

COLDFUSION Developer's Journal

ColdFusionJournal.com

August, 1999 Volume: 1 Issue: 4

Get Info on the **Allaire Developer Conference**
go to www.allaire.com/conference/
Oct. 24-26, 1999

Editorial

It's Just Getting Better
Chad Sitrler page 5

Tips & Techniques
Processing Data Files
Raymond Thompson page 24

Tips & Techniques
Safe and Sound Globally
David Schwartz page 36



IMHO
Allaire CF, Java and JRun
Jeremy Allaire
page 50

CFUG Spotlight
page 49

SYS.CON
PUBLICATIONS

TAGGING THE SERVLET BUILDING APPLICATIONS WITH COLDFUSION & JAVA



<BF on CF>: Yes, Yes, It Can Scale!
Okay, I am now officially aggravated



6
Ben Forta

CFDJ Feature:
Taking on the Dragon
Using fusebox to combat complexity



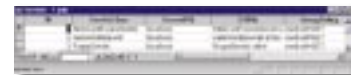
10
Hal Helms,
with Gabe Roffman & Steve Nelson

CFDJ Feature: Putting It All Together
Developing a reusable query by example system



18
Steven D. Drucker
with Robert Segal

CFDJ Feature: Tagging the Servlet PART 1
Competing and complementary technologies playing nicely together



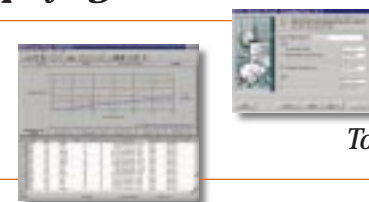
30
Ajit Sagar

What's New: Extending ColdFusion with Java Servlets and JRun



The servlet engine for developing and deploying Java servlets Edwin Smith

Product Review: e-Test Suite
A Web tool for testing e-business apps from development to deployment



46
Tom Taulli

Able Solutions

www.ablecommerce.com

Macromedia

www.macromedia.com

Interland

www.interland.net

EDITORIAL ADVISORY BOARD

STEVEN D. DRUCKER, JIM ESTEN, BEN FORTA,
STEVE NELSON, RICHARD SCHULZE, PAUL UNDERWOOD

EDITOR-IN-CHIEF CHAD SITLER
ART DIRECTOR JIM MORGAN
EXECUTIVE EDITOR M'LOU PINKHAM
PRODUCTION EDITOR CHERYL VAN SISE
ASSISTANT EDITOR NANCY VALENTINE
PRODUCT REVIEW EDITOR TOM TAULLI
TIPS & TECHNIQUES EDITOR MATT NEWBERRY

WRITERS IN THIS ISSUE

JEREMY ALLAIRE, DAN CHICK, STEVEN D. DRUCKER,
BEN FORTA, HAL HELMS, AJIT SAGAR, DAVID SCHWARTZ,
EDWIN SMITH, TOM TAULLI, RAYMOND THOMPSON

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
PLEASE SEND YOUR LETTERS TO
SUBSCRIPTION DEPARTMENT.

SUBSCRIPTION HOTLINE 800 513-7111
COVER PRICE \$8.99/ISSUE
DOMESTIC \$49.99/YR. (6 ISSUES)
CANADA/MEXICO \$69.99/YR.
OVERSEAS \$99.99/YR.
BACK ISSUES \$12 EACH

PUBLISHER, PRESIDENT AND CEO FUAT A. KIRCAALI
VICE PRESIDENT, PRODUCTION JIM MORGAN
VICE PRESIDENT, MARKETING CARMEN GONZALEZ
CHIEF FINANCIAL OFFICER IGNACIO ARELLANO
ACCOUNTING MANAGER ELI HOROWITZ
CIRCULATION MANAGER MARY ANN MCBRIDE
ADVERTISING ACCOUNT MANAGER ROBYN FORMA
ADVERTISING ASSISTANT MEGAN RING
GRAPHIC DESIGNER ROBIN GROVES
GRAPHIC DESIGNER ALEX BOTERO
GRAPHIC DESIGN INTERN AARATHI VENKATARAMAN
WEBMASTER ROBERT DIAMOND
CUSTOMER SERVICESIAN O'GORMAN
CUSTOMER SERVICE ANN MARIE MILILLO
ONLINE CUSTOMER SERVICE AMANDA MOSKOWITZ

EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC. 39 E. CENTRAL AVE.,
PEARL RIVER, NY 10965
TELEPHONE: 914 735-7300 FAX: 914 735-6547

COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
IS PUBLISHED BIMONTHLY (6 TIMES A YEAR)
FOR \$49.99 BY SYS-CON PUBLICATIONS, INC.,
39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306.
APPLICATION TO MAIL AT PERIODICALS POSTAGE RATES
IS PENDING AT PEARL RIVER, NY 10965
AND ADDITIONAL MAILING OFFICES.

POSTMASTER

SEND ADDRESS CHANGES TO:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON PUBLICATIONS, INC.
39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306

© COPYRIGHT

COPYRIGHT © 1999 BY SYS-CON PUBLICATIONS, INC.
ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE
REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS,
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY OR ANY
INFORMATION STORAGE AND RETRIEVAL SYSTEM,
WITHOUT WRITTEN PERMISSION.

FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR.

SYS-CON PUBLICATIONS, INC., RESERVES THE RIGHT TO REVISE,
REUBLISH AND AUTHORIZE ITS READERS TO USE
THE ARTICLES SUBMITTED FOR PUBLICATION.

WORLDWIDE DISTRIBUTION

BY CURTIS CIRCULATION COMPANY 739 RIVER ROAD,
NEW MILFORD NJ 07646-3048 PHONE: 201 634-7400

DISTRIBUTED IN USA

BY INTERNATIONAL PERIODICAL DISTRIBUTORS
674 VIA DE LA VALLE, SUITE 204, SOLANA BEACH, CA 92075
619 481-5928

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

It's Just Getting Better...



BY CHAD SITLER

I spent a week in San Francisco at Sun Microsystems's JavaOne show. Quite an interesting, exciting show! Personally, I spent most of my days there interviewing many of the movers and shakers of the Java industry. This was all very cool, but I missed having my usual interaction with ColdFusion developers.

Luckily, on the second day of the show, I had the opportunity to sit down with Jeremy Allaire and Paul Colton of Live Software to talk about Allaire's acquisition of Live.

Life is funny. It just so happened that the special issue of *Java Developer's Journal* for the JavaOne show has an article, written by Ajit Sagar, about using "Competing, yet complementary, technologies," those being Java and ColdFusion. This article is in the June issue of *JDJ* and I highly recommend that you get a copy. In this issue of *CFDJ* Ajit has written an article complementary to the one in *JDJ*.

Talk about growing pains... "Anyone who has checked out the *CFDJ* Web site or skipped ahead through this issue may notice that we are changing a lot in the way of design and layout. Much of this is a direct result of your feedback. You'll notice that we jazzed up the article layout quite a bit. Most important, you'll also notice an abundance of source code printed in the magazine itself. This is entirely due to your requests. The only exception is Ajit's article, due to the abundance of code he provided. As always, this and all code will be included on the *CFDJ* Web site.

I'm rather fond of this issue. I'm sure I don't need to tell you this, but take a look at Ben Forta's article. Finally, someone addresses the scalability issue with CF. And he addresses it quite well. The previously mentioned article by Ajit is a definite read. Tom Taulli reviews RSW Software's e-TEST Suite, and Jeremy Allaire talks about the history of Allaire and Live Software's merger.



Reader Feedback...

I just got the ColdFusion Journal last night here in Mexico. I made the city newspaper in ColdFusion (<http://www.vanguardia.com.mx>) last month. These guys add about 400 news items every day to the database. We had about 10,000 news items in the first month and the applications started to get slow and worse, the server memory and processors started to work more than they should. I used the cachedwith-in="#CreateTimeSpan(0,1,0,0)#" and I was expecting it to work about 20% faster after adding your example from the article. The applications work mainly on search queries that use the NEXT RESULT BUTTON and the cache is so useful for this purpose. It made my application 10 times faster than it used to be by adding this simple attribute...Keep up the good work...and thanks...you saved me from getting a bigger server.

Ing. Guillermo Dewey Guerra
Soporte Metro Cybercomercio

Picked up your mag yesterday. Love it. Read it while the kids played on the swings. Gonna subscribe (make the boss pay). I'm trying to convert all Canadians to CF. Lots of resistance from the the army of Perl. Potential Topic: can somebody please explain ColdFusion session management to me? It's got to be a great idea. I just can't figure it out. The virtual pet explanation in Forta's book leaves me feeling dizzy. I (and others I know) end up building my own system based on IP address, Time-Format(Now()) and a little database.

Mike Corbridge
Ontario Ministry of the Environment

The premier issue was passed among the different developers and we all loved it! There was a problem in your Web site that I experienced and I found the response from Chad to be prompt and refreshing. I eventually had to call because I was not clear enough about the problem location but Chad stepped up, we chatted and it was fixed right away. His dedication is obvious and I am sure the publication will go far. Keep up the great work and the issues rolling.

KD Nyegaard
ERISS Corporation

You put out a superb product. Recently I was trying to find an answer to how to update multiple records in CF and had spent four days trying to figure it out. I had received my *CFDJ* a week earlier but I did not have time to read it. As a manager and a developer, my time proves to be very precious. In any case, out of frustration one evening (around one in the morning), I just sat back and started reading my *CFDJ* to relax (strange, huh?). The first article I read was the solution to my problem! The Caras article in the Special edition, "Dynamic Variable Length Forms." I sent a message to Caras thanking him for spending the time writing the article and to the *CFDJ* Forum thanking *CFDJ* for publishing it... . Again, thanks... .

Elbert T. Shaw
Science Applications International Corporation

*As always, we love to hear from all of you.
Keep the comments coming!*

ABOUT THE AUTHOR

Chad Sitrler is editor-in-chief of ColdFusion Developer's Journal. Chad can be reached at chad@sys-con.com.

Yes, **YES**, It **CAN** Scale!

BY
BEN
FORTA



Okay, I am now officially aggravated and I'm turning this issue's column into a gripe session



Barely a day goes by without someone wanting me to reassure them that ColdFusion scales. Whether it's Web administrators who are experiencing poor application performance, partners who want to be sure ColdFusion can handle their anticipated load, developers whose bosses are threatening their jobs over failed deployments, or press and analysts positioning ColdFusion as a "low-end solution." Whichever it is, inevitably I end up in the position of defending ColdFusion and its scalability.

Well, enough is enough. I'm going to set the record straight, even if I offend folks in the process. I was one of ColdFusion's first end users and I still write lots of ColdFusion code. I've been involved in helping direct ColdFusion's development through several major

releases, and I'm the author of the best-selling books on ColdFusion. I also spend a considerable amount of time interacting with other ColdFusion developers. Without blowing my own horn too much, I think I am more than qualified to address this issue. So here goes.

Yes, like the rest of you, I have heard those statements positioning ColdFusion as an "entry-level" or "low-end" solution. And I also think I know how ColdFusion earned that reputation. Do any of you write in Visual Basic? If so, you know that "real programmers" don't take you or your code seriously. Visual Basic suffers from an image problem, even though thousands of corporations trust the language with mission-critical applications. Why is that? Simply because Visual Basic

makes application development easy, even fun – and any language that's easy just can't be a high-end tool, can it? After all, if you want to write high-end code you need expensive tools, a massive learning curve and complex language elements – or so some developers would have you believe.

And while I'm not going to get into a debate on the pros and cons of Visual Basic, I do believe that ColdFusion suffers from the exact same image problem.

The single most important aspect of ColdFusion – the feature that in my opinion has made ColdFusion as successful as it is – is its ease of use. As every one of you knows, with ColdFusion you really can hit the ground running. Most ColdFusion developers are knocking out working code within hours of installing the product – and not just experimental code. I'm talking about real working applications interacting with real live data. That's what we love about the product – how quickly and efficiently we can get tasks done. But ColdFusion's minimal learning curve and incredible ease of use is a double-edged sword. Yes, it makes our lives easier and lets us get our jobs done quicker, but at the same time, to many, it clearly positions the product as a low-end tool.

Let me put it another way. If Allaire had made the product more complicated and required a massive learning curve – if developers had to understand the technicalities of features like C's pointers, C++'s polymorphism and Java's garbage collection – if there were complex compile and link steps needed before you could test a change – well, then ColdFusion would be perceived as a high-end tool. Talk about irony.

Now in all fairness, there's some truth to this perception (for both ColdFusion and Visual Basic). Let me explain.

Data Return

www.datareturn.com

'You can use the tools in your toolbox to build a great house. You can also use those same tools to build a really terrible one. It's not the tools, it's how you use them'

Part of the reason that more complex languages are perceived as higher-end tools is that the tools are so difficult to learn that by the time developers are writing real code they've already learned the dos and don'ts of application development – that and the fact that the learning process generally tends to eliminate less experienced developers. Thus most developers who write in those languages are writing better code. But that has nothing to do with the language itself. Rather, it has to do with the developers, their experience and the kind of code they write.

And *that*, I believe, is the real issue.

Most ColdFusion developers lack the background and discipline of traditional application development. That's not a bad thing; on the contrary, it's exactly what Allaire intended. Web application development should be accessible to all developers regardless of prior experience. At the same time, when ColdFusion doesn't scale, it's completely unfair to point fingers at the product rather than to the true source of the problem.

ColdFusion development is easy; good ColdFusion development is not. Writing solid, robust and effective ColdFusion code requires that you master a diverse range of skills – from basic coding techniques, code organization,

reuse, and memory and cache management to database design and interaction. And for the record, when I have helped debug or diagnose scalability problems, 99 out of 100 times that last item has been the culprit (which is why I dedicated two entire columns to it – see *CFDJ* Vol. 1, issues 2 and 3).

A reader who sent me a message in response to my column on query caching (*CFDJ* Vol. 1, issue 2) said he had been close to dumping ColdFusion because it couldn't scale – that was until he implemented some of the caching suggestions I offered. He wrote that the site now runs 10 times as fast as it did, and he is definitely not going to dump ColdFusion any time soon. And while I'm glad I was able to help, I wonder how many other developers are blaming ColdFusion when it is definitely not the problem.

If you're concerned about scalability (and you should be), consider the following:

- *Clean up your code* – obfuscated code is hard to read and hard to maintain, and it makes ColdFusion have to work harder too.
- *Break your code into small, bite-sized chunks* – smaller code blocks can be reused easier, they can be cached more effectively and they help avoid code obfuscation.
- *Use the tools the language offers* – arrays and structures are a little harder to use than simple variables, but using them correctly can improve code performance dramatically.
- *Optimize database activity* – from the right relational database design, optimized SQL, and the appropriate use of keys and indexes to the right kind of hardware and caching configurations – all of these impact scalability.
- *Eliminate unnecessary database access and processing* – caching, caching, caching – the sites that scale best are designed not to hit databases or perform complex processing unless absolutely necessary.
- *Perform regression testing and load impact analysis before deploying your application* – the fact that an app works on your desktop doesn't mean it'll work well under a heavy load. Plus it's highly unlikely that the way you tested the app and the way end users use it will be the same. The application will probably be stressed where you least expect it.

Use ColdFusion's performance monitoring and logging options – hey, if ColdFusion is telling you there's a problem, the least you can do is listen.


- *Set up server clusters.* Machines go down – it's a fact of life. Always strive to eliminate any single point of failure. ColdFusion Enterprise comes with clustering technologies that may be used on their own or in conjunction with hardware-based solutions to ensure uptime.

This is just a high-level list of ideas, and the list is definitely not complete (although it's potential fodder for future columns).

I've seen some truly great code written in ColdFusion – two exceptional examples within the past few weeks alone (I'm not going to mention company names; I have no permission to do so. Suffice it to say that both are high-visibility public e-commerce sites with highly recognizable branding). Both were developed in ColdFusion by developers with extensive development experience and the ability to apply that experience to ColdFusion development. I've also seen some truly horrid ColdFusion code (heck, I even wrote some of it myself, back when I was first starting). Great code and horrid code – both are possible, both are doable and both are fully supported by ColdFusion.

I'm not saying you can't be a good ColdFusion developer without prior development experience. But I am definitely saying that if you think you'll be knocking out bulletproof code minutes after installing ColdFusion for the first time, you're sadly mistaken. Don't confuse the euphoria you experience from seeing your first apps work with the confidence that you're ready to roll out production-quality code.

Here's the bottom line. The next time you're running into scalability problems, remember this: ColdFusion is not an application, it's a toolbox. You can use the tools in your toolbox to build a great house. You can also use those same tools to build a really terrible one. It's not the tools, it's how you use them.

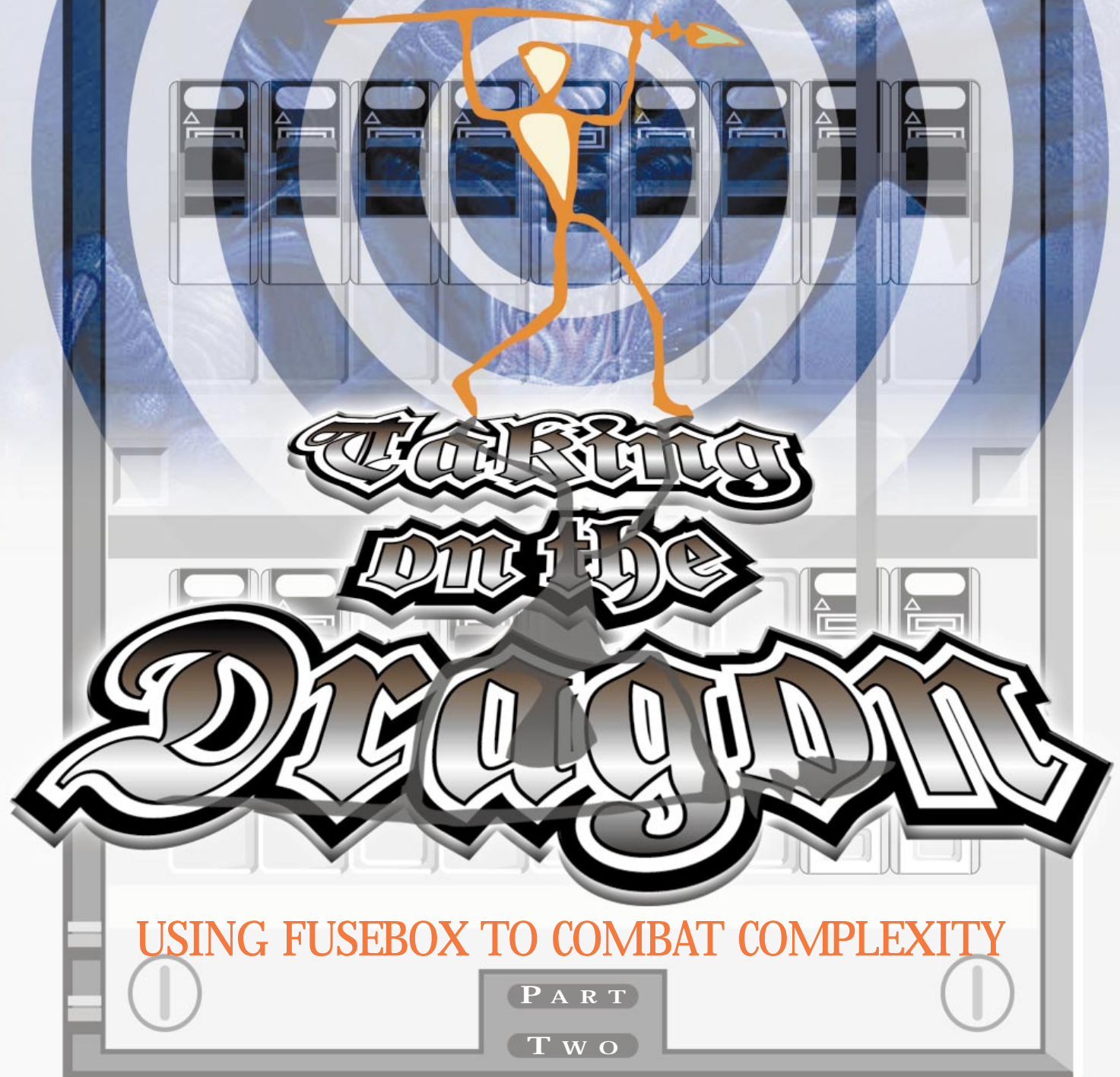
And that's why I titled this column ColdFusion “CAN Scale,” not ColdFusion “DOES Scale.” Yes, ColdFusion CAN scale. Will it? Well, that's up to you. 

ABOUT THE AUTHOR

Ben Forta is Allaire Corporation's Product Evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit and its sequel, Advanced ColdFusion 4.0 Development (both published by Que), and he recently released Sams Teach Yourself HomeSite 4 in 24 Hours.

Macromedia

www.macromedia.com



by Hal Helms with Gabe Roffman and Steve Nelson

In the last issue of *CFDJ* (Vol. 1, issue 3) I spoke of the problem of complexity and how it threatens our best efforts at software development. With the need for software greatly increasing – especially Web applications – we need to find ways to manage development and tame the complexity monster.

ColdFusion is a powerful ally in this quest. Its simple tag-based syntax shields programmers from much of the minutiae of programming and speeds development of Web applications. No, it's not a magic bullet, but when combined with Fusebox application development methodology (www.fusebox.org), ColdFusion can make programmers

far more productive and lets them use their scarce resources better.

In this article you're going to walk through building an actual application (albeit small), and you're going to hear from a couple of developers who used Fusebox exclusively on a large e-business project. First, here's a quick overview of Fusebox.

Fusebox: Building on a Hub-and-Spoke Architecture

A Fusebox application uses a hub-and-spoke architecture. A single ColdFusion file forms the hub of our system. In a sense, this file or fusebox is the application as it determines what the system can and will do.

Individual files (called fuses) are used by the fusebox to accomplish discrete tasks. Ideally, the fuse is constructed so it can be used in any number of applications without change.

The Fusebox application works by receiving a request for a (fuse)action and then calling on the appropriate fuse to do its job. Having accomplished its task, the fuse returns the action to the fusebox.

You can view a Fusebox application like a modern stereo device. The fusebox file is like the schematic diagram; to the trained eye it provides a maximum amount of information in a very small space. To carry the analogy further: the individual stereo pieces use many of the same individual components. Though they differ greatly in function, a stereo receiver and an amplifier are made of similar parts, such as diodes, transistors and resistors.

Simplicity Is Key

Because these parts have been designed to handle a very specific task, they're generic in nature and can be used in any number of electronic devices. The designer of a stereo amplifier doesn't have to find (or invent) "amplifier diodes" that are different from "receiver diodes." Imagine the havoc that would result if every new device was wholly different from the last. This is the world most programmers have grown up in. Familiarity with the landscape around us has made us fail to recognize how unacceptable we would find such a state in other areas of our lives. It's no wonder software projects have such high failure rates.

Fusebox was designed to break down complexity by offering tools and methods that cut the complex whole into more manageable parts. Far from being an obscure, overly complex methodology, Fusebox greatly simplifies the programmer's life. Like all good methodologies, it exists not to bind users but to empower them. As you go through this article with its rules and instructions, remember that rules are a shortcut to help you master Fusebox; when they no longer serve you, feel free to change them.

Examining a Fusebox

The center or hub of the application is the fusebox file (which is how the methodology got its name). The ColdFusion file, a model of simplicity, is made up of only two sections. The first section is responsible for a *definition* of variables that will affect the entire application. You can either define each variable in the fusebox itself or `<cfinclude>` a page that holds this information. In large projects you may want to do the latter so people can make changes to this information without taking the risk of opening the fusebox itself. You can view the definition section of a fusebox in Listing 1.

The second part of the fusebox is responsible for *direction*. At any one time, the application has one and only one fuseaction. This type of architecture is sometimes called *state transition* because the application passes through a series of states brought about by different user actions. Each of these actions – clicking a button, entering information in a form – is associated with a fuseaction. A single `<cfswitch>` statement determines the value of this variable that directs the program flow. The variable "attributes.fuseaction" can be examined at any point in debugging a running program to see exactly what state the system is in. You can examine a section of this code in Listing 2.

'In the end,
our best
weapon
against
complexity
is beauty'

—David Gelertner

The Default Fuseaction

You'll want the fusebox to be called automatically when a user sends out an HTTP request to the appropriate directory. Instead of your users being burdened with the need to type in something like "www.yourDomain.com/myApplication/index.cfm?fuseaction=login", a default fuseaction tells the program, "When no other fuseaction has been specified, use this one." This makes it ideal for holding the fuseaction that causes a user to log in, although you can use any fuseaction the fusebox can handle.

Notice in Listing 1 that a call has been made through a `<cfmodule>` tag to a custom tag, "FormURL2Attributes.cfm". I mentioned the reason for this in the last article, but it bears repeating. One of the design specs for Fusebox was the ability to call applications as a single custom tag. What happens when you call a custom tag? ColdFusion takes any variables passed and makes them part of the attributes collection. To reference them, you must use the "attributes." prefix. But how will fuses know whether a variable passed to them is a form variable or a URL variable, or has been converted to an attributes variable?

To spare our small-minded fuse the confusion this would cause, we use the custom tag "FormURL2Attributes" to convert both form and URL variables into attributes variables.

With this done, our fuse can rest assured that *all* variables it encounters will be attributes variables, making for a happy fuse. And a happy fuse is a productive fuse.

A Fuse Under the Microscope

Take a look at a small (but important) fuse, `act_Decide.cfm`. This fuse encapsulates the advanced decision-making techniques used by many managers today.

```
<!-- act_Decide.cfm -->
<!--FUSEDOC
'      Responsibilities: I accept an employee request and
then give the user the SAME answer their manager will give
them. I use a highly accurate algorithm.
'      Author: hal.helms@utaweb.com
'      Required Attributes: question
'      Optional Attributes: supplication (e.g. Please)
'      Return Attributes: N/A
'      Return Fuseaction: RFA if answer is yes, RFA2 if answer
is no
' END FUSEDOC-->
```

```
<cfoutput>
<!-- Decision-making algorithm -->
<cfloop from="1" to="1000" index="pos">
  <cfif Val(2*pos) NEQ Val(pos + pos)>
    <cfset theAnswer = "yes">
    <cfset theURL = "#RFA#">
  <cfelse>
    <cfset theAnswer = "no">
    <cfset theURL = "#RFA2#">
  </cfif>
</cfloop>
```



```
After careful consideration, the answer
to your request (#attributes.question#)
is #theAnswer#. You can click <a
href="#Self?fuseaction=#theURL#">here</a
> for an in-depth explanation of the fac-
tors that went into the decision.
</cfoutput>
```

You know enough about Fusebox now to deduce that the variable “self” must be another name for the fusebox, since all action must return to the fusebox. The purpose for aliasing the fusebox as “self” involves the details of how Web servers work. If you write the code with the fusebox filename hard-coded, what will happen if your client’s Web server doesn’t recognize your fuseaction file as the one it serves up by default? By making a parameter of the initial file that’s called, you avoid the possible need for global search-and-replace operations.

The concept of return fuseactions is similar. The “prime directive” for fuses is “always return to the fusebox” – accomplished with a variety of methods (see the last issue’s article). But what happens after the fuse returns the action depends on the individual application – and it may also depend on some action that occurs while control rests with the fuse. In other words, the fuse may have several possible exit states depending on what occurs (e.g., fuse successfully performs its function, fuse canceled by user action, action canceled by internal error, etc.).

While return fuseactions aren’t an explicit part of the Fusebox specifications, you may find the concept of return fuseactions very useful if your goal is to create generic, reusable code. If you choose to implement this scheme, your fuse will define in what ways it can exit and the name of a variable for each exit state. It’s a good idea to put this information in the documentation header of the fuse itself.

Understanding Program Flow

You have now seen all the elements of Fusebox. Now it’s time to walk through the actual execution of a Fusebox application. For this example, the fusebox (index.cfm in this case) and all associated fuses are in a directory called “AI”. The sole subdirectory is “images.” I created a default fuseaction to deal with the case where the application is initially called. In this case I’ve defined the default fuseaction to be “login.”

```
<!-- Provide a default fuseaction -->
<cfparam name="attributes.fuseaction"
default="login">
```

Now when the user requests www.yourDomainSpace.com/ with no fuseaction specified, the fusebox will use the default “login.” The fuse card for the fuseaction “login” looks like this:

```
Fuseaction: login
Fuse: dsp_Login.cfm
Responsibilities: I provide a form for
the user to enter his/her username and
password.
Required attributes: DSN
Optional attributes: CSS
Return attributes: userName, password
RFA: RFA
```

This tells the programmer everything he or she needs to know to build the fuse. In fact, by describing ahead of time the specifics of the fuse, you can break the “critical path” of the fuse-building process into several tasks that can be done simultaneously. In a simple fuse like this the time saved is minor, but on larger jobs, having the ability to add resources to reduce the time to navigate the project’s critical path can mean the difference between success and a learning experience.

Now it’s time to build your first fuse.

```
dsp_Login.cfm
<body>
<cfform ACTION="#self?fuseaction=#RFA#"
METHOD="POST">
<table BORDER="0" ALIGN="center" CELL-
PADDING="3" CELLSPACING="1">
<tr>
<td COLSPAN="2" ALIGN="center">User
Login</td>
</tr>
<tr>
<td>User Name</td>
<td><input type="text" name="user-
Name"></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="pass-
word"></td>
</tr>
<tr>
<td COLSPAN="2" ALIGN="center">
<input type="submit" value="Submit">
</td>
</tr>
</table>
</cfform>
</body>
```

(Note: What isn’t shown here is the Fuse-doc header. This provides information about the fuse that can be pulled automatically from the fuse to create real-time online documentation, which is the subject of another article.)

The next link in the application chain is validating the information the user has provided against a list of approved users and passwords. Here is the fuse card for validateLogin.

```
Fuseaction: validateLogin
Fuse: act_Login.cfm
Responsibilities: I check the info sent
```

```
to me against a database of approved
usernames and passwords. I have two
exit states: RFA for a successful login;
RFA2 for a bad login.
Required attributes: DSN, userName,
password
Optional attributes:
Return attributes: userID
RFA: RFA if successful else RFA2
```

The code is shown in Listing 3.

Some fuses can provide for multiple RFAs, depending on the results of the fuse’s operation. Here, the act_validateLogin fuse will use RFA if the login is successful. If not, it uses the value of the variable RFA2.

Benefits of Fusebox Methodology

I mentioned earlier that usually the cost of maintaining code exceeds the initial development cost. Using RFAs can help maintain code by insulating fuses from the changes made to other fuses. This is the idea behind object-oriented programming, and it also works well in the Fusebox world. For example, if you later move from using a database of names and passwords to an LDAP server, your changes can be made in this fuse only. Reducing dependencies reduces complexity and makes applications more durable and robust.

RFAs are parameters passed into the fuse. Like all other information in the Fusebox architecture, they come from the fusebox.

```
<cfcase value="login">
<cfset attributes.css = "normal.css">
<cfset attributes.RFA = "validateLogin">
<cfinclude TEMPLATE="dsp_Login.cfm">
</cfcase>

<cfcase value="validateLogin">
<cfset attributes.RFA = "showUserMenu">
<cfset attributes.RFA2 = "badLogin">
<cfinclude TEMPLATE="act_Login.cfm">
</cfcase>
```

The entire structure of an application can be done by an experienced programmer using fuse “stubs,” which can then be given to more junior programmers. Since the interface is well defined, programmers can concentrate on a single task without worrying about what other parts of the program will be affected by their work.

This division of labor cuts development time and costs. It protects the application from the condition sometimes known as “infinite bugs,” which occurs when a programmer makes a change to one part of the code and unwittingly affects other portions. In complex projects this can quickly become a vicious cycle. In addition, with Fusebox, programmers are able to work at the level of their expertise without inheriting the need to do all the tasks in the application.

Intr@Visiontm

**BACK OFFICE
IN A BOXsm**
WITH

With

- **Feature-rich functionality right out of the box**
- **Standard user authentication model**
- **Supercharged with IVOTM – Intr@Vision Object architecture**

Nutrition Facts

Serving Size: 1 "In Your Face Intranet"
Servings per container: ∞

Intr@Vision Hand-Built

Lines of code for security	1	who knows
Extensible	100%	maybe
Time to complete	a little	a lot
Cost to complete	a little	a lot
Reusability	100%	0%
# of hi-octane colas required	1	100+
# of slices of pizza required	3	100+
# of all-nighters	0	10+

Ingredients: Intr@vision FoundationTM is just that: the foundation for your entire intranet. In combination with ColdFusion[®], you get one of the most powerful rapid application development environments for building out the rest of your intranet. You can focus your energies on developing business solutions and not on the infrastructure.

Serving Suggestion:

Serve on top of one cup ColdFusion[®], add two cups business process, stir gently.

Allow business processes to rise to the top.

Sit back, relax and enjoy the fruits of your well-chosen development framework –
Intr@Vision

Go to
www.galileodev.com
to sign up for your
free sample,
or call 770-643-9176.

New for Fall '99

- **Time Tracking**
- **Expense Reporting**

another quality
product from



Sharpening Your Pencils...

Now it's time for you to put everything you've learned into practice and actually build a Fusebox application! In the last article we had a meeting with Tom and his group. From the feedback they provided I've distilled the general requirements for the application (see Table 1). Next to each one I've identified the appropriate fuseactions and fuses.

Making the Fuse Cards

Fuse cards come next. To help get you started I've done the cards for newActionItem and sendMessage.

Fuseaction: newActionItem

Fuse: dsp_newActionItem.cfm

Responsibilities: I give the user a form to create a new action item. I have two exit states: RFA to continue; RFA2 to cancel.

Required attributes: userID

Optional attributes: CSS

Return attributes: selectedUsers, actionItemText

RFA: RFA if successful else RFA2

Fuseaction: sendMessage

Fuse: act_SendMessage.cfm

Responsibilities: I send a message off to selectedUsers through e-mail.

Required attributes: DSN, userID, message, selectedUsers

Optional attributes:

Return attributes:

RFA: RFA

I suspect that as you begin working with this you'll see some omissions in the application structure shown and you'll want to modify it. Changes can occur to either the interface specification or the interface implementation of a fuse. I'm using the word interface not to refer to the user interface, but to the agreed-upon ways that different components of the system interact with each other. For example, the interface for validateLogin specifies that it expects a username and password. Whether it validates this against a database, an LDAP server or some other mechanism is an implementation decision.

One of the benefits of the Fusebox structure is that it allows for changes to occur in the implementation. You can change and add to your application – so long as the interface, on which other parts of the application depend, doesn't change.

In contrast, changes to the interface specification should be done only in the architectural phase. It's expected that during this phase – while you're creating fuse cards and laying out the overall architecture – you'll see areas you've missed, or simply change your mind about how to handle a fuseaction. This is part of the process. The more experience you have using Fusebox, the quicker you'll be able to adapt this methodology to your own style.

Young programmers will often skip lightly over this stage, anxious to begin coding. This is a mistake; once the architecture is finalized, the interface should be treated as golden. Any changes are likely to create havoc with the application. (You can frame this and use it as an ally the next time your manager requests a "minor" slipstream change!)

A Little More Help

Now it's your turn to finish the application. A great writer was once asked what he feared most in the world. Without hesitating he replied, "A blank piece of paper." And it was the great Albert Einstein who once remarked that "learning by example isn't one way of learning; it's the only way." In that spirit I offer you some more help in accomplishing this,

although you'll probably learn more if you try it on your own for a bit before taking advantage of my help. Regardless, you can find a fully functioning stub application with database, as well as some complete sample fuses at www.teamAllaire.com/hal.

Break Time

Learning new things means keeping an open mind, which is heroic work! So you deserve a little encouragement along the way. Let your brain cells relax as I talk with Steve Nelson and Gabe Roffman about the development of a large e-business site they just finished. It was done using only Fusebox methodology and structures. Another magazine is famous for their...interviews. Think of this as one of those interviews – without the pictures.

Requirements	Fuseactions	Fuses
User logs onto system	login validateLogin	dsp_Login.cfm act_ValidateLogin.cfm
Add a user	newUser addUser	dsp_NewUser.cfm act_AddUser.cfm
Retire a user	retireUser	act_RetireUser.cfm
Give a user administrative privileges	editUser updateUser	dsp_EditUser.cfm act_UpdateUser.cfm
Create a new action item	newActionItem addActionItem	dsp_NewActionItem.cfm act_AddActionItem.cfm
User accepts action item assigned to them	none needed default condition	—
User rejects action item assigned to them	rejectActionItem	act_RejectActionItem.cfm
User wants information on action item assigned to them	requestInfo sendMessage	dsp_RequestInfo.cfm act_SendMessage.cfm
User wants information on action item assigned by them	requestStatus sendMessage	dsp_RequestStatus.cfm act_SendMessage.cfm
Update the action item's history	none needed – different fuseactions will handle this themselves	—
View the action item's history	showHistory	dsp_ShowHistory.cfm
Complete the action item assigned to you	completeActionItem sendMessage updateAssignment	dsp_CompleteActionItem.cfm act_SendMessage.cfm act_UpdateAssignment.cfm
Delete the action item assigned by you	deleteActionItem sendMessage updateActionItem	dsp_DeleteActionItem act_SendMessage.cfm act_UpdateActionItem.cfm
Reassign an action item sent	reassignActionItem sendMessage updateAssignment	dsp_ReassignActionItem.cfm act_SendMessage.cfm act_updateAssignment

TABLE 1: General requirements for a Fusebox application

HostPro

www.hostpro.com

Steve: Trying to read and understand how and why other developers in your team did something. Trying to find and

Gabe: Right. Time was extremely tight. We had certain milestones we had to meet for meetings with investors. And the

Hal: Have either of you ever done projects of this size before – and if so, how did your experiences with Fusebox and without Fusebox compare?

Steve: I've personally done a number of similar projects – especially as it relates to working in a distributed development environment. Programmers are notoriously bad at documenting and commenting their code. And in a large project, documenting gets pushed off to the end. So without Fusebox it would take me hours – even days – to figure out what was going on as new code was added. But with Fusebox it's like looking at the architectural drawings for a house. With a little practice you can see what's happening very quickly.

Gabe: Yes, when we first came up with the Fusebox idea, we didn't think of this particularly, but it's turned out to be a huge benefit. I've done large jobs before, and using what I would call an object-based approach – like Fusebox – we were able to modularize just about all the functionality on the site so that we were never repeating or retyping the same code in different parts of the site. When we decided to make changes that would speed up processing or increase functionality, we'd only make that change to one module instead of having to track down in the application code where the change affected other parts.

Hal: What other benefits did you see from using Fusebox?

Steve: Reduced development time. Lower costs.

Gabe: Easier maintenance of the code as we developed it.

Steve: Yes. Software always changes. New people are added to the team who have new ideas. Instead of new people spending all their time trying to figure out the code base to date and commenting on how much better they would have written it, they can be given a problem to solve and work on that.

Gabe: Having an agreed-upon architecture based on Fusebox definitely helped. It made deciding on how to build something clear. And that's sometimes the longest part of a project! Also, it made it really easy to turn over the project to the customer's in-house programming staff. They didn't work on developing the original site at all, but they were able to expand the architecture because they could grasp the structure with which it was created.

Hal: Okay. What changes would you make in retrospect?

Gabe: I would definitely use Fusebox again, but I would want to figure out a better way to work with the designers than

what we had. All the communication was done by e-mail. I think it would have made life easier if we had set up something like Allaire Forums and used that.

Hal: Steve, what about you?

Steve: Well, I was working on three jobs simultaneously. It was just one of those weird things where everything I bid on, I was getting. But they all had to be done overnight. It was a blessing in disguise...or something.

Gabe: Yeah, for a while our office was starting to look like a set from *The Shining*. I kept expecting Steve to start writing on the walls.

Steve: What was it? All work and no play makes Homer – something something.

Hal: Go crazy?

Steve: Don't mind if I do!

Gabe: I see it's time for somebody's m-e-d-i-c-a-t-i-o-n.

Steve: No fair! What did he say?

Hal: Thanks, guys. 

About the Author

Hal Helms lives in Atlanta, Georgia, where, as chief technologist for User Technology Associates, he heads a team of developers working to simplify Web development. He can be reached at hal.helms@utaweb.com.

hal.helms@utaweb.com

FINDaHOST.com

coming soon

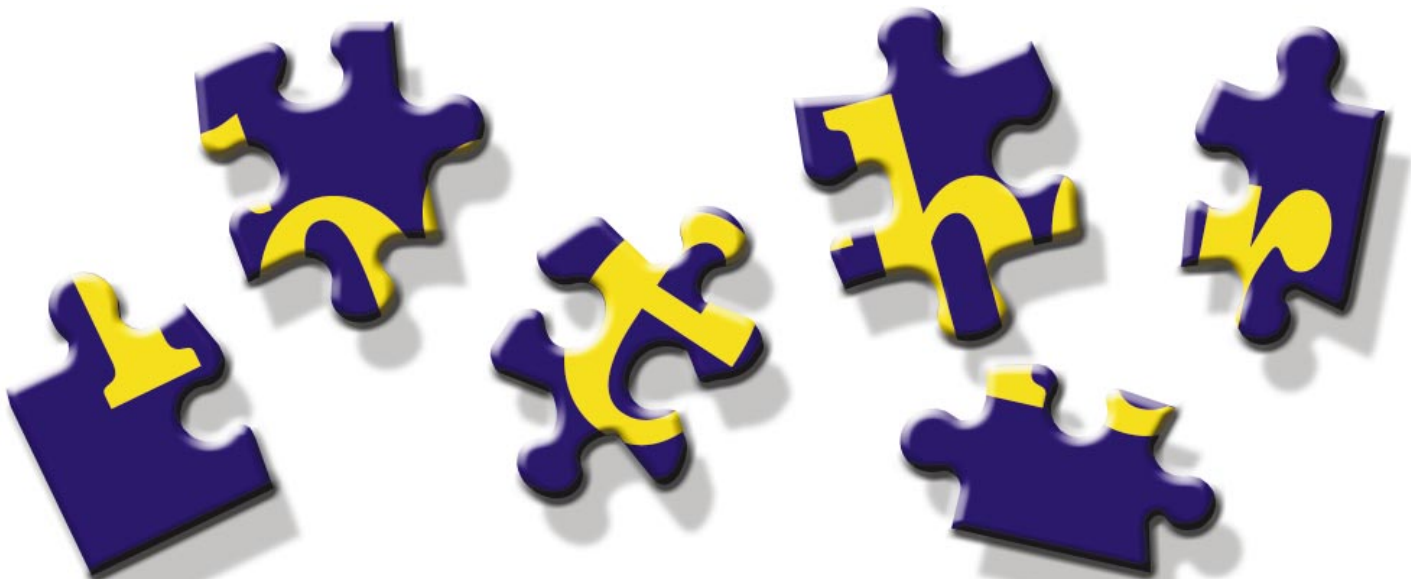
- RATEaHOST.com
- FINDaDEDICATEDserver.com
- RATEaDEDICATEDserver.com
- FINDISPs.com
- RATEanISP.com

FINDaHOST.com is the premier directory for quickly finding web hosting providers and seeing how well they compete against each other. With hundreds of listings and customer ratings available, you will be able to make an informed decision before moving your web site to their servers.

www.FINDaHOST.com

a division of  **cdR** interactive Network

Putting It All Together



Developing a Reusable Query by Example System

A common function of ColdFusion applications is the query-by-example interface (QBE) that allows end users to select from a list of properties in order to find matching records. It generally involves creating a simple HTML form. Based on end user input into the form, you construct and execute a SQL query on an action page, displaying the results to the user.

Creating such a mechanism in ColdFusion is fairly perfunctory. During the last three years of developing with CF, I must have created 5,637 such interfaces. Perhaps I'm a slow learner or a glutton for punishment, but finally, at number 5,638 I decided to call it quits. I figured there had to be a better, faster solution and steadfastly refused to code another QBE.

The challenge was laid out before me. How do I create a cross-platform reusable query component with a customizable interface that could be deployed in any application? Well, the result is depicted in Figure 1. The coding methodologies involved integrating nested frames, nested CFML custom tags, CFWDDX and copious amounts of JavaScript. The coding methods involved in this example are complex enough to make one developer exclaim, "That thing is the QBE of Death!" The name stuck.

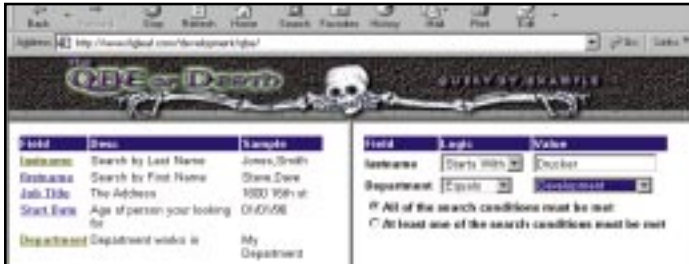


FIGURE 1: Query-by-example interface

Key Technology: A Brief Tutorial on Custom Tags

ColdFusion 3.x allows us to create reusable components built from CFML with protected variable scoping. CF custom tags can't overwrite the contents of variables from the calling page unless explicitly ordered to do so. This innovation has spawned, at last count, over 450 reusable tags available for download from the Allaire tag gallery (www.allaire.com/tag-gallery/). Most are royalty free and unencrypted. Invoking CF 3.x custom tags can be accomplished by using one of the following conventions.

Method 1: Using Simple Syntax

<CF_mytag [optional parameter list]> where mytag is the name of the custom tag file with a .CFM extension and the optional parameter list contains a series of name and value pairs. For example, running the file alert.cfm and passing a parameter named message involves the following syntax:

```
<CF_alert message="Red Alert!">
```

ColdFusion will find and execute the alert.cfm located in either the same directory as the calling page or in a subdirectory of the CustomTags directory of your ColdFusion install directory.

Developing advanced database applications with ColdFusion is deceptively simple

Method 2: Using <CFMODULE> with the Template Attribute

The <CFMODULE> tag allows you to explicitly specify the URL path of your custom tag. If your custom tag was located one directory up from the calling directory, the invocation syntax would be the following:

```
<CFMODULE template="../alert.cfm"
MESSAGE="Red Alert">
```

Method 3: Using <CFMODULE> with the Name Attribute

Using <CFMODULE> with the name attribute allows you to specify the location of a custom tag underneath the CustomTags subdirectory using dot notation. Calling a file located in the C:\cfusion\customtags\javascript\popups directory resolves to the following:

```
<CFMODULE name="javascript.popups.alert"
MESSAGE="Red Alert">
```

For the custom tag to "see" the parameters, they must be prefaced with the attributes prefix. The custom tag depicted in Listing 1 receives the parameter and generates a simple JavaScript alert box.

```
<CFPARAM NAME="attributes.message"
DEFAULT="I have nothing to report">
```

```
<CFOUTPUT>
<SCRIPT LANGUAGE="JavaScript">
    alert("#attributes.message#");
</SCRIPT>
</CFOUTPUT>
```

While these calling methods allow for the easy passing of a fixed number of parameters and values, they're not well equipped to deal with the variable number of associated parameters we might encounter when deploying a reusable QBE. Fortunately the enhanced tag architecture found in ColdFusion 4.x allows for the creation and invocation of multiple nested subtags. Using this new feature, we can create a series of integrated custom tags providing virtually unlimited flexibility, as described in Listing 2.

Using <CFMODULE> to invoke nested custom tags is largely undocumented, however, it yields more flexibility in determining the location of custom tags. Listing 2 can be rewritten using the following structure:

```
<CFMODULE TEMPLATE="qbe.cfm" [tagparams] >

<CFMODULE TEMPLATE="querycriteria.cfm" [tag params]>
...
...
...
</CFMODULE>
```

ColdFusion will find and execute the alert.cfm (see Listing 1) file located in either the same directory as the calling page or in a subdirectory of the Custom Tags directory of your ColdFusion install directory. During this operation, the variable thistag.executionmode is equal to the text string "start." CF then executes each <CF_QUERYCRITERIA> subtag in turn (see Listing 2). When the </CF_QBE> tag is reached, the QBE.CFM file is reexecuted; however, at this point the variable

thistag.executionmode is equal to the text string "end." The QBE tag accommodates this dual-execution process using the <CFSWITCH> construct as depicted below.

```
<CFSWITCH expression="#thistag.execution-
mode#">
  <CFCASE value="Start">
    <!-- do processing from <CF_QBE> Call
    -!
  </CFCASE>
  <CFCASE value="End">
    <!-- do processing for </CF_QBE> Call
    -!
  </CFCASE>
</CFSWITCH>
```

Key Technology: Nested Framesets

The <CF_QBE> interface as depicted in Figure 1 is split into three sections. The top left frame displays a list of columns that the user may select to form the basis of his/her search. Once s/he makes a selection, the data field, along with the appropriate boolean logic constructor, appears in the top right frame. The nested frameset definition, depicted in Listings 3 and 4, allows the bottom submit/requery button to persist on the screen throughout the process of executing and refining the search criteria. It also allows us to keep field selections persistent by storing them as WDDX JavaScript arrays in the button frame while destroying and re-creating the contents of a data frame, thus decreasing reliance on cookie support and session variables memory overhead. Of course, one drawback to this approach is that if the user should hit the browser RELOAD button, all their selections will be lost.

Key Technology: JavaScript Document.write()

Both Netscape 2.0+ and IE 3.x+ support what I term "poor man's DHTML." While dynamic HTML allows you to programmatically change the contents and format of a document without initiating a full-page reload, its utility is diminished since only advanced browsers support it. Worse still, the syntax varies significantly between IE and Netscape. You can, however, simulate cross-platform client-side redraws through the JavaScript document.write() method. In many instances, using document.write you may be able to reduce or eliminate calls to the Web server, since a page's contents are coming not from the Web server but from within the browser itself. This results in faster application performance and enhanced overall scalability. Listing 5 details an excerpt from the QBE drawCriteria() function which is responsible for continually updating the search criteria frame whenever the user makes a selection. The JavaScript open() method clears out a frame to accept data. Next, I programmatically out-

put the ubiquitous <HTML> and <BODY> tags, followed by a form, table and JavaScript loop to output table rows containing the selected fields. The JavaScript close() method closes the frame for writing.

Key Technology: Passing Tag Attributes to a Base Tag Through Structures

When CF encounters the <CF_QBE> starting tag, it outputs a series of JavaScript validation routines used to parse search criteria entries. It also includes the Allaire supplied wddx.js, a library of JavaScript functions used to manipulate client-side WDDX recordsets.

For every <CF_QUERYCRITERIA> tag called, all the parameters passed to the sub-tag are aggregated in a CF structure, called thistag.assocattribs. The contents of query-criteria.cfm, depicted in Listing 6, are quite simple – a series of <CFPARAM> statements specifying default values, followed by the <CFASSOCIATE> tag, which makes these parameters visible to the parent tag.



Finally, the second call to the parent tag, </CF_QBE>, actually outputs the JavaScript required to write the field selector list, instantiates the JavaScript object that will hold the user search criteria and redraws the selected fields. Listing 7 demonstrates a CFLOOP whereby all of the attributes aggregated from the multiple nested subtags are rendered as an array of JavaScript objects during the </CFQBE> sequence.

Key Technology: Web-Distributed Data Exchange (WDDX)

WDDX is a generic format for storing and passing data between servers, or in this case between CF and the browser. The ColdFusion <CFWDDX> tag is used to both serialize data (convert CF to WDDX format or CF to JavaScript objects) and to deserialize data (convert WDDX format to CF or JavaScript objects to CF). Both simple (text strings) or complex (arrays, structures or queries) variables may be translated between the domains.

Notice that in Listing 8, the CFWDDX tag appears between the <SCRIPT></SCRIPT> tags. The ColdFusion server takes the empty CF server-side recordset and converts it to a series of JavaScript arrays that may be accessed through the variable name SEARCHCRITERIA. The columns of SEARCHCRITERIA can be referenced in Javascript as searchcriteria.fieldname, searchcriteria.logic, etc., and the rows as searchcriteria.fieldname[0], searchcriteria.fieldname[1], etc. The ACTION parameter of the CFWDDX tag is used to indicate how to translate the data object. CFML2JS specifies conversion from ColdFusion to JavaScript. Later, we'll see how <CFWDDX> translates end-user selections back into a CFQUERY recordset object.

Each time a search field is chosen from the list of available fields (in the left frame), the field appears in the right frame and a new row is added to the SEARCHCRITERIA object. This is done through a call to the addfield function, depicted in Listing 9. Note that the addRows method is defined in wddx.js and is similar to the ColdFusion QueryAddRow function. Also, be aware that while CFQUERY recordsets begin with a row offset of 1, their JavaScript counterparts begin with a row offset of zero, hence the -1 modifier.

As the user chooses which search fields to use in the query, the list of chosen fields in the right upper frame grows. Columns are set up so the user can choose which relational operator to use with the example data s/he enters. The HTML code that displays these columns, after the DEPARTMENT field is chosen from the search field list, is depicted in Listing 9. Note that it was dynamically generated from within the browser using the document.write() methodology.

The code in Listing 10 sets up the two columns for the first chosen field (department in this case). Since this is the first field (and first row in the SEARCHCRITERIA object), the form elements are named with a 0 following the descriptive text (logic and data); subsequent elements will be named logic1, data1, logic2, data2, etc. There are two event handlers associated with these elements. When the logic element selection is made, the onChange event fires and updates the current row (0) of the logic property of the SEARCHCRITERIA object in the button frame with the value (text) of the selection chosen. If the first option is chosen (REMOVE, value=0), the drawCriteria() function is also called to redraw the field list in the right frame. The second event fires when the element named data0 is visited. This event simply updates the current row (0) of the value property of the SEARCHCRITERIA object in the button frame.

When the Run Query button is pressed, the doRunQuery function is called. The following is a small portion of that function:

Backsoft

www.backsoft.com

```
wddxSerializer = new WddxSerializer();
wddxPacket =
ddxSerializer.serialize(searchcriteria);

document.forms[0].querypacket.value=wddx-
Packet;
document.forms[0].mybutton.value='Requery
';
document.forms[0].sub-
mit();
```

The first line above creates an instance of the WddxSerializer object (defined in wddx.js). The wddxPacket object is then given a value equal to the serialized version of the SEARCHCRITERIA object. The hidden field, named querypacket in the button frame, is then assigned the value of wddxPacket. At this point, the hidden field querypacket's value is (contents have been reformatted for easy viewing):

```
<wddxPacket version='0.9'>
<header/>
<data>
<recordset rowCount='1'
fieldName='fieldname,logic,value,datatype,
display,deletedyn'>
<field name='fieldname'><string>EmpDe-
partments.DepartmentID</string></field>
<field
name='logic'><string>=</string></field>
<field
name='value'><string>1</string></field>
<field
name='datatype'><string>numeric</string><
```

```
/field>
<field name='display'><string>Depart-
ment</string></field>
<field
name='deletedyn'><string>N</string></fiel-
d>
</recordset>
</data>
</wddxPacket>
```

The wddxPacket above simply uses a taglike notation to identify the type of data contained between each set of tags (recordset, field and string). The recordset consists of one row with the specified fields. Each field then has a name and value associated with it delimited by the tags describing the datatype of that field.

The next step is to execute the query. The form in the querystart.cfm template (in the button frame) is submitted and the action page (runquery.cfm) constructs the query by decoding the above wddxPacket.


The runquery.cfm template calls the qbedecode.cfm template which actually decodes the wddxPacket and constructs most of the where clause in the query.

```
<CFWDDX ACTION="wddx2cfml"
input="#attributes.querypacket#"
output="myquery">
```

The above tag is called from qbedecode and converts the contents of the querypacket variable back into the original query (with a number

of new rows). The newly converted query is called myquery. The rest of the qbedecode template loops through the myquery rows and constructs the where clause based on the fieldname, logic, value and datatype fields in the query.

Putting It All Together

Developing advanced database applications with ColdFusion is deceptively simple. Certainly, there's a lot that can be done using only CFML, however, the real payoff comes from mixing server-side technologies with client-side JavaScript and DHTML. In the coming months, we'll continue to investigate and develop highly reusable and scalable constructs using the techniques described here. In the meantime, have fun with the QBE of Death and (hopefully) never have to code another complex query system again. 

About the Authors

Steve Drucker is the CEO of Fig Leaf Software with offices in Washington, DC and Atlanta, Georgia. He founded the first ColdFusion users group (DC-CFUG), coauthored the ColdFusion Web Database Construction Kit (first and second editions) and is a certified Allaire instructor. He can be reached at sdrucker@figleaf.com.

Robert Segal, a certified Allaire instructor, works at Fig Leaf Software. He has been developing applications for the past 10 years and using CF for the past three. He can be reached at rsegal@figleaf.com.

sdrucker@figleaf.com rsegal@figleaf.com

LISTING 1: A very simple custom tag

```
<CFPARAM NAME="attributes.message" DEFAULT="I have nothing to
report">

<CFOUTPUT>
<SCRIPT LANGUAGE="JavaScript">
alert("#attributes.message#");
</SCRIPT>
</CFOUTPUT>
```

LISTING 2: Invoking the QBE of Death

```
<HTML>
<BODY>
<CF_QBE fieldlistframe="parent.dataframe.fieldframe"
criteriaframe="parent.dataframe.scriteria"
fieldcriteriaurl="framesetup.htm"
fieldcriteriaframe="parent.dataframe"
buttonframe="buttonframe"
actionframe="dataframe"
ACTION="runquery.cfm">

<CF QUERYCRITERIA DISPLAY="lastname"
FIELDNAME="EmpEmployees.lastname"
DATATYPE="string"
MAXLENGTH="10"
EXAMPLE="Jones,Smith"
DESCRIPTION="Search by Last Name">

<CF QUERYCRITERIA DISPLAY="firstname"
FIELDNAME="EmpEmployees.firstname"
DATATYPE="string"
MAXLENGTH="10"
EXAMPLE="Steve,Dave"
DESCRIPTION="Search by First Name">
```

```
</CF_QBE>
```

```
</BODY>
</HTML>
```

LISTING 3: Setting up top-level frameset

```
<!--index.htm - starts everything --$

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">

<html>
<head>
<title>QBE of Death, By Steve Drucker</title>
</head>

<FRAMESET rows="*,50" frameborder="No" framespacing="0" BOR-
DER=0>
<FRAME NAME="dataframe" SRC="framesetup.htm">
<FRAME NAME="buttonframe" SRC="querystart.cfm">
</FRAMESET>

</html>
```

LISTING 4: Setting up search selection frameset

```
<!--
framesetup.htm - our field selection and search criteria
frames
--$

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">

<html>
```


Processing Data Files

How to use ColdFusion to track information



BY
RAYMOND
THOMPSON

My company developed and now maintains an application in ColdFusion that tracks samples from collection to final payment. These samples are used to verify environmental quality and identify hazards on and around a major nuclear facility. If you have any knowledge of history, I'm sure that you would recognize the significance of the Oak Ridge Reservation.

A large part of this sample collection process is entering and maintaining information in a database that tracks all the sample information. This information was either collected with bar code readers taken from the sample containers themselves or text files were e-mailed to the user and printed out; then the user would manually enter all the data.

A better method needed to be developed to allow entry of the data. The current method was tedious and prone to errors. With the significance of the data collected, eliminating errors was important.

Application Specifics

The application that maintains the sample information has been entirely transitioned from a character-based system to a Web-based system. ColdFusion was the development tool of choice because of the flexibility and rapid development time.

The application database is running on Ingres that sits on a DEC (now Compaq) UNIX server. Windows NT provides the ColdFusion services. All database accessing is done through ODBC over the network through software provided by OpenLink. When the database is transitioned to Oracle, the access will be with native drivers provided with ColdFusion. The browser of choice is Netscape as a company standard.

Transferring the File

Since the file is sent as an e-mail attachment, the file will be stored on the user's machine when the e-mail is delivered. Where this file is stored on the user's machine was not important, the key point being that the file resides on the user's machine.

It was necessary to develop a form that would deliver the file from the

user's machine to the NT server so that ColdFusion would have access to the file. A form was built that included all the usual elements to make the form presentable.

The key element on the form was the HTML tag `<INPUT ...>` to specify that a file is to be transferred. The browser interprets this tag by placing an input box and "Browse" button on the screen. By clicking on the "Browse" button the user selects the file to be processed. This file is the data that was delivered by e-mail to the user.

The precise syntax of the tag used is shown in Listing 1. The type parameter specifies that this is a file tag and will be used to specify a file. The size and maxlength parameters are arbitrary and can be whatever looks good on the form. The accept parameter indicates the type of file that is to be transferred.

When the user submits the form, the contents of the file are delivered to the server in the variable specified in the name field of the `<INPUT...>` tag.

Getting the File

The next form in the application is the form that actually does the processing. Before the file can be transferred a destination file has to be determined. The file name needs to be unique so that multiple users can do these transfers without fear of conflicts in data files.

Prior to the CFFILE tag there is a CFSET variable that determines a unique name for the file. This CFSET gets the name of the temporary directory and appends the name of a temporary file that contains the user ID of the user. This user ID was stored early in the application as a cookie variable. The GETTEMPFILE function takes two parameters – the desired directory and the first few characters of the file name

– and returns the name of the file to be used to store the uploaded file.

The tag that actually does the processing is the CFFILE tag. This tag specifies the action that's required, in this case UPLOAD. The filefield is the name of the field that contains the name of the file. In this case it's the form variable that came from the first form. This name is shown in the "name" field in Listing 1. The destination of the file is also specified in the tag and needs to be located on a file resource that the ColdFusion server can access.

The CFFILE tag also provides an action when the file is a duplicate of one already on the destination resource. The example in Listing 2 informs ColdFusion to overwrite the file. Because ColdFusion was allowed to set the file name on the server, there should not be any duplicates.

After the file is transferred, some variables are used to store information about the file that was received. These aren't necessary to do the transfer; they're just used to provide some informative feedback to the user on a subsequent screen.

The last item needed to get the file is the CFFILE tag with an action of READ. This tag takes the file and stores the data into a ColdFusion variable.

All went well at this point and the file was transferred. But I was in for a surprise when the file arrived. Having cut my teeth many years ago on "Big Iron," I was used to record-level processing. In other words, files were a series of fields that were contained in individual records. This is not quite what was delivered by ColdFusion. So some work was going to be required.

There is no record-processing logic in ColdFusion and the data that's received must be processed as a stream of data. It became my responsi-

Alive.com

www.alive.com

Internet

www.inte

t World

ernet.com

bility to properly break up the data into units that were acceptable to my application.

File Specifics

The file is an ASCII text file with the data fields comma delimited, which is the format the scanners produced. The fields can be variable in length, presenting a problem, but since the fields were comma delimited at least they could be picked out, which was the key to solving the problem.

There is also the possibility of one of the fields in the file having multiple entries separated by a slash (/), requiring processing as if they were separate records sharing common data.

Preparing the Data

Since the fields in the file were delimited by commas, treating the file data stream as a list item seemed to be a good choice. Before this could happen, the data in the file needed to be cleaned up. And since the fields in the file were separated by commas, another method had to be devised to separate the records into list items.

The individual records are just ASCII files, so at the end of each record a carriage return and a linefeed are inserted. This provided the necessary delimiters to properly split the data into unit records.

Listing 3 contains constants that are used in the processing of the file. I like to use constants rather than hard-coded numbers. It makes code changes easier and helps the readability of the code.

Listing 4 contains the code that actually takes the data file and breaks it up into individual components. The CFLOOP breaks the file stream into record components. Since the data file

has carriage returns and linefeeds at the end of each record, the records are separated on these boundaries.

All occurrences of carriage returns are moved from the data stream. Then a delimiter character that shouldn't occur in the data stream replaces the linefeeds. The character chosen was the vertical bar (|). If the data should ever contain this vertical bar, the code will fail. But some character had to be chosen and this seemed logical.

I could have used the actual linefeed character as the delimiter between records. This was discounted because during the debugging process I needed to display the data and the linefeed character didn't display well.

Processing the Data

At this point the individual records have been separated in such a way that ColdFusion should be able to process the data. What is left for the loop to process is a list of individual records separated by the vertical bar; the individual fields in the record are separated by commas. This is pretty much what was needed.

Because of the possibility of one individual field having multiple components, representing what needed to become multiple records, another list loop was needed. This list loop looks at the items in the field that can be separated by a forward slash. If there is only one item, the loop will only execute once. For multiple items the list will loop the necessary times.

All that was left at this point was to extract individual fields from the records. These were treated as list items and the LISTGETAT function was used. These fields are then stored in arrays for further processing.

There are some checks in the processing that make sure the list items in

the record are long enough. If the list length is not long enough the array items are just replaced with default items.

Cleaning Up

The last thing that needed to be done was to clean up the data on the server by getting rid of the transferred file. Listing 5 indicates the code that does this cleanup. Again it's the CFFILE tag and the action is "delete". The name of the file that was stored in a prior variable indicates the file name. The file is then removed from the server to avoid clutter.

Summary

The processing of data files sometimes becomes necessary in an application. With some careful thought, ColdFusion can process most files.

The significance of files is that they're nothing more than a stream of data. ColdFusion makes no distinction about the data or type of data. It's up to the developer to process the file in units that are compatible with the type of data.

In this case the use of the List functions in ColdFusion allowed the processing to proceed fairly easily. All that was necessary was to do some preparation on the file before processing.

Using file processing tags that are available to the developer significantly eases the effort required to process files. With a little thought and a careful look at the data, ColdFusion is quite capable of processing external data.

This is one example of using ColdFusion to do some needed processing on a user's data file. The savings to the user were significant and well worth the effort.



ABOUT THE AUTHOR

Ray Thompson has 30 years' experience in the software industry as a manager and software developer. Currently he is working in Web application development for Q Systems in Oak Ridge, Tennessee.

RAYT@QSYSTEMS.

LISTING 1

```
<input type="File" name="SourceFile" align="LEFT" size="25"
maxlength="30" accept="application/msexcel">
```

LISTING 2

```
<cfset
BarCodeFile=GetTempFile(GetTempDirectory(),"#Cookie.UserID#")
>
<cffile action="UPLOAD" filefield="Form.SourceFile" destina-
tion="#BarCodeFile#" nameconflict="OVERWRITE">
<cfset BarCodeFileDirectory=File.ServerDirectory>
<cfset BarCodeFileName=FILE.ServerFile>
<cfset SourceFileDirectory=File.ClientDirectory>
<cfset SourceFileName=File.ClientFile>
<cffile action="read" file="#BarCodeFile#" variable="Sample-
Data">
```

LISTING 3

```
<cfset LF=Chr(10)>
<cfset ListDelim="|">
<cfset SampleDelim="/">
<cfset FieldCOCPos=1>
<cfset MatrixPos=2>
<cfset SampleIDPos=4>
<cfset MethodPos=5>
<cfset SampleDatePos=6>
<cfset UnknownMethod="** UNKNOWN **">
<cfset Duplicate="** DUPLICATE **">
<cfset DefaultProponent="RADIOLOGICAL">
```



```
<cfloop index="SampleLine"
list="#UCase(Trim(Replace(StripCR(Trim(SampleData)),LF,'','all')))" delimiters="#ListDelim#">
<cfif ListLen(SampleLine) EQ SampleDatePos>
<!---
```

```
<cfloop index="MethIdx"
list="#Trim(ListGetAt(SampleLine,MethodPos))#"
delimiters="#SampleDelim#">
<cfset Success=ArrayAppend(FieldCOCHold,Trim(ListGetAt(Sam-
pleLine,FieldCOCPos)))>
<cfset Success=ArrayAppend(MatrixHold,Trim(ListGetAt(Sample-
Line,MatrixPos)))>
<cfset Success=ArrayAppend(SampleIDHold,Trim(ListGetAt(Sam-
pleLine,SampleIDPos)))>
<cfset Success=ArrayAppend(MethodInputHold,MethIdx)>
<cfset Success=ArrayAppend(SampleDateHold,ListGetAt(Sample-
Line,SampleDatePos))>
<!-------
```

```
<cfset Success=ListFindNoCase(SampleBarMethod,MethIndx)>
<cfif Success GT 0>
```

```
<cfset Success=ArrayAppend(MethodConvertHold,ListGetAt(SampleFixMethod,Success,ListDelim))>
<cfelse>
<cfset Success=ArrayAppend(MethodConvertHold,UnknownMethod)>
</cfif>
</cfloop>
<cfelse>
<!-------
```

```
<cfset Success=ArrayAppend(FieldCOCHold,"")>
<cfset Success=ArrayAppend(MatrixHold,"")>
<cfset Success=ArrayAppend(SampleIDHold,"")>
<cfset Success=ArrayAppend(MethodInputHold,"")>
<cfset Success=ArrayAppend(MethodConvertHold,"")>
<cfset Success=ArrayAppend(SampleDateHold,"")>
</cfif>
</cfloop>
```

```
<cffile action="DELETE" file="#BarCodeFile#">
```

The code listing for
this article can also be located at
www.ColdFusionJournal.com

www.WeAreCFHosting.com

CLEAR AND SIMPLE

- ColdFusion 4.0, Active Server Pages
- MS SQL Server, Access, FoxPro
- Redundant DS3 Connection
- FrontPage 98 Support
- SSL Secure Server
- Shopping Cart
- ECommerce
- CyberCash



* Does not include set up fee and InterNIC domain registration fees.

Plans starting

\$14.95 / month*



Toll Free: (877) 684-
6784

BUILDING APPLICATIONS

PART 1

TAGGING
THE
SERVLET

by Ajit Sagar

Online stores are the new, next-generation, "revolutionize the world as we see it today" way of doing business. In the context of business transactions, online stores use the global Internet to facilitate the purchase and sale of goods and services. The ability to support online sales is an essential component of the new e-business paradigm for Internet-based businesses today. Putting together an enterprise-level application for an Internet store involves design and integration of various technologies that play specific roles in a distributed computing environment. A distributed topology is a prerequisite for building such Internet applications since the Internet is inherently distributed in nature.

Due to the plethora of technology alternatives available in the computing arena today, designing architectures for an enterprise application involves choosing between technologies based on feasibility, applicability, cost, availability and several other factors. No solution is "the right one." The hard part is to figure out which technology to use to provide a particular functionality. The challenge is to take competing and complementary technologies and make them play nicely together.

This is the first in a series of articles on some of the prominent technologies available to build a simple Internet-based Ticket Store application. This application offers online purchase of airline tickets as well as goods sold in airports. Our discussion focuses on two prominent technologies: ColdFusion and some components of the Java platform, name-

ly Java applets and servlets, RMI and JDBC. The modules implemented using CF will be presented here and in two subsequent issues of *ColdFusion Developer's Journal*. The modules implemented using Java will be discussed in Volume 4, issues 6, 7, and 9, of *Java Developer's Journal*.

One of the main objectives of this design is to illustrate how Java servlets can be used as access mechanisms in server modules that serve up data to application modules implemented in ColdFusion. Currently, a majority of services across the Internet are accessed through the use of CGI scripts. Servlets offer an attractive alternative to CGI, especially if you're using Java to provide the server-side functionality. This article will concentrate on why ColdFusion and Java servlets make a great combination for developing Web-based

applications and how Java servlets can be accessed from ColdFusion templates. We'll walk through the design of a Custom Tag called CF_Servlet that will allow invocation of Java servlets from ColdFusion templates.

As I'd like to keep this discussion focused on servlet access from ColdFusion, I'll refrain from going into detail on servlets themselves. There are several good books and articles on servlets and I've provided references to a couple of them at the end of the article. While you don't need to be an expert on servlets to benefit from the discussion here, a basic familiarity with CGI, Java, URLs and ColdFusion Custom Tags is assumed.

technologies playing nicely together
competing and complementary

WITH COLD FUSION & JAVA



- *To add dynamism to the Web applications by enabling client-to-server interactivity.*

ColdFusion's most powerful feature is its capability to connect to data sources that are maintained in other applications. It allows the building of dynamic queries on the fly to retrieve data from such applications. This makes it the ideal tool for creating dynamic Web application components such as shopping carts, account management modules, purchasing modules and customer profiles.

Servlets on the other hand excel at making server-side services available to the client in a dynamic and interactive fashion. They can be used to efficiently access a variety of services offered across different tiers of a distributed architecture. Basically, servlets serve HTML to the client. They also bring access control and enhanced security into the equation. They are closely tied to the Web server they run on and thus help extend the server's capabilities to the client. One thing servlets are *not* designed for is building sophisticated GUIs. They are primarily HTML servers.

The focus in Java technologies is shifting toward server-side Java, and that makes servlets a prominent player in developing Web applications. Typically, ColdFusion-based applications would use servlets:

1. To access server-side services offered by Java applications
2. To gain access to RMI/CORBA services
3. To perform sophisticated computation-intensive tasks on the server as opposed to burdening the client with this responsibility

Accessing Servlets

Clients can access servlets in the following ways:

- **Via a URL:** This is the same as connecting to any site via a browser. The query string parameters at the end of the URL are picked up by servlets in the same manner as CGI programs.
- **Via Server-Side Includes (SSI):** These are used to embed servlets within HTML pages. This is achieved through the use of `<SERVLET>` tags in .html files. Files con-

taining the `<SERVLET> ... </SERVLET>` tags are named with an .shtml extension. They can be accessed via any Web browser via a URL, similar to regular .html files.

The remainder of this article focuses primarily on creating the CF_Servlet Custom Tag. This tag allows the ColdFusion templates to access servlets using either of the mechanisms listed above. It also allows the information needed to invoke the desired servlet to be specified from an Access database. Figure 1 shows the database with the names of three example servlets that can be invoked via the CF_Servlet tag.

Before getting into the details of the tag, let's go over the pieces required to invoke the servlet on a Web server:

For a URL invocation we'll need:

- *The name of the servlet class*
- *The URL for the Web server that houses the servlet*
- *Any query parameters that the servlet expects in the query string*

For an SSI invocation we'll need:

- *The name of the .shtml file*
- *The URL for the Web server that houses the servlet*

I've written the following two sample servlets that can be used to test the CF_Servlet tag. You can download these (and the corresponding SSI [.shtml] files) from www.ColdFusionJournal.com:

```
- HelloColdFusionServlet, HelloColdFusionServlet.shtml
- TicketServlet, TicketServlet.shtml
```

These servlets are clearly simplistic and exist only to demonstrate the use of the CF_Servlet tag.

The servletURL.cfm template

Before developing the CF_Servlet tag, we'll walk through the development of a template for invoking the servlets from a URL string. This is available in the file servletURL.cfm and is shown in Listing 1. The first part of the code is the usual HTML tags and the specification of the title for the browser. After this, the first statement in the `<BODY>` of the document initializes the variable url. It is initialized to "http://" since all URLs will begin with the HTTP

Why Servlets?

So why would the developers of a full-fledged application server like ColdFusion be interested in using Java servlets? There is certainly an overlap in functionality. Some of the common reasons for opting to use Java servlets and ColdFusion templates are:

- *To access relational databases from the next tier of the distributed system.* ColdFusion achieves this through a rich library of SQL-based tags and ODBC. Servlets achieve this via the use of JDBC.
- *To extend the capabilities of the Web server that serves up the data.*

ID	ServletClass	ServerURL	SSIFile	QueryString
1	HelloColdFusionServlet	localhost	HelloColdFusionServlet.cfm	method=GET
2	HelloWorldServlet	localhost	HelloWorldServlet.shtml	method=GET
3	FingerServlet	localhost	FingerServlet.shtml	method=GET

FIGURE 1: CF servlets table

protocol (at least for our purposes). Basically, the purpose of this template is to build up this variable based on the input from the browser.

The first thing we check is whether a trace parameter has been passed in. If it has, it will print out the final URL for invoking the servlet. If it hasn't, only the output from the servlet will be printed. The trace parameter can be used for debugging.

Next, the variable database is initialized to "cfservlets." This is the name of the table illustrated in Figure 1. The idea is that if the servlet has an entry there, the data for invoking it doesn't need to be supplied with each invocation. If the servletdatabase parameter is supplied, the template will pick up the data from the database name supplied as the value for that parameter. The variable database is initialized with this value.

The servletclass is a required parameter. Hence it is specified in the <CFPARAM> tag. None of the <CFPARAM> tags have any default values in this template because this template is meant for the generic invocation of servlets. In the absence of a parameter, it's better for it to fail than to invoke a default servlet.

The next section is the CFQUERY tag that retrieves the fields for invoking the servlet. The key used is the servletclass, which is also the primary key in the database. Once the records are obtained from the table, the next step is to build the URL. The URL is built as follows:

```
<CFOUTPUT QUERY="servlets">
  <CFSET "url" =
    "#url#/#ServerURL#/#Servlet-
    Class#">
</cfoutput>
```

The URL is built up of four segments - the string "http://", the URL for the server, the directory for the servlet plug-ins on the server and the name of the servlet. The standard directory for servlets is the directory "servlet" under the Web server's root directory. A typical URL at this point would be:

```
http://myURL.com/servlet/MyServlet
```

If we were picking up all the data from the database, at this stage our URL would be complete. The program would skip the <CFELSE> section and go on to check whether any queryString parameter was supplied in the

URL that was used to invoke the servletURL template. If so, it's appended to the end of the URL (demarcated with a "?" character).

Next, the trace variable is checked to see whether the URL needs to be printed out. This is followed by a check for the method parameter, which is used to specify "GET" or "POST" for the servlet. The servletURL template doesn't support "POST" yet. If the method parameter isn't supplied, the default service() method of the servlet is called.

The actual invocation is made in the CFHTTP tag. If the "GET" method was specified, it is passed to the servlet and the doGet() method of the servlet is called. If not, no method is explicitly specified. This causes the servlet's service() method to be invoked (which in turn calls doGet()); hence the end result is the same).

The last part of the template is the only significant piece as far as the browser user is concerned. The file content is output using the CFHTTP.FileContent variable.

This template can be used to invoke servlets from the URL. Figure 2 illustrates two such invocations. The first one causes the servlet to be invoked using data from the database. The second supplies the data in the URL and also turns on the Trace flag to print out the URL being invoked. The URLs used in the screens are:

```
http://localhost/Test/servletURL.cfm?serv-
letdatabase=cfservlets&servletclass=Hel-
loColdFusionServlet
```

and

```
http://localhost/Test/servletURL.cfm?serv-
erurl=localhost&servletclass=HelloColdFu-
sionServlet&Trace=ON
```

The CF_Servlet Custom Tag

Listing 2 shows the CF_Servlet tag, which is available in the file servlet.cfm. The changes from the template to the Custom Tag are trivial. Basically, instead of reading the parameters from the URL, they are obtained from the ATTRIBUTES supplied by the calling template. The values of the corresponding ATTRIBUTE fields are copied into appropriate variables in the tag.

Listing 3 illustrates how the CF_Servlet tag is used. The file invokeHelloColdFusionServlet.cfm contains the code to invoke the

HelloColdFusionServlet in three different ways to demonstrate the CF_Servlet tag's flexibility. The first one uses the "database invocation." It also supplies three NAME=VALUE style parameters (although they're not used by the servlet). The second invocation supplies the server URL and other data in the parameters passed to the tag. The third invocation uses the SSI file to invoke the servlet. The output of all three invocations is displayed on the same page. This is illustrated in Figure 3.

Listing 4 shows an invocation of the TicketServlet via its SSI file. The TicketServlet simply sends back a hard-coded response for the flight query. However, it illustrates some key points in this design. The output of this invocation is displayed in two separate sections of the resultant page. The first half (Flight request Parameters) shows the parameters passed into the servlet; the second half, the output from the servlet that comes back to the ColdFusion Custom Tag.

Listing 4 represents the basis for the interaction between a ColdFusion application and the service exposed by the TicketServlet. In a real application this query could have gone across to another server, done some comparison shopping for airline tickets, and so on. The key thing is that the ColdFusion application is abstracted from all this. It basically submits a query and gets a price quote back.



FIGURE 2: Servlets invoked from template

So What's the Big Deal?

There's nothing that the CF_Servlet tag provides you that you couldn't have gotten by just typing the URL in the browser. However, what it does allow you to do is set up a basic communication between a ColdFusion template and a servlet residing in, perhaps, another tier of the application. The output from the servlet is going to be available in your template and can feed the front-end application.

The protocol for communication between ColdFusion and Java servlets as demonstrated in this article is simple and string based. But let's stop and think for a minute. The server side of this equation is in Java. One of Java's primary features is dynamic class loading. What if object names were supplied to the servlet by ColdFusion templates? These objects could be instantiated on the fly, used and then discarded. Well-known and published services (such as CORBA/RMI services) could be accessed via this simple means of communication.

Allaire

www.allaire.com


```
<!-- This custom tag is used for establishing a connection
with a servlet -->
```

```
<HTML>
<HEAD>
<TITLE>
    Java Servlet Invocation Custom Tag
</TITLE>
</HEAD>

<BODY>

<!-- Retrieve the input parameters -->

<!-- The specifications for invoking the servlet may be
specified either in the
    parameters specified in the URL that invokes the
CFServlet tag or may be
    picked up from a database that contains the servlet
specifications.
    The parameters specified in the URL override the data-
base specs.
-->

<CFSET "url" = "http://">

<!-- If the variable "servletdatabase" is defined, the
servlet specs can be
    pulled up from the database.
-->

<CFIF IsDefined("trace")>
    <CFOUTPUT>Trace is ON<BR></cfoutput>
</cfif>

<CFSET database="cfservelets">

<CFIF IsDefined("servletdatabase")>
```

```

<FSET database=#servletdatabase#>
<CFPARAM NAME ="servletclass">

<CFQUERY DATASOURCE=#database# NAME="Servlets">
    SELECT ServletClass,
           ServerURL,
           SSIFile,
           QueryString
    FROM Servlets
    WHERE ServletClass = '#servletclass#'
</cfquery>

<CFOUTPUT QUERY="servlets">
    <CFSET "url" = "#url##ServerURL#/servlet/#Servlet-
Class#">

</cfoutput>

<!-- If the database is not specified, the servlet specs
are supplied as
    input parameters. Pick up the URL first (it is now a
required parameter.
-->
<CFELSE>

    <!-- The Web Server URL is a required parameter -->
    <CFPARAM NAME="serverurl">
    <CFSET "url"="#url##serverurl#">

    <!-- The servlet may be invoked via a Server Side
Include (.shtml) file
-->
    <CFIF IsDefined("ssifile")>
        <CFSET "url"="#url##ssifile#">

    <CFELSE>
        <!-- The
servlet

```

BROWSER COMPATIBILITY MADE EASY!

<CF BROWSERHAWK>

BrowserHawk™ is the ultimate solution for the elusive task of accurately recognizing web browsers visiting your web site and their capabilities. Using this revolutionary tool, you can easily create dynamic web sites which properly support the wide variety of browsers in use today.

Without this accurate accounting for browser differences, web sites are prone to serious problems including script failures, inconsistent content presentation, reduced content availability, and even browser crashes. These problems can mean lost sales, increased costs for customer support and related web maintenance, and can affect a company's public image.

BrowserHawk™ frees you from all the hassles and complexities involved in detecting and accounting for browser differences. It allows you to easily produce sites with a consistent look and feel and level of operation for all visitors to your site, regardless of their browser used, with graceful degradation for older browsers.

Get Started today!

Get Your
FREE
30-day trial!

800-932-6869
sales@cyscape.com

www.cyscape.com/free



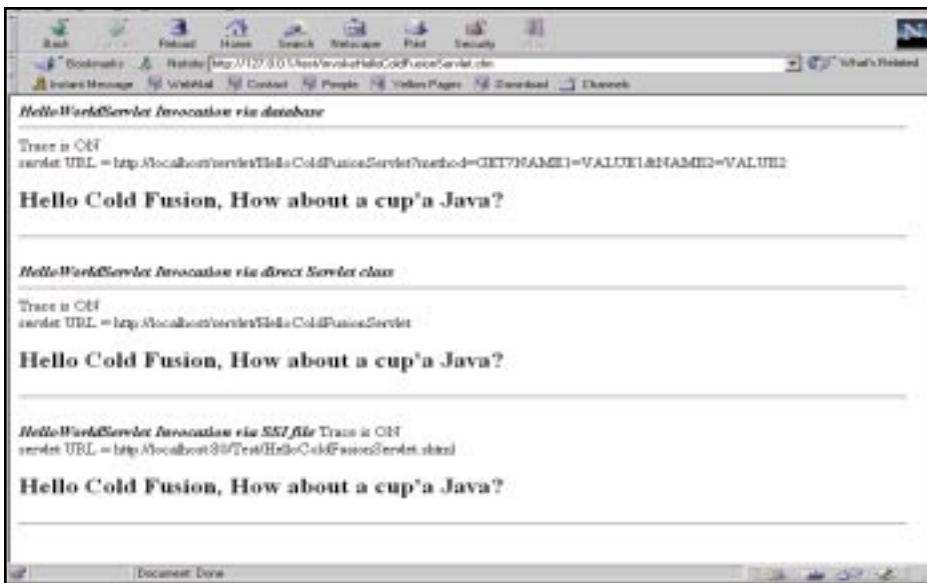


FIGURE 33 Output of three invocations

Development Environment

The code shown in this article was developed on NT 4.0. The servlets were tested in JRun Pro 2.2. The version of ColdFusion used was 4.0. The JDK version for the Java classes was 1.17B, and JSDK 2.0 (Java Servlet Development Kit) was used for the servlets.

The following files are available at www.ColdFusionJournal.com. The ".java" files will need to be installed in JRun in order to run the ColdFusion templates:

Listing 1: *ServletURL.cfm*

Listing 2: *Servlet.cfm*

Listing 3: *invokeHelloColdFusionServlet.cfm*

Listing 4: *invokeTicketServlet.cfm*

Listing 5: *HelloColdFusionServlet.java*


Listing 6: *HelloColdFusionServlet.shtml*

Listing 7: *TicketServlet.java*

Listing 8: *TicketServlet.shtml*

A Microsoft Access database file is also available for the reader to download: *cfervlets.mdb*.

The Online Airline Ticket Store

In subsequent articles in this series we'll use the foundation built here to develop an example e-business store. The Online Ticket Store is an Internet-based application that allows an Internet user to log in and purchase tickets via a browser. It also offers an online store to purchase items normally offered in duty-free shops in the regular international airports. In the next article we'll build some of the basic components of the store such as a shopping cart and a profile manager. We'll also add some intelligence to our queries for purchasing tickets and getting some meaningful quotes. 

References

Servlets: Hunter, J., and Crawford, W. (1999). *Java Servlet Programming*. O'Reilly and Associates.

ColdFusion Custom Tags: Forta, B., et al. (1999). *Advanced ColdFusion 4.0 Application Development*, pp. 114-132.

About the Author

Ajit Sagar, a member of the technical staff at i2 Technologies in Dallas, Texas, holds an MS in computer science and a BS in electrical engineering. He focuses on Web-based e-commerce applications and architectures. Ajit is a Sun-certified Java programmer with nine years of programming experience, including two and a half in Java. You can e-mail him at ajit_sagar@i2.com.

ajit_sagar@i2.com

Fusion FX

www.fusionfx.com

Safe and Sound Globally

Establish a security framework for your applications



BY
DAVID
SCHWARTZ

In my 13 years of software development I haven't seen a technology or platform as exciting as the Web. Now we can create truly global applications that can run from any browser anywhere in the world.

With this awesome power and reach come security issues. The first question clients ask is, "If we put our data on the Internet, will it be safe?" Thanks to the security features inherent in ColdFusion, we can answer with a confident "Yes!" Given the priority placed on security, when you sit down to write an application your first task should be to establish a security framework.

I'll show you how to create a generic security system that will be reusable, upgradable and scalable as your needs evolve. It will work with every ODBC-compliant database as well as with direct drivers like Oracle, and it's easy to do and maintain.

We'll be creating one database table and four ColdFusion templates for this project:

1. **User Table** – will store a list of users and their access permissions
2. **Application.Cfm** – sets the stage for your ColdFusion application
3. **Logon.Htm** – an HTML form for

users to type their user name and password

4. **Logon.Cfm** – a ColdFusion template to check the user name and password
5. **Security_Check.Cfm** – a ColdFusion template to check users' access rights whenever they try to do something that requires permission

Step 1: Build a Foundation

Create a database table (see Table 1) called Users to hold, for example, user names, passwords and access permissions. When a user logs on to your site, he or she will type a user name and password that will be checked against this user table. The Users table will also store their access permissions, such as Edit and Add. For this article we'll use an Access database; however, you can create the table in Access, SQL Server or Oracle. The logon will function the same.

After you create the Users table, add a record, fill in all the fields and set

the Admin field = True. Now you can test the logon and add other users.

On the ColdFusion server you'll need to create a System DSN (data source) so ColdFusion can access the tables.

Step 2: Set the Stage

ColdFusion uses a special file, Application.Cfm, to enable you to set various options and create application-wide variables, which are important since they remain active for the entire user's session and are accessible from all of your ColdFusion templates. Every time your program requests a CFM file, ColdFusion will automatically run the Application.Cfm file. Since Application.Cfm is always run, it's a convenient place to put code that always needs to run (see Listing 1).

Now that we've created the Users table and the Application.Cfm file containing your session variables, we'll build a logon form in which the user can enter his or her user name and password.

Step 3: Who Are You?

Create an HTML form called Logon.htm, which will need two input fields and a submit button (see Listing 2). This will be used to enter the user name and password.

Notice how the `<FORM ACTION="Logon.cfm">` tag calls the logon.cfm template.

Step 4: Is That You?

Create Logon.cfm, which will be used to check the user name and password in the Users file, as in Listing 3.

Logon.cfm queries the Users file to find a user with a matching user name and password. If a user is found, the security rights are set. If the user name and password aren't found, the user will get a message. I use session vari-

Users Table

First_Name	c15	
Last_Name	c20	
User name	c10	
Password	c10	
Edit	Logical	Edit records (Insert or Update)
Add	Logical	Add records to files
Delete	Logical	Delete records
Admin	Logical	Add and change user permissions

TABLE 1: Users database table

ables to store the settings since they're stored in memory and are fast. This sets the stage for the entire user connection. Now, whenever a user tries to access a page or feature in your application, you simply check the session variable for permission.

Step 5: Just Checking

Checking the user's access rights is now a simple matter. Security_Check.Cfm will check the values of the session variable for the access rights you're requesting:

```
Security_Check.Cfm
<!-- Simply check if the users
rights are true. -->
<CFPARAM name="Attributes.Rights">

<cfif #Attributes.Rights# neq
"True">
You do not have permission to use
this feature.<br>
Contact your system administrator
to upgrade your access level.
<cfabort>
</cfif>
```

All you have to do is pass the rights to it. At the top of each CFM that you want to check the access rights to, add one line:

```
<Cf_Security_Check Rights="#Session.Add#">
```

This will automatically call the script. If the user doesn't have access rights, a message will be presented and the user won't be permitted to go any further.

Step 6: Testing 1...2...3...

A sample ColdFusion form that checks if the user has Admin rights before being allowed to view the user database is given in Listing 4.

Another benefit of using session variables is that they timeout. This means if a user leaves his or her computer for a coffee break, ColdFusion will automatically reset the session variables. Another user trying to get in won't be able to. You can set the timeout length in the Application.cfm. I use 20 minutes.

Step 7: I Want More...

The security framework presented here offers a great start for building secure Web applications with ColdFusion and it's easy to expand on the functionality. For example, you could add fields to the user file for different access permissions, like accessing word processing files. Then you only need to add a session variable to the Application.Cfm and logon.Cfm files.

Whenever you create a ColdFusion template and want to check the users' access levels, just add one line of code to the top:

```
<Cf_Security_Check Rights="#Session._____"#">
```

In the next article I'll show you how to extend the security functionality and use it with SSL and database authentication.



ABOUT THE AUTHOR
David Schwartz is president of Array Software Inc., a software company based in New York City. He has been developing turnkey custom database software for 13 years.

DS@ARRAYONE.COM

RSW

www.rswsoftware.com

LISTING 1

```
Application.Cfm
<!-- Define the application wide parameters and variables-->
<cfapplication name="MyApp"
    clientmanagement="Yes"
    sessionmanagement="Yes"
    setclientcookies="Yes"
    sessiontimeout="#CreateTimeSpan(0,0,20,0)"#
    applicationtimeout="#CreateTimeSpan(1,0,20,0)"#
    clientstorage="REGISTRY">

<!-- Initialize application variables. Here we create the
session variables to logon users and track them. -->
<cflock timeout="30" throwontimeout="Yes" name="SessionLock">
<cfparam name="Session.UserName" default="">
<cfparam name="Session.UserFirstName" default="">
<cfparam name="Session.UserLastName" default="">
<cfparam name="Session.View" default="">
<cfparam name="Session.Delete" default="">
<cfparam name="Session.Add" default="">
<cfparam name="Session.Admin" default="">
</cflock>
```

LISTING 2

```

Logon.Htm
<HTML>
<HEAD>
<TITLE>User Logon</TITLE>
</HEAD>
<BODY>
<!-- the following Form Action tells the browser to call
the Logon.Cfm template when the user presses Submit -->
<form action="/Article/Logon.cfm" method="POST" post="YES"
multipart="YES">
<CENTER>
<TABLE BORDER="0">
<TR>
<TD><B>Logon</B></TD>
</TR>
<TR>
<TD>User name</TD>
<TD><INPUT TYPE="TEXT" NAME="Username" SIZE="10"
MAXLENGTH="10">
</TD>
</TR>
<TR>
<TD>Password</TD>
<TD><!-- Note: the following input field is set to "password".
This will automatically show a "*" when the user types their
password. -->
<TD><INPUT TYPE="PASSWORD" NAME="Password" SIZE="10"
MAXLENGTH="10">
</TD>
</TR>
<TR>
<TD>&nbsp;</TD>
<TD><INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Logon">
</TD>
</TR>
</TABLE>
</CENTER>
</FORM>
</BODY>
</HTML>

```

LISTING 3

```

Login.Cfm
<!-- Query the user database. Check if the user name and
password are found. -->
<!-- Please note, if your Access database does not have a
password you don't have to pass the username and password in
the query (For SQL Server and Oracle you must). -->
<cfquery name="Login"
datasource="MyDataSource"

```

```

        maxrows=1
        dbtype="ODBC"
        username="david"
        password="david">
SELECT * FROM Users WHERE Username = '#Form.UserName#' And
Password = '#Form.Password#'
</cfquery>

```

```
<!-- If the user name and password were found then the user
is valid and the query will have at least one record. If the
record count is not equal to 1 then the logon is invalid.
-->
<cfif #Logon.RecordCount# NEQ 1>
Your user name or password didn't match. Press the [Back]
button to retry.
<cfabort>
</cfif>
```

```
<!-- If the logon is valid we set the session variables and
let the user into the program. -->
<cflock timeout="30" throwtimeout="No" name="SetSession">
  <cfset Session.UserName = #Logon.UserName#>
  <cfset Session.LastName = #Logon.Last_Name#>
  <cfset Session.SessionFirstName = #Logon.First_Name#>
  <cfset Session.View = #Logon.Search#>
  <cfset Session.Delete = #Logon.Delete_Files#>
  <cfset Session.Add = #Logon.Add_Files#>
  <cfset Session.Admin = #Logon.Admin#>
</cflock>
```

```
<!-- Since the login was successful direct the user to your
home page. -->
<cflocation url="https://www.arrayone.com/Article/View.Cfm"
addtoken="Yes"></cflocation>
```

LISTING 4

```
View.Cfm
<!-- The first thing to do is check if the user has "Admin"
permission. If they don't, processing will be stopped in the
Security Check.Cfm. -->
```

```
<cf Security Check Rights="#Session.Admin#">
<HTML><HEAD><TITLE>Users</TITLE></HEAD>
<FORM ACTION="default.cgi" METHOD="POST" ENCTYPE="application/x-www-form-urlencoded">
<B><FONT SIZE="5" COLOR="#CC9900" FACE="Arial Narrow">View
Users</FONT></B><br>
<cfquery name="Users" datasource="MyDataSource" dbtype="ODBC"
username="david" password="david">
SELECT * FROM Users Order by Last_Name
</cfquery>
<table border="0">
<tr bgcolor="#ECB723">
<td><b></b></td>
<td>First Name</td>
<td>Last Name</td>
<td>Add?</td>
<td>Delete?</td>
<td>Admin?</td>
</tr>
<cfoutput query = "Users">
<tr>
<td>#Users.CurrentRow#</TD>
<td>#Users.First Name#</td>
<td>#Users.Last_Name#</td>
<td>#YesNoFormat(Users.Add_Files)#</td>
<td>#YesNoFormat(Users.Delete_Files)#</td>
<td>#YesNoFormat(Users.Admin)#</td>
</tr>
</cfoutput>
</table>
</FORM>
</BODY>
</HTML>
```

CODE
LISTING

The code listing for
this article can also be located at
www.ColdFusionJournal.com

Allaire

www.allaire.com

Extending ColdFusion with Java Servlets and JRun

The servlet engine for developing and deploying Java servlets

BY
EDWIN
SMITH

The ColdFusion Markup Language is the easiest way to generate dynamic Web content from a database. Its tag-based scripting commands are simple for HTML authors to learn, yet powerful enough for building full-featured Web applications. Custom ColdFusion tags can be developed in either CFML or C++ using the CFX plug-in API.

Java servlets are becoming the most effective way to build server-side Web plug-ins. Because servlets are written in Java, they can run just about anywhere and connect to nearly anything. They're faster than CGI scripts and easier to write than native platform plug-ins such as Microsoft's Internet Server API (ISAPI), Netscape's Server API (NSAPI) and CFX. You can write entire applications with servlets or use them as "web glue" to "webify" anything from a printer to a mainframe.

JRun is the award-winning, industry-leading servlet engine for developing and deploying Java servlets. It's an easy-to-use Web server plug-in that allows you to deploy Java servlets and JavaServer Pages (JSP). JRun was developed by Live Software, which was recently acquired by Allaire Corporation. For more information on JRun and to download a free copy visit www.livesoftware.com.

With a few simple examples this article will demonstrate how easy it is to invoke Java servlets within your CFML pages using JRun.

CF_SERVLET

Anyone familiar with the `<SERVLET>` tag will immediately understand `<CF_SERVLET>`. It invokes the named servlet running on JRun, and the results are included in the output of the page. Common HTTP headers and tag attributes are available to the servlet using the `ServletRequest.getParameter()` method.

For example, let's say we want to invoke JRun's `EmailLiteServlet` (included in the JRun Servlet Pack One) to send an e-mail after a form submission. Using NCSA-style Server Side Includes

(SSI), the invocation would look as follows:

```
<SERVLET CODE=EmailLiteServlet>
<PARAM NAME="host" VALUE="pop.some-
host.com">
<PARAM NAME="dest"
VALUE="http://www.myhost.com/suc-
cesspage.shtml">
<PARAM NAME="from"
VALUE="me@myhost.com">
<PARAM NAME="to" VALUE="you@some-
host.com">
<PARAM NAME="subject"
VALUE="Servlets are cool">
<PARAM NAME="body" VALUE="Because
they can run from ColdFusion">
</SERVLET>
```



The same servlet can be invoked much more simply from ColdFusion as follows:

```
<CF_SERVLET CODE=EmailLiteServlet
host="pop.somehost.com"
dest="http://www.myhost.com/suc-
cesspage.cfm"
from="me@myhost.com"
to="you@somehost.com"
subject="Servlets are cool"
body="Because they can run from
ColdFusion">
```

By default, `CF_SERVLET` connects to a JRun running on the same machine as ColdFusion. Since JRun's proxy protocol is TCP/IP based, however, you can invoke servlets anywhere you have a JRun, even through a firewall. The following code shows using the `JRUNPROXY` attribute to access a JRun running on the MyServer.com host:

```
<CF_SERVLET CODE=MyCoolServlet
JRUNPROXY=MyServer.com:8081
Cool="you bet">
```

Because you're still running inside a ColdFusion page, you can even dynamically choose where to connect to JRun. For example, the user could enter it in a form field:

```
<CF_SERVLET CODE=RenderNewDo
JRUNPROXY=#Form.JRUNPROXY#
Cool=#Form.HAIRSTYLE#>
```

The advantages of `CF_SERVLET` stem from CFML being an easy-to-use, tag-based scripting language and servlets being easy-to-write, powerful plug-ins.

Allaire Developer's Conference

www.allaire.com

How It Works: CFX_JRUN

CF_SERVLET is a CFML wrapper around CFX_JRUN, a C++-based JRun connector. JRun's out-of-process, connector-based architecture makes it simple to plug into any server environment (Web server or application server). CFX_JRUN hooks JRun into ColdFusion, translating between the CFX API and JRun's proxy protocol. Like JRun's other native connectors, CFX_JRUN has a very small footprint. Once CFX_JRUN is configured into ColdFusion, CF_SERVLET can be dropped into any CFML page.

What About <CF_Anywhere>?

<CF_Anywhere> is a servlet-based CFML engine included in JRun. It translates a CFML subset into Java servlets on the fly, in a manner similar to JSP. Although <CF_Anywhere> also uses the CF_SERVLET tag to invoke servlets, the implementation is completely different from that of ColdFusion. Your custom servlets can thus be invoked equally well from a CFML page, whether that page is executing on ColdFusion or <CF_Anywhere>. The same thing can't be said for other custom CFX plug-ins.

ABOUT THE AUTHOR

Edwin Smith is a principal software engineer at Allaire Corporation and works closely with all aspects of JRun engineering, particularly Java/native integration, system architecture and performance optimization.

“”


ColdFusion,
CF_SERVLET
and JRun
together make
a great
combination:
the power of
CFML scripting
plus the power
of Java servlets

Pricing Information

JRun Pro (Win, UNIX, Novell, Mac versions): \$595 per processor

JRun Pro Unlimited (Win, UNIX, Novell, Mac versions): \$1,995 per machine

Conclusion

ColdFusion, CF_SERVLET and JRun together make a great combination: the power of CFML scripting plus the power of Java servlets. If you're a CFML developer considering writing a ColdFusion custom tag, you now have a choice other than CFML, C++ or CFX_J (Java-based version of the C++ CFX API). With CF_SERVLET you can write a Java servlet that can be used by CFML developers or with any Web server or application server. If you're a third-party Java servlet developer writing Web plug-ins for your application, a single servlet could replace your custom ISAPI, NSAPI and CFX plug-ins and broaden your customer base dramatically. If you're a Java servlet developer who wants to tap into the huge market for CF custom tags, CF_SERVLET and JRun are your solution. 

EDSMITH@ALLAIRE.COM

Infoboard

ColdFusion Developer's Journal

Look What's Coming!

Attention Advertisers:

Don't Miss your chance to advertise in our next

Allaire Developer Conference

Special Issue! Call Robyn Forma

1-914-735-0300

or email robyn@sys-con.com



ADVERTISING INDEX

ADVERTISER	URL	PH	PG
ABLE SOLUTIONS	WWW.ABLECOMMERCE.COM	360.253.4142	2
ALIVE.COM	WWW.ALIVE.COM	206.674.7700	25
ALLAIRE	WWW.ALLAIRE.COM	888.939.2545	33,39,41,43
BACKSOFT	WWW.BACKSOFT.COM	888.222.6047	21
BIZNIZ WEB	WWW.WEBPUBLISHINGTOOLS.COM	281.367.4016	45
CATOUZER	WWW.CATOUZER.COM	604.662.7557	51
CYSCAPE	WWW.CYSCAPE.COM	800.932.6869	34
DATARETURN	WWW.DATARETURN.COM	604.662.7551	7
FINDADEVELOPER.COM	WWW.FINDADEVELOPER.COM	440.918.0394	45
FINDAHOST.COM	WWW.FINDAHOST.COM	440.257.6690	17
FUSION FX	WWW.FUSIONFX.COM	800.780.2422	35
GALILEO DEVELOPMENT SYSTEMS	WWW.GALILEODEV.COM	770.643.9176	13
HOSTPRO	WWW.HOSTPRO.NET	888.638.5831	15
INFORBOARD	WWW.INFOBOARD.COM	800.514.2297	42,45
INTERLAND	WWW.INTERLAND.NET	800.323.4532	4
INTERMEDIA	WWW.INTERMEDIA.NET	650.424.9935	52
INTERNETWORLD	WWW.INTERNET.COM	800.500.1959	9
MACROMEDIA	WWW.MACROMEDIA.COM	800.457.1774	3,9
RSW SOFTWARE	WWW.RSWSOFTWARE.COM	508.435.8000	37
SITEHOSTING.NET	WWW.SITEHOSTING.NET	888.463.6168	29
VIRTUALSCAPE	WWW.VIRTUALSCAPE.COM	212.460.8406	47

Allaire Experts

www.allaire.com



Another Satisfied Advertiser...

'Catouzer's flagship product Synergy, a CF based application framework for corporate Intranets, is targeted at CF developers and system integrators.

As the leading ColdFusion publication, ColdFusion Developer's Journal was the logical place for the focus of our advertising campaign.

Placing an advertisement in the first two issues of CFDJ has been instrumental in increasing traffic to our web site and downloads by over 150%, which has resulted in a substantial increase in revenue.'

Gerry Haag
Director of Sales & Marketing
Catouzer, Inc.

About the Author

Jeremy Allaire is a cofounder and vice president of technology strategy at Allaire. He helps determine the company's future product direction and is responsible for establishing key strategic partnerships within the Internet industry. Jeremy has been a regular author and analyst on Internet technologies for the past seven years, and he holds degrees in both political science and philosophy from Macalester College.

jeremy@allaire.com

www.ColdFusionJournal.com



SYNERGY 1.5
Catouzer Inc. suite 501-1228 hamilton st. vancouver, b.c. v6b 2s8 canada tel: 604 681-1111

www.catouzer.com

email: synergy@catouzer.com

Able Solutions

Enter the realm of browsable store building and administration – from your browser. Build "your_site.com" with secure Merchant Credit Card Processing. Maintain inventory, add discounts and specials to keep your customers coming back. Increase sales with cross selling and membership pricing.

11700 NE 95th Street, Suite 100, Vancouver, WA
www.ablecommerce.com • 360 253-4142

Alive.com

You don't have to be an HTML programmer to create media-rich e-shows for your Web site. Use Alive's templates and prompts to quickly create your own slides, add audio, video and graphics, then synchronize it all as easily as clicking your mouse button. Need to put it on the intranet? Or the Web site? Yes! Alive could well be the most time and cost effective communication tool you'll ever own.

83 S. King St., Ste 414, Seattle, WA 98104
www.alive.com • 888 386-9969

Backsoft Corp.

BackTalk for SAP Web application server is the fastest way to build and deliver scalable applications that integrate browser, server and SAP database technologies. BackTalk for SAP is the first web-enabled solution for the SAP R/3 system which uses Allaire's ColdFusion Technology. It leverages the scalability of Allaire's ColdFusion Architecture and builds an intuitive development tool to Web-enable SAP.

5971 Cattle Ridge Blvd, Suite 101, Sarasota, FL 34232
www.backsoft.com • 941 378-2325

BiznizWEB
Internet Publishing
Big Bucks? Heavy Web Expertise?
Operations Nightmare?
It doesn't have to be that way!
www.webpublishingtools.com
local portals city guides calendars classifieds

DYNAMIC
WEB
PUBLISHING
recruitment directories

Catouzer, Inc.

With the Synergy 1.0 Web application framework, creating custom intranet applications is a breeze. The Synergy Application Development Kit (ADK) gives you the tools to rapidly develop your custom applications, which can be fully integrated and managed under the Application Services Layer (ASL).

1228 Hamilton Street, Suite 501, Vancouver, B.C. V6B 2S8, Canada
www.catouzer.com • 604 662-7551

Data Return Corporation

Data Return offers extensive support for customers utilizing ColdFusion from Allaire. With customers delivering over 50,000 user sessions per day, we know how to provide high availability solutions for this advanced application server. Our technical support staff has extensive experience in coding custom ColdFusion Tags as well as managing Microsoft SQL server. We also support CyberCash for customers interested in real-time credit card processing. Our combination of support and experience offer an ideal environment for deploying your applications developed for ColdFusion.

801 Stadium Dr., Ste 117, Arlington, TX 76011
www.datareturn.com • 800 767-1514

Fall Internet World

Whether you're a webmaster evaluating E-commerce software, an ISP rethinking your business strategy, a marketing executive looking for proven online opportunities or an IT professional who needs to secure an enterprise network, we've got something for you. Totally immerse yourself in intensive multi-day programs or spend just a couple of hours exploring a specific topic of interest.

Jacob K. Javits Convention Center, NYC

www.internet.com • Conference: October 4-8, 1999

FINDaDEVELOPER.com
• Post & Search Job Listings
• List Your Web Development Services For Free
www.findadeveloper.com

Fusion FX, Inc.

Fusion FX, Inc., is a full-service Web site design and hosting company. Our strength in ColdFusion Development, as well as with custom AbleCommerce Store building, and the fact that we have a Team Allaire member on staff will give you peace of mind because you know your ColdFusion development is in good hands.

819 Peacock Plaza, Suite 535, Key West, FL 33040
www.fusionfx.com • 800 780-2422

infoboard
NT and UNIX
Cold Fusion Hosting
Development Consulting
Oracle, Informix, MS-SQL, E-Commerce Plugins
1-800-514-2297
sales@infoboard.com
www.infoboard.com

Galileo Development Systems

The Intr@Vision family of products provides the basis for automating your business processes over the web using a standard, extensible framework. With a focus on reducing the cost of doing business and improving cash flow, Intr@Vision Time and Intr@Vision Expense offer a web-based alternative to your current paper processes. Contact us today for more information on our products.

2695 Long Lake Dr., Roswell GA 30075
www.galileodev.com • 770 643-9176

HostPro

HostPro offers packages to suit just about anyone's growing bandwidth, storage and feature requirements. Our web site hosting solutions are packaged to meet specific needs based on popular criteria. This includes UNIX and NT packages, e-commerce solutions that range from simple to turn-key, databases like mSQL to mission-critical MS SQL, multimedia like RealAudio, RealVideo and all the programs and languages to extend your site's capabilities

3250 Wilshire Blvd., Suite 707, Los Angeles, CA 90010
www.hostpro.net • 213 252-9779

Intermedia, Inc.

Our advanced virtual hosting packages (powered by Microsoft Windows NT and Internet Information Server 4.0) offer an environment supporting everything today's advanced Web developer or sophisticated client could ask for. Complete ODBC support is available on plans B and C. We support Microsoft Index Server on all hosting plans.

953 Industrial Avenue, Suite 121, Palo Alto, CA 94303
www.intermedia.net • 650 424-9935

Macromedia Corp.

Macromedia is the leading developer of cross-platform software tools for digital media creation and publishing. Macromedia's products are used by organizations to create and deliver interactive applications that use a full range of media, from text and graphics, to animation. The product line includes Director, Authorware, ShockWave, FreeHand, and Dreamweaver.

600 Townsend Street, San Francisco, CA 94103
www.macromedia.com • 415 252-2000

RSW Software

RSW Software is a wholly owned subsidiary of Teradyne, Inc., and specializes in Web application testing software. Established with the goal of providing best-in-class testing products, RSW offers a suite of products called the e-TEST Suite, which automates the process of testing business-critical Internet and intranet applications.

44 Spring Street, Second Floor, Watertown, MA 02172
www.rswsoftware.com • 508 435-8000

Sitehosting.NET

Successful electronic commerce starts at SiteHosting.net; a division of Dynatek Infoworld, Inc., which provides total Web development services. We offer personal and efficient customer service with reliability at value prices. All our plans include access to SSL (Secure Socket Layer). We support ColdFusion, Active Server Pages, Real Audio/Video, Net-show Server, and more. Our hosting price starts at \$14.95/month.

13200 Crossroads Parkway North, Suite 360, City of Industry, CA 91746
www.sitehosting.net • 877 684-6784

Virtualscape

Why host with Virtualscape? Nobody else on the Internet understands what it takes to host ColdFusion like we do. Virtualscape is the leader in advanced Web site hosting. From Fortune 500 extranets to e-commerce sites and more, developers recognize our speed, stability, reliability and technical support.

215 Park Avenue South, Suite 1905, New York, NY 10003
www.virtualscape.com • 212 460-8406

e-TEST Suite from RSW Software

A Web tool for testing e-business apps from development to deployment

BY
TOM
TAULLI



There's nothing more annoying (and scary) than putting your credit card information on a Web site and getting a server error. This happened to me when I tried to buy an airline ticket. I entered my information three times...and was charged for three tickets.

In the wild west of the Web, it's crucial to have airtight Web sites. So how can you test for such things? One way, of course, is to purchase an automated tool. One definitely worth considering is e-TEST Suite from RSW Software.

The first tool is e-Tester, a scripting tool that puts your Web site through the wringer - you record (and play back) certain keystrokes that a typical user would enter at your site.

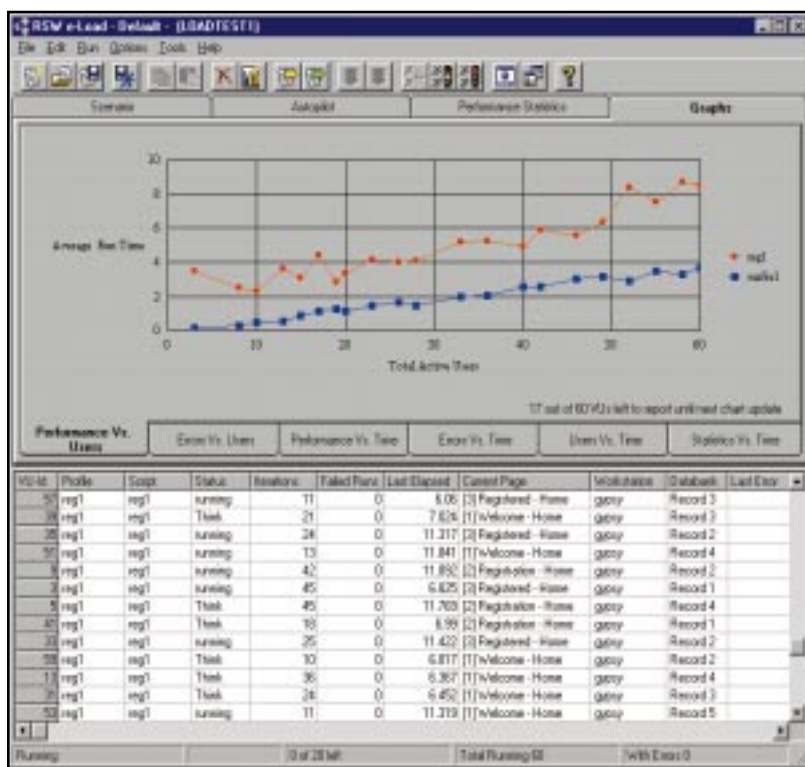
The software is built for Windows NT 4.0 and Win95 and 98. You'll need, at a minimum, 16 MB of RAM and 20 MB of hard drive space. Although e-Tester is built around MS Internet Explorer, you can set up the software to use Netscape Navigator.

e-Tester is intuitive, and no programming is required. You'll spend most of your time at the main window, which is divided into various panes. In the middle of the screen you have the Web browser pane; the results pane is at the bottom of the screen and at the left is a list of the scripts.

The documentation has some helpful tutorials that will get you up to speed quickly.

Creating a script is simple. Click the record button, enter the keystrokes and press the stop record button. When you play back the script, e-Tester runs a resource validation test and then shows the results. You'll see checks for the integrity of the referenced resources - links, images and so on. This will establish the baseline.

Let's say you make some changes to the site and then test the site again. The changes are compared against the baseline. Errors are displayed using color-coded flags in a tree form. Double-click a node on the tree and you'll



E-LOAD: Produces real-time graphs and reports to show you results while the test is in progress.

hyperlink to the corresponding page. If you make changes to the page, this will establish a new baseline.

If your site is large, you can use the e-Spider feature, which as the name implies will automatically map out your site and place it into a script, thus establishing a baseline.

With your scripts you can perform a variety of useful tests such as analyzing server response times, text matching (making sure you're putting the right text into the page), inserting a form element (confirming that the correct types of values are going into the form fields) and inserting an external callout. Basi-

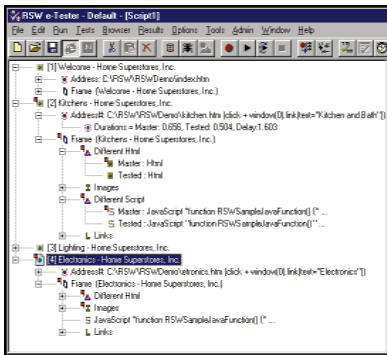
cally, you can create your own customized test using an OLE .dll file. To do this, you can use popular programming languages such as Visual Basic and Microsoft Visual C++.

There's also the RSW Data Bank Wizard. The wizard allows you to feed huge amounts of data into your site (e.g., account numbers, names, passwords). e-Tester will then take this information and see whether your site is working properly. You can easily modify the Data Bank information with a text editor, word processor or spreadsheet.

Part of the e-TEST Suite is a product called e-Monitor. Once your Web site

ABOUT THE PRODUCT
e-TEST Suite (e-Tester, e-Load, e-Monitor)
Requires: Microsoft Windows NT 4.0 or Microsoft Windows 95,98
Minimum Recommended Memory: 64MB
Minimum Disk Space: 30MB
System: IBM compatible PC, Pentium Pro or faster processor recommended
Price: \$27,000

ABOUT THE COMPANY
RSW Software Inc.
44 Spring Street,
2nd Floor
Watertown, MA 02472
Phone: (508) 435-8000
Fax: (617) 924-2238
www.rswsoftware.com
info@rswsoftware.com



E-TESTER: Visual Scripts capture and test all the objects on each page and indicate errors with color-coded flags.

goes live, you want it to perform optimally. e-Monitor provides 24/7 monitoring of your site, alerting you to problems in real time.

The error-reporting mechanisms include log failures, e-mail alerts, restarting of executables and so on. The error messages are descriptive, indicating time or error, script name and page, and error number and explanation. You can even integrate all this with CA UniCenter and Tivoli.




E-MONITOR: Automatically generates scripts for continuous monitoring, periodic daily testing or timed batch regression testing.

Also part of the e-TEST Suite is e-Load; it includes a Scenario Manager that allows you to simulate hundreds or thousands of users accessing your site. With a click of the mouse, you can bring up a test script and specify the number of virtual users. You can vary the number of users over time to gauge scalability, which is becoming crucial. With the Scenario Manager you can also do data-driven testing. In other words, you can create scenarios of dif-

ferent users accessing different database records simultaneously, which is done by using the Data Bank information. What is really cool, though, is that you can get real-time reports that graph the performance of the site. So far, e-Load integrates with such Web development environments as BroadVision, NetDynamics, WebObjects, ColdFusion and MS ASP.

To see for yourself, download an evaluation copy from the RSW Web site at www.rswsoftware.com.

A variety of major companies currently use e-TEST. For example, Countrywide, a mortgage company, plans to use e-TEST to develop over a thousand Web applications during the next two years. According to Countrywide, they have improved performance in their application server development two to three times over.

The complete suite is expensive at \$27,000. But it's a small price to pay for reliability of your Web site. 

ABOUT THE AUTHOR

Tom Taulli is the CEO of Blueprint Interactive (www.bpia.com), which develops Internet applications for the enterprise.

TTAULLI@BPIA.COM

Virtualscape

www.virtualscape.com

**Take a look
at our specials
this month!**

EASTLAND DATA SYTEMS Internet Shopping with Java Shopping Cart

...Described as the most progressive and interactive form of shopping on the web today... This Java Applet provides a complete user interface package for Internet Shopping Web Sites. Using Java technology we produce a drag-and-drop shopping user interface that is fun and easy to use, encouraging shoppers instead of frustrating them with confusing controls that are hard to follow. And the easier it is to shop, the more you sell.



\$294⁹⁹

Hybrid Shopping Cart

This Java Applet provides a complete user interface package for Internet Shopping Web Sites. A "Hybrid" is defined as an offspring of two varieties. A blending of the best features from our CGI and Java shopping products, we took the most powerful aspects of Java technology: real-time, on-screen updating and computational capabilities. And combined those with the most desirable features of our CGI shopping Cart, namely it's flexibility and compatibility with web designers with artistic talent.



\$294⁹⁹

CGI Shopping Cart

The Shopping Cart automates the Shopping Process to make shopping on your site intuitive, straight forward, and enjoyable! It's one of the most affordable Shopping Carts because it was designed for small businesses. Specifically for entrepreneurs who are testing the Internet waters, and can't or don't want to make large investments into bells and whistles for their site. But simply want to make shopping on their site easy for the customer.



\$294⁹⁹

Guaranteed Best Prices

JDJ Store Guarantees the Best Prices. If you see any of our products listed anywhere at a lower price, we'll match that price and still bring you the same quality service.

Terms of offer:

- Offer good through August 30, 1999
- Only applicable to pricing on current versions of software
- August issue prices only
- Offer does not apply towards errors in competitors' printed prices
- Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

Attention Java Vendors:
To include your product in JDJStore.com,
please contact jackie@sys-con.com



**GUARANTEED
BEST PRICES
FOR ALL YOUR
JAVA RELATED
SOFTWARE
NEEDS**

ALLAIRE

ColdFusion 4.0

With ColdFusion 4.0, create Web applications for self-service HR solutions, online stores, interactive publishing and much more. The integrated development environment that has all the visual tools you need to create Web applications quickly and easily. From simple to sophisticated ColdFusion gives you the power to deliver the Web solutions you need-faster, and at a lower cost.



ColdFusion Studio 4.0 \$354⁹⁹
SkillBuilding with ColdFusion Interactive Training CD \$284⁹⁹

ALLAIRE

JRun - Live Software

JRun is the industry-leading tool for deploying server-side Java. JRun is an easy-to-use web server 'plugin' that allows you to deploy Java Servlets and JavaServer Pages. Servlets form the foundation for sophisticated server-side application development. Java servlets are platform independent, easy to develop, fast to deploy, and cost-effective to maintain.



JRun \$558⁹⁹

PROTOVIEW

Java Enterprise Editions

The Java Enterprise Editions offer you a choice between comprehensive packages of award-winning AWT or JFC components along with Enterprise Level Support and subscription service. Powerful, extendable, lightweight components built on the foundation of JFC, the ProtoView JFCSuite contains JFCDataCalendar, JFCDataExplorer and JFCDataInput. The JSuite (AWT) includes the DataTableJ grid component with JDBC and Visual Café database support. Also includes TreeViewJ, CalendarJ, TabJ and WinJ.

JSuite Enterprise Edition \$818⁹⁹
JFC Enterprise Edition \$818⁹⁵
JSuite \$328⁹⁹
JFCSuite \$408⁹⁹

GALILEO DEVELOPMENT SYSTEMS

Intr@Vision Foundation

Intr@Vision Foundation helps bring ColdFusion development to the next level. It provides an out-of-the-box application security architecture for handling your most complex intranet and extranet needs. Instead of spending 30% of your development time adding security to every application you build, it gives you a proven solution with a single line of code. Intr@Vision Foundation allows your developers to focus on building business solutions, not infrastructure.



Intr@Vision Foundation \$3499⁹⁹

ALLAIRE

HomeSite 4.0

HomeSite is the award-winning HTML editing tool that lets you build great Web sites in less time, while maintaining Pure HTML. Unlike WYSIWYG authoring tools, HomeSite gives you precise layout control, total design flexibility and full access to the latest Web technologies, such as DHTML, SMIL, Cascading Style Sheets and JavaScript. HomeSite 4.0 is the only HTML editor featuring a visual development environment that preserves code integrity.



HomeSite 4.0 \$87⁹⁹

INSTALLSHIELD

InstallShield Java Edition 2.5

InstallShield Java Edition 2.5 is the powerful tool developers require to produce bulletproof InstallShield installations with Java versatility. You can target your application for multiple systems with cross-platform distribution. And InstallShield Java Edition 2.5 offers the key features and functionality designed to let developers go further in distribution and deployment.



InstallShield Java Edition \$474⁹⁹

KL GROUP

JProbe Suite

JProbe Profiler is the most powerful tool available for finding and eliminating performance bottlenecks in your Java code. JProbe Coverage makes it easy to locate individual lines of untested codes and reports exactly how much of your Java code has been tested. JProbe Threadalyzer lets you pinpoint the cause of stalls and deadlocks in your Java applications and makes it easy to predict race conditions that can corrupt application data.



JProbe Profiler w/ Standard Support (inc. JProbe Memory Debugger) . . . \$464⁹⁹
JProbe Coverage w/ Standard Support \$464⁹⁹
JProbe Threadalyzer w/ Standard Support \$464⁹⁹
JProbe Suite w/ Standard Support \$934⁹⁹

CATOUZER

Synergy SOHO

Synergy is a ColdFusion based web application framework for instantly deployable corporate intranets. It consists of an Application Services Layer, which offers central security and administrative services, and eight core collaborative applications. Synergy's open architecture is designed for implementing existing ColdFusion applications and developing new ColdFusion applications specifically tailored to the customer's needs.



Synergy SOHO \$499⁹⁹

ORDER ONLINE

WWW.JDJSTORE.COM

Information is only
>a click away!<

Twin Cities ColdFusion Usergroup

The recent months have seen a tremendous growth of the ColdFusion Usergroup in Allaire's original homeland. Since December, when we came under new leadership, the group has gone from quarterly meetings to monthly.

We have a variety of skill levels. There is a strong constituency of entry-level developers, several mid-level developers and a few advanced developers.

Our meetings usually start with a presentation on a topic of interest to the group. This might be specific to a piece of complementary software; some aspect of Web serving, database design and optimization; or integrating the latest cutting-edge Internet technology into a ColdFusion site.

This is generally followed by a sharing time when we get to worship each other's accomplishments. People get to share the exciting things they've done over the past month that relate to CF development. This often centers on custom tags, complete sites and design ideas.

Finally, things tend to break down and the table is opened for a discussion about whatever is on the members' minds. This is a sort of brainstorming session with a certain level of hype and dare ("Sure, Joel, why don't you just write a tag to do that?") and a forum for open discussion.

These topics of discussion range from application design styles to security implications and implementations. Recently Jason Baker, the resident voice on site and server security and fault-tolerant systems, gave a presentation on using the Bright Tiger load-balancing software that comes with ColdFusion 4.0 Enterprise.

Our March meeting marked the last for James Moberg, the force behind youthpastor.com and the developer of a number of e-mail-related tags. It also featured our highest-profile guest speaker, Ben Forta. Ben gave a presentation on optimizing ColdFusion applications, highlighting many pitfalls that developers fall into, and introducing some additional development techniques that were new to most of the members.

One of our members, Joel Mueller, has recently joined Team Allaire. Joel has authored a number of tags in the Tag Gallery and has written chapters on COM and WDDX for the forthcoming *Mastering ColdFusion 4.0* book from Sybex. Michael Imhoff, a strong presence on the cf-talk mailing list, has given a presentation on Fusebox and also developed a powerful customizable pop-up help application add-on. I coordinate the CFUG and have developed the BlackBox style of ColdFusion development. I also developed the FusionCart Web shopping system, run JesusFreak.Com and have written for both *CFDJ* and Allaire.

We love to grow and share our ideas and development techniques. Beginners and all-star developers alike are welcome. We have tons of giveaways and are always looking for more reasons why new developers should come and existing members should return. Visit our Web site at <http://colderfusion.com/> for information on meeting times and locations as well as contact information. **CFUG**

by Dan Chick



Ben Forta (top, left center) meets with members of Twin Cities CFUG. Other members are shown in bottom photo. To get a *CFDJ* spotlight on your group, submit info to Chad Sittler, editor-in-chief, at chad@sys-con.com.

CFUG Spotlight

ColdFusion Developer's Journal - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://www.sys-con.com/coldfusion/index22.cfm

COLD FUSION Developer's Journal

CLICK HERE TO SUBSCRIBE TO CFDJ

CFDJ Digital Edition
The April Digital Edition of ColdFusion Developer's Journal is now available to CFDJ subscribers!

CFDJ Source Code
The source code for the ColdFusion Developer's Journal April 1999 issue is now available!

<BE> on <CF>
Catching in on Performance
There's nothing that can kill your applications performance as quickly as database access. This is a shame, considering that...

CFDJ INFO
New Business Models on the Web
By Jeremy Allaire
Growth in Internet commerce continues to fuel an adoption frenzy of Web technologies. The benefits of...

CFDJ Feature
Who Says You Can't Take It With You?
By Bob Hove
Have you ever been off on a trip, away from your home or office computer, and wanted (in, needed) to check your e-mail? Or have you ever been stuck...

ColdFusion Industry News
(March 20, 1999) - Allaire Announces Strategic Partnership with Compaq
(March 29, 1999) - ColdFusion announces their latest release, Synergy 1.5

CFDJ Special Offer
RCN Software offers a FREE copy of *Test Suite* to CFDJ readers.
Download a FREE demo of *Customer's Synergy 1.5*

CFDJ Editorial
Welcome to CFDJ!
By Chad Sittler, Editor-in-Chief
If you've picked up the premier issue of ColdFusion Developer's Journal expecting to read about ways to create an infinite energy source...

Reader Feedback
Ultimate Issue!
The first issue of CFDJ is a real success! This is what I've been looking for a lot of time. I love developing ColdFusion. Currently, I'm developing with CF3 and 4.0, and it's amazing what Allaire is doing. I just want to say congratulations on CFDJ, and by the way, I will get my subscription RIGHT NOW!!
Steve Love (steve123@icq.net)
Internet Developer
League Node Hacker Power Page

Wishing You Success!
There's no doubt that this site and magazine will be extremely successful. It is because of your quick response that I will get on the development and subscribe as soon as possible. Free sites have been willing to provide promotional responses and I understand that probably is extremely difficult to do so. The ones that know, though, seem to be the leaders in the new Internet movement. Amazon, Yahoo, EBay, & eBay, etc.
Thanks and wishing you the greatest of success.
Charles D. Bouda (dbouda@earthlink.net)
Web Developer
Bar Mill, L.P.

You're Off to a Great Start
After I had put your magazine I was overwhelmed. It came in a flash. However, I took the magazine on to my web site for a week. It's a great reference is there - not just a lot of fluff.
It has to be said something that I don't see elsewhere.
The source code is very interesting to me. I've had a few problems with it. For example, the code doesn't seem to be very easy to use. This is also true of the code editors. Programmers do not use left hand side editors. What's wrong with it? Come to the code editor? You're off to a great start. (In looking back to the editors, keep getting better as you go.)
Tom Tasso (tasso@jira.com)

CFDJ Product Review
AllCommerce Developer 2.0
By Tom Tasso
This is possible to do just about anything with ColdFusion. But if there's already a solution on the market that's well tested, why reinvent the wheel?

ColdFusion and CGI
Transitioning to ColdFusion from CGI Programming
By Ben Forta
There's little disputing the pace at which the Web has evolved over the past several years. Fueled by graphical interfaces to a suite of early Web-based...

Copyright © 1999 Sys-Con Publications, Inc.
All Rights Reserved.
Email: info@sys-con.com
ColdFusion and ColdFusion Developer's Journal are trademarks or registered trademarks of the Allaire Corporation. Sys-Con Publications, Inc., is independent of Allaire.

Recent: Ben

Information When You Want It, and when you need it

The Source. Online.
ColdFusion Developer's Journal.

www.ColdFusionJournal.com

Allaire ColdFusion, Java and JRun

BY
JEREMY
ALLAIRE



On June 15 Allaire announced that it intended to acquire Live Software, makers of JRun, the leading server-side Java development and deployment server. With this acquisition Allaire also announced a broader strategy for embracing Java on the server, extending its leading Web application platform with a huge customer base and technology platform, and setting the stage for an integrated application server platform that combines the dominant tag-based rapid development model (CFML) with the dominant server-side object-oriented system programming language, Java. These perfect cousins will form a critical foundation for the Allaire platform.

For Live Software and JRun, this means that their product and technology will be propelled through the Allaire channel and customer base. As a small, independent company, Live Software had achieved incredible success (not unlike the early days of Allaire), having built a base of about 80,000 developers who use the product, becoming the dominant Java Servlet/JSP vendor. With Allaire, JRun can gain the infrastructure needed to be adopted as a standard for server-side Java in enterprises worldwide. Additionally, the JRun team, headed by founder Paul Colton, will help Allaire as we evolve our platform overall.

The Application Server Market: A Short History

Many of you may remember the days of ColdFusion 1.0, which Allaire delivered in the summer of 1995. One of the first commercial Web application development products, ColdFusion quickly achieved dramatic success because of its simplicity, which was achieved through the invention of CFML (then called DBML), which provided a tag-based server-scripting model that tightly integrated with HTML pages. ColdFusion defined page-based dynamic Web applications.

By the fall of 1995 several other companies had emerged with competing products, such as Spider and DBWeb, which took very different approaches to Web applications. Essentially, they provided developers with a simple user interface to define form and report pages. Code was generated, and that was your application. While at face value it was very easy, this model severely limited the flexibility of what could be built.

In 1996 these companies took very different directions and approaches to the market, unlike Allaire, which continued to focus on its core platform, ColdFusion. In early 1996 the folks who made DBWeb, Aspect Software, went into beta on a product that was their "ColdFusion Killer."

At the same time, Spider technologies embarked on the Java path. Java had emerged in 1995 – and was heralded as the next generation of the Web, where applets and Java clients would replace the Web. Spider's approach was different, however. They focused instead on building server-side Web applications using Java. Late in 1996 their company, renamed NetDynamics, pioneered the use of Java on the server, launching the Java application server marketplace.

Within a year well over a dozen companies moved to enter this marketplace, each vendor taking a completely different approach to how your Java code connected to the Web.

In mid-1997 it was becoming increasingly clear to the world that Java in the browser was not going to fly. Recognizing this, Sun tried to push Java as a standard model for developing Web applications. Pushing the concept of servlets, Sun released the Servlet API and provided a standard way to connect Java code to a Web server. The importance of this was that there was the potential to standardize how server-side Java applications were built and deployed.

At the very same time, Paul Colton saw the power of Java on the server and developed and released JRun, the first commercial server for running servlets on any Web server and any platform.

By this time the Web application server marketplace was heating up. Multiple Java application server vendors were jockeying to have the "best" model for implementing Java on the server. Simultaneously, Allaire was plugging away at ColdFusion, defining rapid and productivity-focused Web application development and deployment and expanding its focus to cover more of the Web application server market requirements.

Also at this time Paul Colton and the JRun crew were plugging along, and realized that servlets alone weren't enough to make Java on the server a smash hit. Paul looked at the success of ColdFusion and ASP, and saw that a successful model needed to embrace pages and templates as a way to separate presentation and application logic. Seeing that Java on the server needed an equivalent model, he implemented the technology that would later be standardized by IBM and Sun as JavaServer Pages.

Consolidation continued in the application server market in early 1998, with Sun buying NetDynamics and IBM moving to release WebSphere Application Server. In the background, however, was Live Software, which essentially was giving away its product and building a massive community of developers building with Java

Servlets and JSP. In fact, the day that IBM and Sun announced the intention to create a specification for JSP, Live Software announced it was shipping a commercial product that supported it!

Like Allaire and ColdFusion, Live Software successfully used the Web to market and deliver a powerful and usable platform to a broad base of developers. By listening to customers and relentlessly pushing the technology forward, Live Software has achieved dominant market share for Java server technology.

With the Allaire acquisition of Live Software, Allaire is now the purveyor of two of the leading models for delivering Web applications. Not only are we the leaders, but we're the pioneers in both arenas.

ColdFusion and Java: Perfect Cousins

Developers, naturally, will ask and want to understand how ColdFusion and Java will come together to support an overall platform. In acquiring Live Software, Allaire believed it could not only propel itself into a leadership position in the Java server space, but that we could also bring together two of the dominant language models that have driven the Web platform in recent years.

Indeed, it's our belief that ColdFusion and Java are perfect cousins.

Stepping back and looking at broader trends in application platforms, it's clear that successful platforms need to do two primary things. First, they need to provide the core set of services that all applications require. In the Web platform landscape this includes things like session management, security, clustering, connectivity and messaging integration. Second, a platform needs to be able to support a broad range of developers and project profiles built on that platform. In the world of the Windows platform, we saw that Windows provided a common platform for applications, accessible to a broad range of developers using everything from Access, Visual Basic and PowerBuilder to C and C++. None is "better" than the other, but rather exposes capabilities based on class of developer and project.

Likewise, we believe that in the Web application platform marketplace there is a range of developers spanning business users, HTML developers, RAD developers using CFML and JavaScript, and system programmers building in Java. In fact, all of these need to work together in a common fashion, and integrate into core services provided by the application server platform.

Interestingly, both Allaire and Live Software were heading in this direction. With CFX_J, Allaire demonstrated its first effort to bring Java and ColdFusion together more directly. Live Software, simultaneously, had developed something called Taglets, which allows developers to build custom tags using servlets, running on JRun. From this they derived <CF Anywhere>, a light version of CFML built on Java. With the latest release of JRun we've included something called <CF Servlet>, which allows you to extend ColdFusion using servlets and JRun. As you can imagine, there will be a lot more of this coming out of Allaire in the future!

The important thing to realize here is that the marriage of ColdFusion and Java is a huge win for the Allaire Web application platform and for our broad developer community (now numbering over 300,000!), and that the acquisition of Live Software by Allaire will help ensure that Allaire continues to be the pioneer in Web application platforms.

ABOUT THE AUTHOR

Jeremy Allaire is a cofounder and vice president of technology strategy at Allaire. He helps determine the company's future product direction and is responsible for establishing key strategic partnerships within the Internet industry. Jeremy has been a regular author and analyst on Internet technologies for the past seven years, and he holds degrees in both political science and philosophy from Macalester College.

Catouzer

www.catouzer.com

Intermedia.NET

web hosting



TAKE
control
of your
Web Site

SPECIAL OFFER:
FREE
SET UP
WHEN YOU SPECIFY
PROMO CODE **CFD-J**

NT Hosting with IIS™ 4.0

Front Page™ 98

Cold Fusion™ 4.0

MS SQL Server™ 7.0

Cybercash™

PaymentNet™

WebTrends™

Online Site Management System
allows you to manage:

ODBC DSN Registration

ActiveX DLL & OCX Registration

Custom Cold Fusion Tag Registration

Dedicated Mail Server

(as well as many other features) INSTANTLY!

A must see Reseller Program!



order
ONLINE
today

www.intermedia.net

and have your account activated in less than 10 minutes



INTERMEDIA, INC.
953 Industrial Ave
Palo Alto Ca 94303
sales@intermedia.net