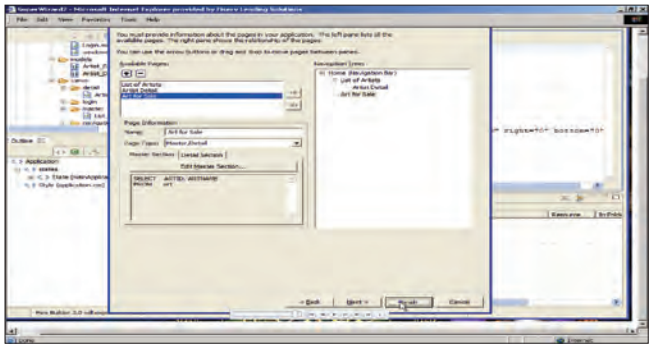


Figure 36

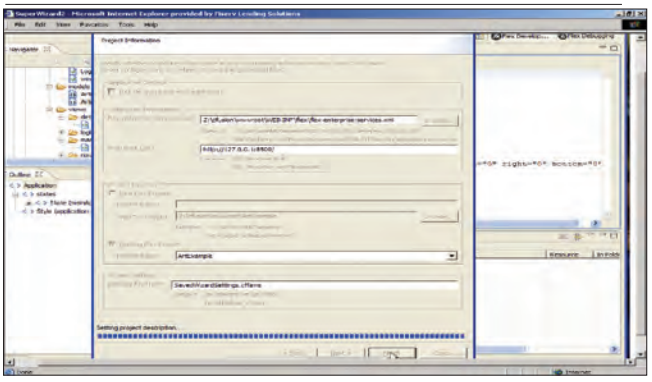


Figure 37

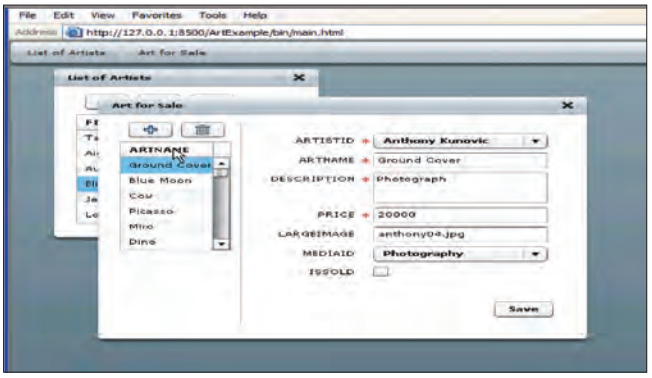


Figure 38

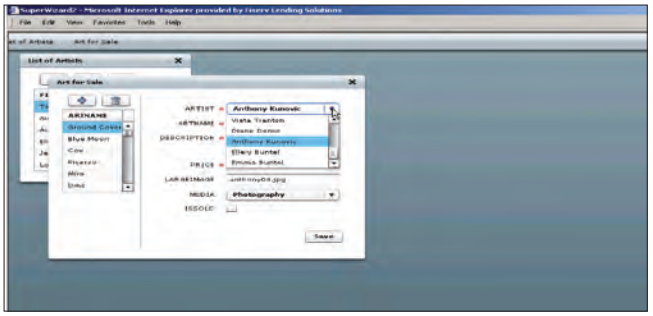


Figure 39

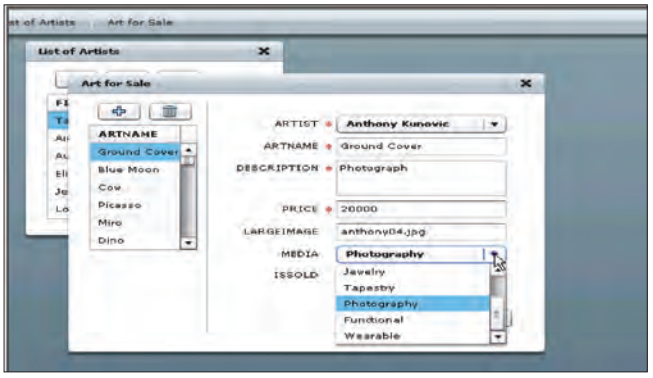


Figure 40

(see Figure 34).

Lastly, change the Input Control for ISSOLD to CheckBox. Press Save (see Figure 35

Drag the Art For Sale page to the Navagation Tree (see Figure 36).

Press Next (see Figure 37).

Press Finish. Run the program (see Figure 38).

When users select the Art for Sale tab, they will see a list of the art being sold that they can add to and delete items from along with the detail of the selected piece on the right (see Figure 39

The user can select information by artist name (see Figure 40).

The user can select information by media type.

To recap, we just learned how to use the CF/Flex Super Wizard, supplied with ColdFusion MX 7.0.2 and loaded into FlexBuilder 2.0, to create a Master, Detail and Master/Detail page using Flex, ColdFusion and Flash, "automatically." The ColdFusion/Flex Wizard is just one of the the ColdFusion extensions included in ColdFusion MX 7.0.2. Check out the others too.

About the Author

Mary McDonald has served the last 4 years as the Northern Indiana Adobe Users Group Manager (see www.ninmug.org), has attended many conferences including CFUNITED, MAX, MXNorth and TodCon. Out of her 16 years working as a software developer, Mary has worked the last 6 years with ColdFusion, SQL, Flash and Flex. She is a Certified 5.0 ColdFusion Developer.

mary_mcdonald@earthlink.net

ZoomFLEX
powered by Shado

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

Like, how long it will take you to:

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

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

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

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

STRAKER

For more information

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