The Future of CF and Flash Pg.42

U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

June 2002 Volume: 4 Issue: 6

SYS-CON MEDIA

by Jeremy Allaire page 6

Macromedia MX & Web Services

ColdFusion Developer's Journal

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

editorial
editor-in-chief
Robert Diamond robert@sys-con.com
editorial director
Jeremy Geelan jeremy@sys-con.com
executive editor
M'lou Pinkham mpinkham@sys-con.com
managing editor
Cheryl Van Sise cheryl@sys-con.com
editor
Nancy Valentine nancy@sys-con.com
associate editors
Jamie Matusow jamie@sys-con.com
Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com
assistant editor
Jennifer Stilley jen@sys-con.com
editorial intern
Stephanie Williams stephanie@sys-con.com

production
VP, Production & Design
Jim Morgan jim@sys-con.com
Lead Designer
Cathryn Burak cathyb@sys-con.com
Art Director
Alex Botero alex@sys-con.com
Associate Art Directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Aarathi Venkataraman aarathi@sys-con.com
Assistant Art Director
Tami Beatty tami@sys-con.com

contributors to this issue
Jeremy Allaire, Charles Arehart, Dennis Baldwin, Ben Forta, David Gassner, Hal Helms, Simon Horwith, Kevin Towes, Bruce Van Horrn

editorial offices
SYS-CON MEDIA
135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9600
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
is published monthly (12 times a year)
by **SYS-CON Publications, Inc.**
postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645

Notes from the Field…

BY **ROBERT DIAMOND**

I'm writing this editorial on the way home from CFNorth in Toronto. It was, I can honestly say, one of the best times I've had at a conference, ColdFusion or otherwise. This was the case for a variety of reasons, not the least of which was the amazing effort that Kevin Towes and his team put forth in creating the most CF-friendly environment possible, with, simply put, some of the best speakers and the biggest names in the industry.

When I was first asked to give a keynote at the show, two thoughts immediately crossed my mind. The first was, "What the heck can I say for an hour that anyone wants to hear?" Quick calls to a few friends gave me the answer: not much.

The second was that they were anticipating 300 attendees, and I haven't spoken in front of that many people since my Bar Mitzvah some 10 years ago. An idea popped into my head to lead a panel discussion instead, a ***CFDJ*** panel. The team here at the magazine and I as the editor are in the unique position of being able to watch and to cover the whole industry. It's an always interesting task, and I thought the best option was to bring that to the conference as a collective effort rather than just rambling on by myself for an hour.

The end result was that we did the first-ever *CFDJ* Authors Panel keynote at the show, which I think went quite well. It's a shame we didn't tape it for posterity – we'll have to do that next time. Taking part on the panel were **Charlie Arehart**, **Ben Forta**, **Hel Helms**, **Michael Smith**, and yours truly (with the easier job of throwing questions that you submitted to the panel). We covered a wide range of topics, from Windows versus Linux (the consensus was yes, one of the two will work for you) to the new features in CFMX and the overall direction the industry is heading toward.

One of the centerpieces of the conference was the overall celebration of the MX announcements. ColdFusion MX is by far the most anticipated new release in ColdFusion history. It's been rewritten, and reengineered, from the ground up.

This advancement is good for anyone developing in ColdFusion, from beginner to advanced, and from those creating the smallest intranet to the largest dot-com site. ColdFusion is sometimes stereotyped as a weak language, a language for beginners. I think it's been unfairly pegged as such because of how easy it is to get up and running fast writing your first CF apps. *Easy* is often equated with being bad in the IT world. The truth is that it's very easy to write simple ColdFusion applications, and that's one reason ColdFusion has been so successful as a product, CFML as a language, and the whole package as an application server.

But saying it's easy to get started doesn't give us or the language the credit we deserve. Writing good applications is just as important a task on the ColdFusion server as it is on any other development platform. And CF programs can be just as complex as anything else available today. MX extends that even further, and the new release brings to the masses several formerly "higher-level" features, which you can take advantage of as much or as little as you choose.

Those developing jointly in Java and ColdFusion will be in heaven with the application server now running atop the J2EE platform. What makes ColdFusion so great for developers of all experience levels and sizes is that it's up to you: if you want to, you can ignore all the new stuff and write CF as you always have – and it'll run better than ever. Or you can leap deeper into the world of Internet application development and be, I think, pleasantly surprised by the powerful sites you can create. ***ColdFusion Developer's Journal*** is here to help you with whichever path you choose. So read on!

ABOUT THE AUTHOR
Robert Diamond is editor-in-chief of **ColdFusion Developer's Journal** *as well as* **Wireless Business & Technology**. *Named one of the "Top thirty magazine industry executives under the age of 30" in Folio magazine's November 2000 issue, Robert holds a BS degree in information management and technology from the School of Information Studies at Syracuse University.*

@ ROBERT@SYS-CON.COM

By Jeremy Allaire

Macromedia MX and Web Services

The MX family of software was introduced by Macromedia last month. Without a doubt, it is the broadest and most impactful set of software releases in the company's 10-year history.

Macromedia MX combines clients, servers, and tools into an integrated family that will deliver rich Internet applications. It deeply embraces the idea of deploying and using "software as services," though the MX approach goes beyond many of the current notions and discussions in the industry about Web services.

What's the Big Idea?

Over the past couple of years, an idea has emerged (some might argue it's an old idea) that software will be transformed into being used as services, rather than as monolithic applications tied to a specific machine or platform. Rather than install software onto computers every time we need some functionality, an end user or corporation can reuse other application assets over the network. The idea expands into the notion of just-in-time delivery of applications. "Software as services" is a big idea. It holds the promise of introducing radical new economies of scale into the manufacture and distribution of software applications. From an end-user perspective, it's the idea that users can access information and applications anytime, anywhere, and from any device.

This is a powerful set of ideas. It's the idea that the value we create in software can be freed from any single delivery platform. It's the idea that rich applications can be used easily across platforms and devices. How close are we today to realizing this set of ideals?

The Web Services Industry Today

The phenomenal focus on Web services over the past year is interesting when contrasted with the actual set of technologies available to fulfill this vision. When comparing the broader vision for software as services to the current world of Web services technologies – protocols like SOAP and formats like WSDL – it looks more like we're perhaps 25% of the way toward fulfilling that vision.

Today Web services are primarily talked about through the lens of specific protocols – SOAP and WSDL, perhaps UDDI, though I've yet to find a customer who's really using UDDI in any significant way. This early focus and discussion is great; finding the holy grail of a common protocol for exchanging objects and data between platforms is an incredible achievement, and the fact that the industry is so focused on it gives us hope for the future of interoperable and open software.

But even the current crop of technologies implemented around these standards has problems and challenges. In the Web services model, application logic is well structured, often as component services. In actuality, the need to deliver well-structured applications has largely eluded Web application development over the past years. In fact, we all know that close to 75% of Web applications aren't well structured; they're conglomerations of dynamic pages, including scripting languages, database code, and presentation logic all together. Indeed, the desire to move to component middleware standards such as EJB, CORBA, and COM has been manifest mostly in the world of high-end architectures within large corporations rather than in the mainstream of Web application development.

As such, there is a huge disconnect between the need for well-structured Web services and the reality that most Web applications are built using 4GL scripting languages, usually are unstructured, and rarely meet the requirements of thoughtfully designed object-oriented systems.

Let's assume for a moment that Web services are strictly defined as the middleware standards for messaging and object marshaling – for example, SOAP and WSDL. Even in that world, the vast majority of Web application developers aren't well positioned to both create and consume Web services without becoming full-on system programmers using complete object-oriented environments such as Java or .NET. The back-end world of Web services needs to evolve to make it simpler for RAD or scripting-level developers to easily create and consume Web services. It would be a huge victory for the industry if those 75% of Web applications built with scripting languages could also share their value and data with other applications.

However, the focus on Web services through this lens largely relegates the topic to discussions of classic back-end middleware rather than a more holistic view of what's necessary to deliver software as a service. Strikingly absent from the current discussion of Web services is any notion at all of what the end-user experience is of these "software services." This is due in part to the type of vendors thinking about Web services – for example, traditional enterprise middleware companies – but it's also because of the hard work required by interoperable messaging and object protocols. These are necessary conditions to any future for software as services.

Getting back to the original vision of software as services, it's quickly apparent that we need a much more holistic discussion and framework for thinking about Web services, one that actually includes end users using this software. What would an end-to-end model for Web services look like?

The Missing Piece: Rich Clients and Web Services

Many of the early visions and discussions of Web services centered on the idea that, in the future, software could be used as a service rather than as monolithic applications that needed to be installed and used on our desktop computers. While the Web has made progress in delivering applications easily to browsers, the world of Web services would take it further by delivering the experience of desktop-quality software that can be consumed as a service. Visions of using "productivity applications" as services were common interpretations by the industry. Examples such as Hotmail and Salesforce.com were cited as leading indicators. In this future world of software services, end users could easily access rich applications from any desktop computer; these applications would always have their personal information, would be interconnected with back-end systems, and could even be portable across devices. This new world, however, would leave behind the document-based or page-based model of the Web for one that was much richer in terms of application capability, and that extended beyond the browser onto desktops and devices.

What *is* the user experience of Web services?

Making software as services a reality

> **Rich clients and Web services combined hold the promise of fulfilling the broader vision the industry has for deploying software as services"**

What's the model for combining rich interfaces with back-end middleware to deliver exceptional new value for end users and the companies that serve them?

Macromedia believes that the perfect complement to Web services as middleware is the emerging category of rich clients. Indeed, it may well be that rich clients and Web services are two sides of the same coin, combining to enable this world of software as services.

What are these rich clients, and what do they require to really meet the requirements of transforming the user experience and providing the logical front-end to back-end Web services?

Rich clients should combine rich content, applications, and communications in a single client environment. By *rich content* I mean richly formatted text, graphics, audio, and video. By *applications* I mean both rich, complex-user interfaces – the kind we expect from a modern desktop computer – and application logic and data deployed in the network. And by *communications* I mean the ability for end users to interact through these clients, to share data, text, audio, and video in real time and nonreal time. Rich clients should provide these capabilities in an integrated manner, where the applications that can target them far exceed what is possible in the world of HTML documents.

Rich clients should allow applications to run in browsers, but also as standalone applications on desktops and laptops. They should also support running on devices. To support these new Internet-connected application types, they should support offline data storage, enabling occasionally connected devices and applications.

Most important, rich clients should anticipate the emerging world of back-end Web services by using a services-oriented architecture for integrating business logic and data across the network, whether that's simply contained in an application server or actually a distributed Web service exposed through a protocol like SOAP.

Rich clients and Web services combined hold the promise of fulfilling the broader vision that the industry has for deploying software as services. The combination will enable rich, business-connected productivity applications. It will transform what's possible on the Internet today. From now on, when people ask you what Web services are and why they're significant, make sure your worldview encompasses this broader perspective on software as services.

Macromedia MX and Web Services

As might be expected, Macromedia has focused on delivering a holistic approach to Web services, one that provides transformative technology for the client, server, and development-tools aspects of Web services.

ColdFusion MX: An Approachable Web Services Environment

Starting with the more common world of Web services as component middleware, Macromedia has introduced major new Web services capabilities in ColdFusion MX, the latest release of this rapid server scripting environment. For those not familiar with CFMX, it's a fundamental shift and repositioning for ColdFusion in the industry, moving from being a proprietary application server to a rapid development and scripting environment that runs on any popular application server, such as IBM WebSphere or Sun's Sun ONE.

Specifically with Web services, Macromedia focused on delivering a Web services engine that would empower RAD and scripting-level developers to easily construct and consume Web services. Using a technology called ColdFusion Components, or CFCs, scripting developers can add simple tags that provide metadata for defining a service and then include arbitrary script inside that metadata. The result is a well-defined component that provides a services-based interface (see Figure 1). Underneath, CFCs are dynamically compiled into JavaBeans and hot-deployed into the containing application server. This gives the mass of scripters the ability to easily construct server-side logic that can immediately be exposed through SOAP and can also be used by rich clients, such as Macromedia Flash Player.

The ColdFusion MX Web services engine is actually based on Macromedia's active involvement in the Apache Axis project, with strong contributions from IBM and others in the Apache community. It is the first commercial server to use this powerful new open-source Web services platform.

While CFCs allow developers to create Web services easily, ColdFusion MX also includes approachable capabilities for consuming and using Web services. Developers can declaratively invoke any Web service and Web service method at runtime; ColdFusion will dynamically generate Java client proxies that handle all the SOAP interactions, and actually transform SOAP results into local variables that can be used from script. Developers don't need to think about parsing SOAP or WSDL, or even of manually building client proxy skeletons, let alone having to invoke interfaces on those proxies.

Web services can also be imported into a script and used as tag libraries, with each method on a Web service exposed as a custom tag, enabling even HTML-level designers and programmers to use Web services.

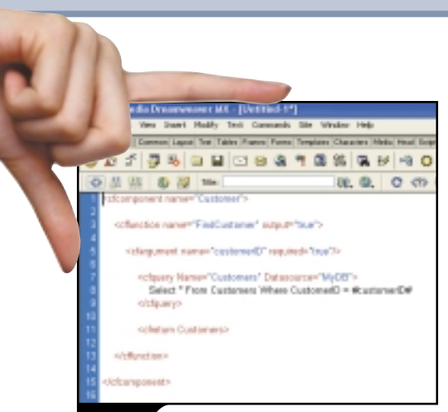


FIGURE 1: ColdFusion Components offer a simple syntax for building Web services.

FIGURE 2: Flash MX introduces powerful UI components.

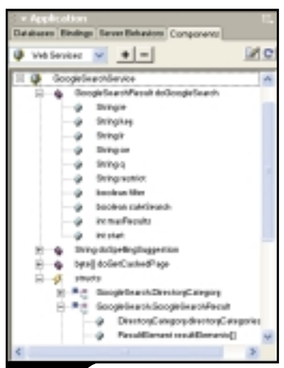


FIGURE 3: Dreamweaver MX enables easy browsing and use of Web services.

Flash Player 6.0: The Internet's Premier Rich Client

In line with our more holistic view of software as services, Macromedia is leading the way with the introduction of a powerful rich client for delivering Internet applications. Macromedia Flash Player has evolved into being a complete environment for delivering rich user experiences across browsers, operating systems, and devices.

Over the last several years, Macromedia Flash Player has emerged as the most ubiquitous client runtime on the Internet. With more than 98.3% of Internet end users having a version of the player, it is now the most widely deployed software environment in the history of computing. Since the introduction of Flash Player 6.0, we've distributed well over 100 million new runtimes, averaging almost 3 million downloads per day. Packed into this new player are powerful new capabilities for delivering applications. Flash Player now provides a more complete programming environment, a visual component model, and fantastic new rich-content types (see Figure 2).

Most important, we've introduced technology for connecting rich client applications with back-end services. Dubbed Macromedia Flash Remoting, this technology provides a high-performance connection between rich clients and back-end services, no matter what the back-end platform or model. Flash Remoting supports using services deployed as CFCs and scripts, Java classes, JavaBeans, EJBs, JMX Beans, C# objects, ADO.NET, and ASP.NET pages. With these services it provides a high-performance binary connection – running over HTTP – between server objects and client objects.

Additionally, Macromedia Flash Remoting provides a simple mechanism for connecting Flash to SOAP Web services, even services deployed on networks and servers that are different from the application's origin domain. Crucially, browser-contained Flash applications follow a strict security sandbox, where they can make requests only to their originating domain. Flash Remoting provides a server-based proxy to any SOAP Web services.

Nonetheless, the Macromedia Flash Remoting model uses a services-based architecture that is back-end independent: client code can use services without ever knowing the type of back-end implementation.

Dreamweaver MX: Creating and Using Web Services

Two tools in Dreamweaver MX stand out with respect to creating and using Web services. Dreamweaver MX includes development tools for creating CFCs for easy authoring, providing one of the quickest ways to create SOAP-accessible Web services. It also includes significant tools for using other people's Web services. The Dreamweaver MX Web services browser allows you to point to any WSDL file or UDDI repository and import a Web service into the development environment. This provides a visual browser of the Web service's interface, and will even generate client proxies for .NET and Java. Once developers find a method they want to use, they can drag-and-drop these into their code and Dreamweaver will create the basic invocation code automatically.

Studio MX: Tools for Creating Web Services

With the MX product family Macromedia has both broadened and focused its suite of design and development tools by providing an integrated suite called Macromedia Studio MX. Studio MX provides everything needed to rapidly create everything from content and graphics to rich and complex user interfaces, server-side logic, and components that interact with XML, databases, and Web services (see Figure 3).

In the MX family Macromedia Flash MX provides the primary authoring environment for creating rich client user interfaces, evolving from its early heritage as a motion graphics design environment to a complete application development tool for client-side applications. Likewise, Dreamweaver MX has also been transformed from being the premier Web authoring and site development package to a complete IDE for Web application development, including powerful new programming and database tools and support for XML and Web services.

Software as Services: Make It Real

Macromedia is passionate about the idea of using software as services. We believe in a holistic view that encompasses both the front-end user experience and the back-end integration layer. Our hope is that by making Web services approachable and affordable, all of us – not just the legions of advanced programmers and classic middleware developers – will be able to take advantage of what they have to offer. Building for the next generation of the Internet should be fun – let's make the Internet interesting again!

About the Author

Jeremy Allaire, CTO of Macromedia, is instrumental in guiding the company's product direction and is its primary technology evangelist, responsible for establishing key strategic partnerships within the Internet industry. Jeremy ia a cofounder and the former CTO of Allaire Corporation, which merged with Macromedia in March 2001.

@ JALLAIRE@MACROMEDIA.COM

Using ColdFusion Components Part 1

BY BEN FORTA

CFCs display the power of objects and the simplicity of CFML

In the May issue of ***CFDJ*** (Vol. 4, issue 5) I introduced you to ColdFusion MX – the just-released ColdFusion upgrade that is nothing short of revolutionary. As I explained in that article, CFMX features lots of incredible new technologies and features. But the one I think is most important (in terms of how it will, or should, impact your development) has to be ColdFusion Components. This month (and continued next month) I'd like to explain in depth what CFCs are and how they should be used.

CFCs Are Objects, Kind Of

To understand CFCs it's important to understand a bit about objects.

But first, a necessary qualification – CFML is not an object-oriented language, and CFCs don't provide all the features and functionality typically provided by OOP languages. This isn't a bad thing – actually, I think the opposite is true. Objects have too long been the exclusive property of languages and syntactical rules that are unnecessarily complicated. There's no reason for developers using rapid development languages, like CFML, not to be able to leverage this type of technology too. Thus the CFC.

At their simplest, and at the risk of offending OOP purists, objects are simply reusable application bits. They are black boxes – magical things that do stuff, whatever you define that stuff to be. If this sounds a bit like custom tags, well, there are similarities, but objects typically do more than custom tags. For example, they often contain not just code (like custom tags) but also data, allowing data and any code that accesses it to be cleanly encapsulated. Objects usually have multiple entry points (methods). They provide a mechanism to automatically run initialization code regardless of the entry point (a constructor). Objects can be adapted and modified, leveraging existing code without actually modifying (and potentially breaking) any of it in the process (inheritance).

If this all sounds a little esoteric, an example should help clarify things.

Objects Demystified

Almost any ColdFusion developer has had to deal with users in their applications. You've probably done things like:

- Used <CFQUERY> to check a user-provided name and password
- Used returned database rows to obtain a user's name and e-mail address
- Used programmatic flow-control to grant or deny access to features based on login name, access level, or security equivalences
- Saved and read user preferences like colors, fonts, language choices, topic and category selections, and more
- Added new users or updated or deleted existing ones
- …and so on

In doing these things, how many places in your code ended up referring to users in some way, shape, or form? How much code did you end up repeating (or copying and pasting)? What would happen if the user tables had to change, and how much code would that affect? What if your databases were replaced entirely by an LDAP server, or new sets of rules and levels had to be applied? How much did nonprogrammers have to know about the guts of your user management to be able to do simple things like display a personalized welcome message?

In an object world you'd have written your application very differently. You'd have created a user object – a black box that contained everything and anything you'd ever need to do with users, as well as all the database (or LDAP) access needed to perform those operations.

To perform operations on a user, you'd then create an instance of the user object, perhaps specifying the user ID in the process. Then you'd be able to use methods (functions) like:

- Add()
- Delete()
- Update()
- Get()

Or even more specific ones, like:

- VerifyPassword()
- GetEMail()
- GetColorPrefs()
- GetDisplayName()

…the idea being that any and all user processing would happen inside the user object. You'd never access user tables directly. Actually, you wouldn't even know (or care) that the data is being stored in a table – that wouldn't matter at all. You'd just invoke methods as needed, letting the code inside the object do its thing.

This is the kind of functionality made possible using CFCs.

Introducing CFCs

Now that you know what CFCs are, let's look at how they're created. True to ColdFusion and its mission, CFCs are incredibly easy to create. In fact, only two steps are involved:

1. Create a file with a .CFC extension (this distinguishes CFCs from ColdFusion templates, which have a .CFM extension).

SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9600

PUBLISHER, PRESIDENT, and CEO
Fuat A. Kircaali fuat@sys-con.com

BUSINESS DEVELOPMENT
VP, Business Development
Grisha Davida grisha@sys-con.com
Chief Operating Officer
Mark Harabedian mark@sys-con.com

ADVERTISING
Senior VP, Sales & Marketing
Carmen Gonzalez carmen@sys-con.com
VP, Sales & Marketing
Miles Silverman miles@sys-con.com
Advertising Director
Robyn Forma robyn@sys-con.com
Advertising Account Manager
Megan Ring megan@sys-con.com
Associate Sales Managers
Carrie Gebert carrieg@sys-con.com
Kristin Kuhnle kristin@sys-con.com
Alisa Catalano alisa@sys-con.com
Leah Hittman leah@sys-con.com

SYS-CON EVENTS
VP, Events
Cathy Walters cathyw@sys-con.com
Conference Manager
Michael Lynch mike@sys-con.com
Sales Executives, Exhibits
Michael Pesick michael@sys-con.com
Richard Anderson richarda@sys-con.com

CUSTOMER RELATIONS / JDJ STORE
Manager, Customer Relations / JDJ Store
Anthony D. Spitzer tony@sys-con.com
Customer Service Representative
Margie Downs margie@sys-con.com

WEB SERVICES
Web Designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
Catalin Stancesco catalin@sys-con.com
Online Editor
Lin Goetz lin@sys-con.com

ACCOUNTING
Chief Financial Officer
Bruce Kanner bruce@sys-con.com
Assistant Controller
Judith Calnan judith@sys-con.com
Accounts Receivable
Jan Braidech jan@sys-con.com
Accounts Payable
Joan LaRose joan@sys-con.com
Accounting Clerk
Betty White betty@sys-con.com

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

2. Use four new tags to create the components, define their functions and arguments, and return a value.

The four new tags are:
- <CFCOMPONENT>: Defines a CFC
- <CFFUNCTION>: Defines the functions (methods) within a CFC
- <CFARGUMENT>: Defines the arguments (parameters) that a function accepts
- <CFRETURN>: Returns a value or result from a function

Other than these new tags (which, incidentally, are the same ones used to create tag-based, user-defined functions in CFMX), CFCs are plain CFML. Within a CFC you can use any tag, function, custom tag, component, and more.

Creating CFCs

Listing 1 offers a simple example to clarify all this, a component that does basic browser identification (determines the browser being used).

This is a simplified version of a file named browser.cfc. As you can see, the code is wrapped between <CFCOMPONENT> and </CFCOMPONENT> tags. The component itself has a single function, IsIE (as in "Is Internet Explorer"), and is defined and named with a <CFFUNCTION> tag. Within the function a variable named *result* is defined and initialized; if the browser identifier contains the Microsoft Internet Explorer identifier, that variable is then set to "yes." And finally, <CFRETURN> returns "result."

This isn't a complete example yet (although it's actually workable as is), but it's a good starting point.

How would you use this code? Save it as browser.cfc, and in the same directory create a test.cfm file that should contain the following:

```
<!--- Invoke browser CFC --->
<CFINVOKE COMPONENT="browser"
          METHOD="IsIE"
          RETURNVARIABLE="result_ie">

<!--- Feedback --->
<CFOUTPUT>
Your browser is:<BR>
IE: #YesNoFormat(result_ie)#<BR>
</CFOUTPUT>
```

<CFINVOKE> is a new tag in CFMX and one of its jobs is to invoke component methods. As seen here, <CFINVOKE> takes the name of the component (*browser*, minus the extension) and the method to execute. To access any returned data, the RETURNVARIABLE attribute provides the name of a variable to contain whatever the function returns.

You can try running this code. If you are running IE it will tell you so, and if not it'll tell you that too.

Writing Complete CFCs

Now that you're familiar with how CFCs work, take a look at Listing 2, a complete version of browser.cfc. This new version is a bit more complex, so let's walk through it together.

<CFCOMPONENTS> now contains four methods:
- IsIE: Checks for IE, returning "yes" or "no"
- IsNetscape: Checks for Netscape, returning "yes" or "no"
- IsDreamweaver: Checks for Dreamweaver (the browser built into Dreamweaver), returning "yes" or "no"
- Identify: Checks for all three, returning "IE" if Internet Explorer, "NS" if Netscape, "DW" if Dreamweaver, and an empty string if unknown (none of the above)

You'll notice that we've added two attributes to every <CFFUNCTION>:
- RETURNTYPE: Specifies the type of the return data. This is used to validate returned data (if it's the wrong type, an error would be thrown) and to document the function.
- HINT: A text description used when the function is documented (more on that in a moment)

In the earlier version of browser.cfc the code checked the CGI variable HTTP_ USER_ AGENT to determine the browser used. But that's not ideal because these functions should be able to check any browser ID passed to them (assuming they're not always checking the current browser).

And so, each of these new functions takes an optional parameter defined with the <CFARGUMENT> tag, which can be used in two distinct ways:
1. To check for required parameters, set REQUIRED="yes"; if the specified parameter isn't passed, an error will be thrown. Additional validation may be performed by specifying a TYPE, in which case ColdFusion checks that the parameter exists and that it is of the correct type.
2. To check for optional parameters, set REQUIRED="no" and specify the DEFAULT value to be used if the parameter isn't provided.

In all four functions <CFARGUMENT> defines a single parameter, browser, allowing calling code to pass any string as needed. If the parameter isn't provided, the default CGI variable is used as before.

The first three functions are basically the same as the one we looked at earlier, but the fourth, identify, which provides an alternate interface to the browser identification code, warrants special mention. Instead of checking for a specific browser (as the three prior functions do), this one checks for them all and returns an ID based on which browser it is. Rather than copy all of the previous code into this new function, it simply invokes those functions directly. But it doesn't invoke them using the <CFINVOKE> tag seen previously; rather, it uses them as functions (which they are), making the syntax much cleaner and simpler.

Listing 3, a revised version of test.cfm, can be used to test all the functions.

It's as simple as that.

Documentation

I mentioned that HINT and RETURNTYPE are used in documenting CFCs. So where is the documentation?

CFCs may be introspected – a fancy way of saying that they can be asked to describe themselves. As they know all about their functions and what they do, they can fulfill this request quite admirably.

Here are two ways to introspect a CFC:
1. Execute a CFC directly in your browser, providing a URL directly to it (something like http://local-host/path/browser.cfc). ColdFusion will generate cleanly formatted documentation on-the-fly, using everything it knows about the component (including the specified HINTs).
2. If you're using Dreamweaver MX, select the Components tab in the application window to have Dreamweaver scan for available CFCs. Dreamweaver will then build a tree control containing all components and their methods. You may right-click on any component or method for more information (or to edit the code), and may even drag a method into the editor window to generate a complete <CFINVOKE> tag.

There's more to introspection, but we're out of space for now.

Summary

CFCs provide the power of objects with the simplicity of CFML. As seen here, CFCs are created using four new tags, saved as .CFC files, and invoked using the <CFINVOKE> tag. But we've barely scratched the surface here. In fact, thus far we've been using CFCs merely as glorified custom tags. Next month I'll show you alternate ways to invoke CFCs, how to make them persist, how to use constructors and inheritance, and much more. And we'll come back to the user explanation we started with.

@ BEN@FORTA.COM

ABOUT THE AUTHOR
Ben Forta is Macromedia's senior product evangelist and the author of numerous books, including ColdFusion 5 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 5 Development. *Ben is working on several new titles on ColdFusion MX. For more information visit www.forta.com.*

Listing 1

```
<!--- Browser id component --->
<CFCOMPONENT>

<!--- Is the browser IE? --->
<CFFUNCTION NAME="IsIE>

 <!--- Init variable --->
 <CFSET result="No">

 <!--- Look for IE identifier --->
 <CFIF FindNoCase("MSIE", CGI.HTTP_USER_AGENT)>
  <!--- Yep, got it --->
  <CFSET result="Yes">
 </CFIF>

 <!--- Return result --->
 <CFRETURN result>

</CFFUNCTION>

</CFCOMPONENT>
```

Listing 2

```
<CFCOMPONENT>

<!--- Is the browser IE? --->
<CFFUNCTION NAME="IsIE"
             RETURNTYPE="boolean"
             HINT="Is browser Microsoft IE">

 <!--- If no browser id passed, used current --->
 <CFARGUMENT NAME="browser"
              REQUIRED="no"
              DEFAULT="#CGI.HTTP_USER_AGENT#"
              HINT="Browser ID, defaults to CGI ID">

 <!--- Init variable --->
 <CFSET result="No">

 <!--- Look for IE identifier --->
 <CFIF FindNoCase("MSIE", browser)>
  <!--- Yep, got it --->
  <CFSET result="Yes">
 </CFIF>

 <!--- Return result --->
 <CFRETURN result>

</CFFUNCTION>

<!--- Is the browser Netscape? --->
<CFFUNCTION NAME="IsNetscape"
             RETURNTYPE="boolean"
             HINT="Is browser Netscape">

 <!--- If no browser id passed, used current --->
 <CFARGUMENT NAME="browser"
              REQUIRED="no"
              DEFAULT="#CGI.HTTP_USER_AGENT#"
              HINT="Browser ID, defaults to CGI ID">

 <!--- Init variable --->
 <CFSET result="No">

 <!--- Look for Netscape identifier and no IE identifier -
-->
 <CFIF FindNoCase("mozilla", browser)
       AND NOT FindNoCase("MSIE", browser)>
  <!--- Yep, got it --->
  <CFSET result="Yes">
 </CFIF>

 <!--- Return result --->
 <CFRETURN result>

</CFFUNCTION>

<!--- Is the browser Dreamweaver? --->
<CFFUNCTION NAME="IsDreamweaver"
             RETURNTYPE="boolean"
             HINT="Is browser Dreamweaver">

 <!--- If no browser id passed, used current --->
 <CFARGUMENT NAME="browser"
              REQUIRED="no"
              DEFAULT="#CGI.HTTP_USER_AGENT#"
              HINT="Browser ID, defaults to CGI ID">

 <!--- Init variable --->
 <CFSET result="No">

 <!--- Look for DW identifier --->
 <CFIF FindNoCase("mmhttp", browser)>
  <!--- Yep, got it --->
  <CFSET result="Yes">
 </CFIF>

 <!--- Return result --->
 <CFRETURN result>

</CFFUNCTION>

<!--- Identify a browser
 Returns: IE - Internet Explorer
          NS – Netscape
          DW – Dreamweaver
          Empty string is unknown
--->
<CFFUNCTION NAME="Identify"
             RETURNTYPE="string"
             HINT="Identify a browser">

 <!--- If no browser id passed, used current --->
 <CFARGUMENT NAME="browser"
              REQUIRED="no"
              DEFAULT="#CGI.HTTP_USER_AGENT#"
              HINT="Browser ID, defaults to CGI ID">

 <!--- Init variable --->
 <CFSET result="">

 <CFIF IsIE(browser)>
  <CFSET result="IE">
 <CFELSEIF IsNetscape(browser)>
  <CFSET result="NS">
 <CFELSEIF IsDreamweaver(browser)>
  <CFSET result="DW">
 </CFIF>

 <!--- Return result --->
 <CFRETURN result>

</CFFUNCTION>

</CFCOMPONENT>
```

Listing 3

```
<!--- Check for IE --->
<CFINVOKE COMPONENT="browser"
          METHOD="IsIE"
          RETURNVARIABLE="result_ie">

<!--- Check for Netscape --->
<CFINVOKE COMPONENT="browser"
          METHOD="IsNetscape"
          RETURNVARIABLE="result_ns">

<!--- Check for DW --->
<CFINVOKE COMPONENT="browser"
          METHOD="IsDreamweaver"
          RETURNVARIABLE="result_dw">

<!--- Identify browser --->
<CFINVOKE COMPONENT="browser"
          METHOD="Identify"
          RETURNVARIABLE="result_id">

<!--- Feedback --->
<CFOUTPUT>
Your browser is:<BR>
IE: #YesNoFormat(result_ie)#<BR>
NS: #YesNoFormat(result_ns)#<BR>
DW: #YesNoFormat(result_dw)#<BR>
ID: #result_id#<BR>
</CFOUTPUT>
```

CODE LISTING

The code listing for this article is also located at

www.sys-con.com/coldfusion/sourcec.cfm

Do You **Object?**

Structures + user-defined functions = CFCs

BY HAL HELMS

As I write this, ColdFusion MX is widely available in a preview release and it looks like it's going to be an excellent one.

CFMX offers several features to ColdFusion developers, including native support for reading and writing XML, Unicode support, better Flash widgets, and Web services. Perhaps the biggest difference, though, is the introduction of what they're calling *ColdFusion Components* – CFCs for short.

CFCs are the integration of two previous ColdFusion pieces: structures and user-defined functions. Structures are wonderful things that allow us to model the idea of a *thing* that has *properties*. The structure can be anything you wish – a model of something physical, or conceptual. An employee is an example of a model of something physical. Employees have properties. What properties you choose to include will be determined by how you plan to use the structure. Properties commonly associated with employees include employeeID, firstName, lastName, dateHired, and department. The code for this is:

```
<cfscript>
 Employee = StructNew();
 Employee.employeeID = '100';
 Employee.firstName = 'Hal';
 Employee.lastName = 'Helms';
 Employee.dateHired = '4/1/1999';
 Employee.department = 'Training';
</cfscript>
```

Instead of my having to keep track of multiple separate variables, with a structure I can work with a single complex variable, using any of the structure's properties when I need to, as in this code snippet:

```
<cfoutput>Hello,
#Employee.firstName#!</cfoutput>
```

If I want information about the employee to persist across page (HTTP) requests, I can set the scope of this single variable to a persistent scope:

```
<cfset Session.Employee =
Duplicate( Employee )>
```

Structures are wonderfully useful, yet are often avoided by inexperienced developers who may be afraid of their seeming complexity (and of their cousins, arrays). This is too bad, as complex variables actually make coding simpler and easier; for this reason I cover arrays and structures in the introductory ColdFusion class I teach.

User-defined functions were introduced to ColdFusion in version 5, much to the happiness of ColdFusion developers. Now you were no longer restricted to ColdFusion's native functions, but could write your own functions. Here's one I wrote that mirrors a popular algorithm used by some managers to decide whether to spend money on improving working conditions for developers:

```
<cfscript>
function decideOnImprovement(
workingConditionRequested ){
 return false;
}
</cfscript>
```

This function could then be used in the same context as ColdFusion's built-in functions:

```
<cfif decideOnImprovement(
'ergonomic chairs' )>
 Of course. This will pay
itself back in a matter of days.
</cfelse>
 Sorry, no. Several management
studies show that workers are
far more productive when they're
cramped and uncomfortable.
</cfif>
```

User-defined functions can be used within user-defined functions, as shown in this other popular algorithm for determining whether to spend money:

```
<cfscript>
function feelingGood(){
 return RandRange( 0, 1 );
}

function decideOnSpending(
amountRequested ){
 if (feelingGood() AND
amountRequested LT 100){
  return true;}
 else return false;
}
</cfscript>
```

Functions can then be called from standard ColdFusion pages:

```
<cfif decideOnSpending(
form.amountNeeded )>
 Spend away!
<cfelse>
 Go away!
</cfif>
```

ColdFusion Components combine these structures and user-defined functions into a whole that is definitely more than the sum of the parts. With CFCs you don't just create Employee; you create a factory for creating employees. CFCs combine structure information about an employee with UDFs that prescribe behaviors of Employee. In CFCs those UDFs are called *methods*.

In the previous code that created an employee structure, we ended up with a structure that defined a single employee. I can display the employee's first name with this line of code:

```
<cfoutput>#Employee.firstName#</
cfoutput>
```

ABOUT THE AUTHOR

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

Nice enough, but what if I want to create another employee? Hmmm…I'll have to create a new structure: Employee2, perhaps. But look at this code for Employee.cfc:

```
<cfcomponent>
 <cffunction name="new">
  <cfargument name="firstName"
type="string" />
  <cfargument name="lastName"
type="string" />
  <cfset this.lastName = argu-
ments.firstName />
  <cfset this.firstName = argu-
ments.lastName />
  <cfset this.salary =
RandRange( 100000,200000 ) />
  <cfreturn this />
 </cffunction>
</cfcomponent>
```

I can create a new employee in several ways. Using the <cfinvoke> tag, it looks like this:

```
<cfinvoke
  component="test"
  method="new"
  firstName="Hal"
  lastName="Helms"
  returnVariable="aPerson" />
```

Or I can use the CreateObject() function:

```
<cfset me = CreateObject( "com-
ponent","test" )>
<cfset me.new( 'Hal', 'Helms'
)>
```

To create a new employee for you, I could give you a form asking for your first and last names. When you submit the form, I use this code to create a new employee.

```
<cfset you = CreateObject(
"component","test" )>
<cfset you.new(
'#form.firstName#',
'#form.lastName#' )>
```

Nice, don't you agree? By wrapping together all the information about who an employee is and what an employee can do, I can simplify the way I think about my code. When I want to know something about you or me, I can find out from a single CFC-created object. In object-oriented programming, we call that *encapsulation*.

There's no doubt that CFCs will offer wonderful new capabilities to ColdFusion – and to ColdFusion programmers. But I'm concerned. In fact, I'm worried. I'm worried because I've noticed a small but growing "crisis of confidence" among ColdFusion programmers, who feel that the new features of ColdFusion MX – in particular, CFCs – will exclude them from programming.

I see this fear reflected in the e-mail sent to me:

With CFCs, will we all have to learn object-oriented theory? I'm not a trained programmer and the idea scares me.

I hear it on talk lists, where developers voice their worries that they just don't understand it all. Worst of all, I hear certain, more experienced developers strengthen and encourage these fears. On talk lists they dismiss newbies by responding with "RTFM" to people struggling with these new concepts. Such experts use their knowledge as a club to beat down those whom they should be helping.

Object-oriented concepts were created to help programmers simplify the complexity of programming, not to make our lives harder. Of course, some implementations of OO seem to forget this. As the legendary Alan Kay once said, "I invented the term *object oriented*, and I can tell you I did not have C++ in mind." Kay was the creator of Smalltalk, the first programming language I learned, which is still considered the purest of OO languages. He designed it so that fifth-grade students would be able to use it. Profound does not mean complicated.

The people at Macromedia seem to "get" this. Rather than turning ColdFusion into an OO language, I see every indication that ColdFusion MX will continue to do what ColdFusion has always done: make life simpler for developers. With CFCs, ColdFusion wraps some of the power of objects in an easy-to-learn and easy-to-use component. Try them out and you'll find new ways to make your code more powerful and easier to manage.

HAL@FUSEBOX.ORG

ColdFusion MX and XML: **Parsing XML** Part 1

Be comfortable with XML files without learning another language

BY DAVID GASSNER

XML– it's here, and you have to deal with it. Whether you need to generate XML files to share data with your business partners, extract content from XML files, or transform XML into some other format...

...the ability to deal with XML has become a common requirement for ColdFusion developers in the past couple of years.

Prior to the release of ColdFusion MX, the <CFObject> tag gave developers a way to integrate ColdFusion with existing XML parsers, including COM objects from Microsoft (also known as *MSXML*) and Java-based libraries from Apache and other vendors. These components have powerful APIs, but their complexity has discouraged some from joining the XML "revolution." And while third-party solutions have made the processing of XML easier, CFMX now includes XML parsing and creation tools that deliver the ease of use and simplicity we expect from ColdFusion development.

This is the first of three articles describing the new functionality. In this article I'll describe the tools for parsing, extracting, and searching for data from XML. The next two articles will cover the creation of XML files and the use of XSLT (Extensible Stylesheet Language Transformation) to turn XML-formatted content into other text formats. One caveat: this article is based on prerelease software, and there may be minor changes in the final version. For complete details click on the Documentation link in the CFMX Administrator. Click on "Developing ColdFusion MX Applications with CFML," then see Chapter 30, "Using XML and WDDX."

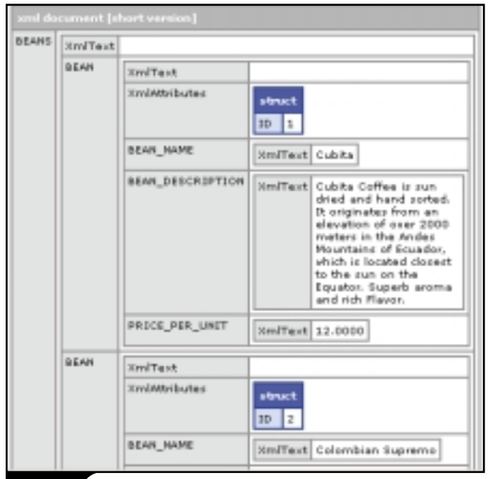


FIGURE 1: XML document object short form

Reading a File

XML parsing technologies can be divided into two categories: event-driven parsers such as SAX (Simple API for XML) and tree-style parsers such as DOM (Document Object Model) and JDOM. The new CFMX tools are modeled on the latter. With these parsers the entire XML file is first read into memory. The text version of the XML is then converted to an in-memory object model, which is navigable through the parser's object structure. Most tree-based parsers require you to learn the names and methods of various software objects to find and retrieve data. In CFMX the XML document is exposed as a variable that behaves in many ways like existing ColdFusion variable types. If you know how to use structures and arrays, you already know most of what you need to deal with XML.

The first step is to read the contents of the XML file into a variable. For this purpose you'll use familiar tools. If the file is available through the local file system, read it with <CFFile>:

```
<CFFile action="READ" variable="xml"
  file="#ExpandPath('.')#\beans.xml">
```

If the file is on a remote server, read it with <CFHTTP>:

```
<CFHttp url="http://localhost/beans.xml"
  method="get">
<CFSet xml=CFHTTP.FileContent>
```

Beans.xml is an XML version of the Beans table from Macromedia's Fast Track to ColdFusion class (www.macromedia.com/support/training). See Listing 1 for its contents.

The XML Document Object

Now comes the fun part: turning the XML file's contents into an object structure takes just one line of code. CFMX introduces a new complex variable type called an *XML document object* and a new function called *XMLParse()*, which knows how to create these new XML objects from text variables. XMLParse() takes the text value of an XML document or document fragment as its only argument and returns the resulting object. Here's the code:

```
<CFSet xmlDoc = XMLParse(xml)>
```

That's it. Now you're ready to retrieve your data. If you want to see the resulting XML document's structure, dump it to the screen with the <CFDump> tag. This debugging utility tag has been in ColdFusion since version 5.0, and has been enhanced to deal with the XML variable type. Use this code to view the structure of your XML document:

```
<CFDump var="#xmlDoc#">
```

See Figure 1 for an example of the output. Note that the CFDump output header says "xml document [short version]." There are two ways of addressing XML content. By default, <CFDump> presents the structure in shorthand (known as the "Basic" structure), the version you'll be most interested in. To see the longhand (or "DOM") model, click the <CFDump> header and the output expands significantly. Click a third time to collapse the tree completely, and again to return to the shorthand presentation.

Retrieving Data

There are two ways to address the parts of an XML document object. The first is the DOM approach. In this version the document object has a child object called XmlRoot, which references the root element of the XML file. This and every other element has a set of child objects with fixed names. The most commonly used child objects are those found in Table 1.

Examine this simple XML fragment:

```
<Software>
  <ColdFusion version="6">J2EE
  Server</ColdFusion>
</Software>
```

To retrieve the value of the "ColdFusion" element from the resulting xmlDoc object, the long form address would be:

```
xmlDoc.XmlRoot.XmlChildren[1].XmlText
```

To retrieve the value of the "version" attribute, the syntax would be:

```
xmlDoc.XmlRoot.XmlChildren[1].XmlAttributes.version
```

This syntax is a bit heavy, though, and presents some problems. For instance, if the XmlChildren array contains elements with different names, you'd have to loop through the array to find the element you're looking for. So CFMX also provides the shorthand (Basic) addressing system. The Basic syntax for addressing content is similar to that for getting information from a ColdFusion structure or array.

Each uniquely named element within its parent is addressed by its name, just like a structure item. If you want to get the value of the text in the uniquely named "ColdFusion" element, and you know the name of the element in advance, the value's address becomes:

```
xmlDoc.Software.ColdFusion.XmlText
```

And reading an attribute becomes:

```
xmlDoc.Software.ColdFusion.XmlAttributes.version
```

If you don't know the names of elements or attributes in advance, replace the dot notation with array syntax, like this:

```
xmlDoc.Software[softwarename].XmlAttributes[attributename]
```

Note: While XML is a case-sensitive technology, ColdFusion follows its usual practice and exposes all element and attribute names as case-insensitive variable names.

XMLName	The name of the element
XMLText	The concatenated value of the element's child text and CDATA nodes
XMLComment	The concatenated value of the element's child comments
XMLAttributes	A structure containing the element's attributes
XMLChildren	An array of child elements

TABLE 1 Commonly used child objects

Looping Through Elements

If there's more than one child element of the same name within a parent element, you can address them as an array of elements, where the array name matches the element name. For the Beans.xml file, output a list of coffee bean names like this:

```
<cfset aBean=xmlDoc.Beans.Bean>
<cfloop from="1" to="#ArrayLen(aBean)#"
  index="i">
  <cfoutput>#aBean[i].Bean_Name.XmlText#<br></cfout
  put>
</cfloop>
```

See Listing 2 for the CF code to output the complete XML file as a list, and Figure 2 for its output.

Searching for XML Data

Just as with relational databases, you may want to query a particular element or set of elements based on the values of their attributes or text nodes. For this purpose CFMX adds a new function called XMLSearch(), which takes two arguments: an XML document object and an XPath expression. XPath, a powerful querying language for XML structures, is implemented in many XML technologies, including XSLT, XPointer, and many XML databases.

Here's a sample XPath query that extracts a single Bean element from our beans.xml file based on the Bean element's ID attribute:

```
<cfset aBeans=XMLSearch(xmlBeans,
  "//BEAN[@ID=1]")>
```

This XPath pattern means: find elements named "BEAN" where the element's ID attribute value equals 1. The double-slash ("//") means the element may be anywhere in the XML hierarchy, so we don't have to drill down one level at a time. XMLSearch() returns an array of elements that match the pattern. In this example, assuming that each Bean's ID is unique, we'd get back an array with a single item: the Bean element with the ID of 1.

There's a lot more to XPath, and it's a language worthy of study. We'll cover it more thoroughly in the third article, as it's a key part of XSLT.

Once we've retrieved the array, it's a simple matter to address the name of the one bean we found:

```
#aBeans[1].Bean_Name.XmlText#
```

See Listing 3 for the CF code to output the details of a single Bean element.

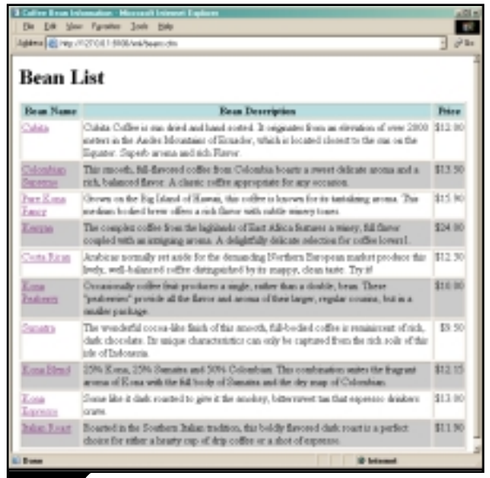


FIGURE 2: Output the XML elements as HTML

Completing the Application

The two files described above, bean_list.cfm and bean_detail.cfm, display, respectively, a list of all XML elements and the detail information about a single element. The last file in the set, beans.cfm, is a control file that calls either the list or the detail view, depending on whether the user has provided a Bean ID in the URL. See Listing 4 for the code.

Limitations

The new ColdFusion XML parsing tools have some limitations:

- ***Large-file handling:*** As a tree-based parser, ColdFusion must allocate enough memory to store the object structure for the entire XML file. As a result, there will be performance and resource-management problems when trying to read the contents of very large XML files. If you encounter one of these files, you'll need to use one of the industry-standard SAX parsers.
- ***Validation:*** This version of the new XML parsing tools doesn't provide validation against Document Type Definition (DTD) or XML Schema files. But if a DTD has been declared, it still must be available to the parser. This can be a problem: the parser is reading from a text variable rather than directly from a file, so it can't find DTD files that are referenced with relative addresses. For instance, the declaration <!DOCTYPE BEANS SYSTEM "beans.dtd"> always fails, as the filename has been provided without the entire path. Fix this either by expanding the name of the DTD file to include a complete address or by removing the DTD reference from the XML file.

Conclusion

ColdFusion's new XML document object follows the ColdFusion philosophy: take a common software development requirement that's fundamentally complex and make it simple. More ColdFusion developers can now be comfortable dealing with XML files as they encounter them without having to learn yet another language or API.

In Part 2 of this series I'll describe the new tools in ColdFusion MX for publishing your data as XML.

ABOUT THE AUTHOR

David Gassner is a Macromedia-certified instructor and the author of an upcoming course on XML development for ColdFusion developers. He teaches classes in ColdFusion, HTML, Java, and XML as well as other Web development skills.

@ DGASSNER@NAPANET.NET

Listing 1

```
<?xml version="1.0"?>
<BEANS>
  <BEAN ID="1">
    <BEAN_NAME>Cubita</BEAN_NAME>
    <BEAN_DESCRIPTION>Cubita Coffee is sun dried and hand
sorted. It originates from an elevation of over 2000 meters
in the Andes Mountains of Ecuador, which is located closest
to the sun on the Equator. Superb aroma and rich
Flavor.</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>12.0000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="2">
    <BEAN_NAME>Colombian Supremo</BEAN_NAME>
    <BEAN_DESCRIPTION>This smooth, full-flavored coffee
from Colombia boasts a sweet delicate aroma and a rich,
balanced flavor. A classic coffee appropriate for any occa-
sion.</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>13.5000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="3">
    <BEAN_NAME>Pure Kona Fancy</BEAN_NAME>
    <BEAN_DESCRIPTION>Grown on the Big Island of Hawaii,
this coffee is known for its tantalizing aroma. This medium
bodied brew offers a rich flavor with subtle winery
tones.</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>15.9000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="4">
    <BEAN_NAME>Kenyan</BEAN_NAME>
    <BEAN_DESCRIPTION>The complex coffee from the highlands
of East Africa features a winey, full flavor coupled with
an intriguing aroma. A delightfully delicate selection for
coffee lovers.</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>24.0000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="5">
    <BEAN_NAME>Costa Rican</BEAN_NAME>
    <BEAN_DESCRIPTION>Arabicas normally set aside for the
demanding Northern European market produce this lively,
well-balanced coffee distinguished by its snappy, clean
taste. Try it!</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>12.3000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="6">
    <BEAN_NAME>Kona Peaberry</BEAN_NAME>
    <BEAN_DESCRIPTION>Occasionally coffee fruit produces a
single, rather than a double, bean. These "peaberries" pro-
vide all the flavor and aroma of their larger, regular
cousins, but in a smaller package.</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>10.0000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="7">
    <BEAN_NAME>Sumatra</BEAN_NAME>
    <BEAN_DESCRIPTION>The wonderful cocoa-like finish of
this smooth, full-bodied coffee is reminiscent of rich,
dark chocolate. Its unique characteristics can only be cap-
tured from the rich soils of this isle of
Indonesia.</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>9.5000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="8">
    <BEAN_NAME>Kona Blend</BEAN_NAME>
    <BEAN_DESCRIPTION>25% Kona, 25% Sumatra and 50%
Colombian. This combination unites the fragrant aroma of
Kona with the full body of Sumatra and the dry snap of
Colombian.</BEAN_DESCRIPTION>
```

```
    <PRICE_PER_UNIT>12.1500</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="9">
    <BEAN_NAME>Kona Espresso</BEAN_NAME>
    <BEAN_DESCRIPTION>Some like it dark roasted to give it
the smokey, bittersweet tan that espresso drinkers
crave.</BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>13.0000</PRICE_PER_UNIT>
  </BEAN>
  <BEAN ID="10">
    <BEAN_NAME>Italian Roast</BEAN_NAME>
    <BEAN_DESCRIPTION>Roasted in the Southern Italian tra-
dition, this boldly flavored dark roast is a perfect choice
for either a hearty cup of drip coffee or a shot of
espresso. </BEAN_DESCRIPTION>
    <PRICE_PER_UNIT>11.9000</PRICE_PER_UNIT>
  </BEAN>
</BEANS>
```

Listing 2

```
<!--- bean_list.cfm
Displays a list of coffee
  beans from an XML file --->

<!--- Import the file from disk and parse into
  an XML document object --->
<cffile action="READ" variable="xml"
  file="#ExpandPath('.')#\beans.xml">
<cfset xmlBeans=xmlParse(xml)>

<h1>Bean List</h1>

<!--- Set an Array reference to
  Elements named "Bean" --->
<cfset aBeans=xmlBeans.Beans.Bean>

<!--- Start the HTML table --->
<table border="1">

  <!--- Header Row --->
  <tr bgcolor="aqua" valign="top">
    <th>Bean Name</th>
    <th>Bean Description</th>
    <th>Price</th>
  </tr>

  <!--- Loop through the list of Bean elements
    and output each Bean's name, description
    and price --->
  <cfoutput>
  <cfloop from="1"
    to="#ArrayLen(aBeans)#" index="i">
    <cfif i mod 2>
      <cfset bgcolor="white">
    <cfelse>
      <cfset bgcolor="silver">
    </cfif>
    <cfset id=aBeans[i].XmlAttributes.ID>
    <cfset price=aBeans[i].Price_Per_Unit.XmlText>
    <tr valign="top" bgcolor="#bgcolor#">
    <td><a href="beans.cfm?id=#id#">
      #aBeans[i].Bean_Name.XmlText#</a></td>
    <td>#aBeans[i].Bean_Description.XmlText#</td>
    <td align="right">#DollarFormat(price)#</td>
    </tr>
  </cfloop>
  </cfoutput>

</table>
```

Listing 3

```
<!---
bean_detail.cfm
Displays detail information about a single coffee bean
   from an XML file
--->

<!--- Import the file from disk and parse into XML doc
object --->
<cffile action="READ" variable="xml"
  file="#ExpandPath('.')#\beans.xml">
<cfset xmlBeans=xmlParse(xml)>

<!--- Search for the one bean in URL.ID --->
<cfset aBeans=XmlSearch(xmlBeans, "//BEAN[@ID=#URL.ID#]")>

<!--- If we didn't get back a single bean, return to list
--->
<cfif arrayLen(aBeans) neq 1>
  <cflocation url="beans.cfm">
</cfif>

<!--- Output bean detail --->
<cfset Bean=aBeans[1]>
<cfoutput>
<h1>#Bean.Bean_Name.XmlText#</h1>
<p>#Bean.Bean_Description.XmlText#</p>
<p>#DollarFormat(Bean.Price_Per_Unit.XmlText)#/lb.</p>
<hr>
<p><a href="#cgi.script_name#">Back to List</a></p>
</cfoutput>
```

Listing 4

```
<!---
  beans.cfm
  A file to call either the detail or list view,
    depending on whether a Bean ID has been provided
 --->

<!--- Begin HTML Output --->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
 <title>Coffee Bean Information</title>
</head>

<body>

<cfif IsDefined("URL.ID")>
  <!--- If the user clicked on a single bean,
    display detail info --->
  <cfinclude template="bean_detail.cfm">
<cfelse>
  <!--- If the page was called without a click,
    display a list of beans --->
  <cfinclude template="bean_list.cfm">
</cfif>

</body>
</html>
```

CODE LISTING

The code listing for this article is also located at

www.sys-con.com/coldfusion/sourcec.cfm

What's Online

www.sys-con.com/coldfusion

June **2002**

CFDJ Online

Visit www.sys-con.com/coldfusion and take advantage of the exciting options ***ColdFusion Developer's Journal* Online** has to offer! Stay ahead of the competition and be the first to know what's happening in the ColdFusion industry with the most interesting and up-to-date information. Learn about upcoming events, receive free newsletters, participate in discussion groups, find the best deals on the latest ColdFusion products, and more!

Web Services Edge 2002 West Conference

You won't want to miss the *i*-technology event of the year! Sign up today to take part in the Java, XML, Wireless, .NET, and Web Services Edge 2002 West Conference and Expo October 1–3 at the San Jose Convention Center in San Jose, California. This event will feature high profile, expert IT professionals speaking about the most important and exciting IT issues. The Readers' Choice Awards for ***Java Developer's Journal*** and ***XML-Journal***, known as the "Oscars of the software industry," will also be announced at this event. Come and get a first-hand look at the most important and influential products of the year!

CF Buyers Guide.com

CFBuyersGuide.com is the most-read CF resource on the Internet. This CF resource contains the best books, consulting services, content management tools, custom tags, database tools, design services, and e-business software available. Take advantage of this free listing – add a product or service that you think deserves recognition, or search through the existing information on the best CF resource on the Internet!

ColdFusion Jobs

Are you looking to break into the exciting world of ColdFusion? Want to expand your job options? Or are you an employer with available positions? Visit ColdFusion jobs to post your résumé, scroll through a listing of the hottest CF jobs around, or talk to employment experts. You can also visit our Career Guide for important news and tips, user groups for coaches and training, Tech-Agent to have great jobs e-mailed to you weekly, and much more! If you're an information technology professional and are curious about the job market, and if you demand privacy and don't want to waste time, you've found the right site!

ColdFusion Store CD Specials

Get *ColdFusion Developer's Journal – The Complete Library*, the most complete library of ***CFDJ*** articles in one CD! There are over 250 articles covering such topics as Custom Tags, Finding a Web Host, Using XML and XSLT with ColdFusion, Fusebox, and more! Keep them on hand for research and review. Order today and receive almost $10 off the regular price!

FREE TUTORIAL
JUNE 27TH WITH WEB SERVICES EDGE 2002 EAST REGISTRATION. LIMITED OFFER!

TO REGISTER: **www.sys-con.com or Call 201 802-3069**

web services world tour EDGE 2002

$195 REGISTRATION FOR SYS-CON SUBSCRIBERS
BEST EDUCATIONAL VALUE FOR THE HOTTEST TECHNOLOGY SKILLS!

Take Your Career to the Next Level!

SEATING IS LIMITED. REGISTER NOW FOR THE CITY NEAREST YOU! WWW.SYS-CON.COM

Learn How to Create, Test and Deploy Enterprise-Class Web Services Applications

TAUGHT BY THE INNOVATORS AND THOUGHT LEADERS IN WEB SERVICES

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASIC TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI AND XML, PLUS MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS AND REMOTE REFERENCES.

RECEIVE 4 FREE 1-YEAR SUBSCRIPTIONS ($345 VALUE!)

SHARPEN YOUR PROFESSIONAL SKILLS.

KEEP UP WITH THE TECHNOLOGY EVOLUTION!

"Presented an excellent overview of Web services. Great inside knowledge of both the new and old protocols. Great insight into the code piece."
– Rodrigo Frontecilla

"Very articulate on the Web services SOAP topic and well-prepared for many questions. I've learned a lot from this seminar and I appreciate this seminar for my job. Thank you!"
– Kenneth Unpingco, Southern Wine & Spirits of America

"I liked the overview of Web services and the use of specific tools to display ways to distribute Web services. Good for getting up to speed on the concepts."
– B. Ashton, Stopjetlag.com

Echoed over and over by Web Services Edge World Tour Attendees:

"Good balance of theory and demonstration."

"Excellent scope and depth for my background at this time. Use of examples was good."

"It was tailored toward my needs as a novice to SOAP Web services – and they explained everything."

WHO SHOULD ATTEND:

- **Architects**
- **Developers**
- **Programmers**
- **IS/IT Managers**
- **C-Level Executives**
- ***i*-Technology Professionals**

IF YOU MISSED THESE...

BOSTON, MA (Boston Marriott Newton) SOLD OUT!
WASHINGTON, DC (Tysons Corner Marriott) SOLD OUT!
NEW YORK, NY (Doubletree Guest Suites) SOLD OUT!
SAN FRANCISCO, CA (Marriott San Francisco) CLASSES ADDED SOLD OUT!

BE SURE NOT TO MISS THESE...

Each city will be sponsored by a leading Web services company

...COMING TO A CITY NEAR YOU

2002

NEW YORK AT WEBSERVICES EDGE 2002 EAST	JUNE 27
BOSTON	JULY 10
SAN FRANCISCO	AUGUST 6
SEATTLE	AUGUST 27
AUSTIN	SEPTEMBER 10
LOS ANGELES	SEPTEMBER 19
SAN JOSE AT WEBSERVICES EDGE 2002 WEST	OCTOBER 8
CHICAGO	OCTOBER 17
ATLANTA	OCTOBER 29
MINNEAPOLIS	NOVEMBER 7
NEW YORK	NOVEMBER 18
SAN FRANCISCO	DECEMBER 3

2003

CHARLOTTE	JANUARY 7
MIAMI	JANUARY 14
DALLAS	FEBRUARY 4
BALTIMORE	FEBRUARY 20
BOSTON	MARCH 11

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE LOWEST REGISTRATION FEE.

TOPICS HAVE INCLUDED:
Developing SOAP Web Services
Architecting J2EE Web Services

On May 13th, the San Francisco tutorial drew a record 601 registrations.

SYS-CON *i*-TECHNOLOGY EDUCATION

REGISTRATION FOR EACH CITY CLOSES THREE BUSINESS DAYS BEFORE EACH TUTORIAL DATE. DON'T DELAY. SEATING IS LIMITED.
NON-SUBSCRIBERS: REGISTER FOR $245 AND RECEIVE THREE FREE ONE-YEAR SUBSCRIPTIONS TO *WEB SERVICES JOURNAL*, *JAVA DEVELOPER'S JOURNAL*, AND *XML-JOURNAL*, PLUS YOUR CHOICE OF *BEA WEBLOGIC DEVELOPER'S JOURNAL* OR *WEBSPHERE DEVELOPER'S JOURNAL*, A $345 VALUE!

TO REGISTER: **www.sys-con.com or Call 201 802-3069**

Get More from Oracle with <CFQUERY>

Part 1 of 2

There's more to this tag than meets the eye

BY NEIL ROBERTS

Using <CFQUERY> opens up a whole range of Oracle functionality.

This includes calling your own functions, formatting data ready for your ColdFusion templates, and using Oracle bind variables to reduce the load on your Oracle database.

Lesson 1 in using Oracle with ColdFusion is that <CFINSERT> and <CFUPDATE> don't work – the database is accessed using either <CFSTOREDPROC> or <CFQUERY>, the last mentioned being the focus of this article. Hopefully I can provide insight into the wide range of Oracle functionality that can be drawn into your ColdFusion templates by effectively utilizing this one tag. Having opened with that rather profound statement, I'll immediately water it down by saying <CFQUERY> becomes a whole bunch better when used in conjunction with <CFQUERYPARAM>.

When I started at my current job I implemented <CFQUERY> across the board and saw performance improvements of up to 80%. It was like a dream. One thing to note when you try it yourself, however: the first time you execute the query shows no discernible difference in execution time whether or not you use the <CFQUERYPARAM> tag. Hang in there though, and execute the query again – you'll see a big difference.

This is because of the way Oracle executes a new query. It has to be parsed, the execution plan has to be determined, and then the query is executed. Oracle tries to avoid doing this over and over whenever possible, so it caches queries in the shared global area, or SGA, as it's known. If the same query is executed again, it doesn't have to repeat the whole process and therefore executes much faster.

For a real-world example look at the following queries. They retrieve profile information for visitors to a Web site (all driven by an ID stored in a cookie). As you might expect, it's likely that such queries will be run thousands of times a day. If you don't use <CFQUERYPARAM>, you'll have thousands of versions of what is effectively the same query, just using different IDs – all having to be parsed and validated, and all having to have execution plans determined. In short, you won't be maximizing your use of the SGA.

```
SELECT name, eye_color, shoe_size
FROM users
WHERE id = 12

SELECT name, eye_color, shoe_size
FROM users
WHERE id = 99
```

If you use <CFQUERYPARAM>, you capitalize on Oracle bind variables and a process called *late binding*. The query stored in the SGA looks something like the example below, where the :1 is a placeholder into which different parameters can be passed. As far as the SGA is concerned, it's running the same query over and over again, and each time it's executed it gets shifted to the top of the pile within the SGA. Hence it's much more likely that you'll have a cached copy to run. This is an effective use of the SGA, as the parsing and creation of the execution plan is carried out only once and hence is much faster.

```
SELECT name, eye_color, shoe_size
FROM table
WHERE id = :1
```

Another aspect to bear in mind is that Oracle's SGA is case sensitive and hence will regard

```
SELECT ID, NAME
FROM TABLE
```

as different from

```
select id, name
from table
```

The SGA won't recognize that they're the same, so you'll miss out on a performance gain.

Performance Drag

Carrying on with the performance theme, another possible drag is the lazy programmer – oh, yes, you know you're guilty – specifically:

```
SELECT *
FROM table
```

instead of listing the columns you need.

This isn't slower, I hear you protest, but it is. The reason lies in the way Oracle processes queries. If you use *, it has to go to the data dictionary and find what columns are in the table before the query can be executed. This adds another stage to the execution process, which naturally makes it slower. Also, why retrieve the columns back to ColdFusion if you're not going to use them? It increases the size of your ColdFusion page and the amount of network traffic. Even if you're running ColdFusion and Oracle on the same server, the data still needs to be transferred between applications, which can be a real performance lag.

On the same thread, why make more trips than you have to? By using the often overlooked <CFQUERY> attribute BLOCKFACTOR, you can prevent this too. What BLOCKFACTOR does is define how many records will be returned at once. The default is one, so if you have a query returning 50 records, that means 50 round-trips to Oracle. If you were to up the BLOCKFACTOR to 50, that's one round-trip to Oracle, which is faster, faster, faster, and that's what we like. Use this attribute sparingly, however, as it really works well only when you know how many rows are going to be returned. Setting this to the maximum 100 won't necessarily speed things up. In fact, you may even see performance degradation.

While I love speed, and endeavoring to tune my SQL gives me a warm feeling inside, there are a number of other neat things you can do with a <CFQUERY> tag. There are inserts, updates, and deletes, but my favorite by far is calling your own Oracle functions in your queries. It's fairly standard to be able to use Oracle functions such as COUNT, UPPER, and SUM in your queries, and you should use them where appropriate. After all, Oracle is very good at data manipulation; let's not forget that's what it's famous for!

I know that often ColdFusion has similar functions, but properly using Oracle will speed up your templates, and using Oracle functions to preformat data will reduce the template size, especially if you use a methodology like Fusebox, which promotes reuse of queries. You only have to do this formatting once in the query rather than many times in different display templates.

I digress. I was writing about calling your own Oracle functions. This was first introduced in Oracle 7 but has been simplified in Oracle 8 so all you have to do now is create your function and compile it successfully. Then you can call it something like this:

```
SELECT id, get_name(id, type) as
full_name
FROM table
WHERE id = 33
```

This means you can carry out your own formatting, processing, and general data manipulation and have it output as if it were a column in the database. So what happens? Well, you're just calling a function as you would in normal PL/SQL and the returning value is output into, in this instance, the variable full_name. The use of the keyword "as" assigns the result of the function call to the variable "full_name". From that point on the output can be treated like any other column output from a query. Beware, though, this method really works only with data retrieval, formatting, and the like. If the function you're calling does something like an insert, update, or delete, it won't work, and ColdFusion won't let you know. So keep a keen eye out.

What if you just want to call a function without its being part of a query? All is not lost. I know ColdFusion doesn't provide an explicit method for calling functions, as it does for procedures, but don't lose faith. It can be done, and very simply. This is what you do:

```
SELECT get_name(id) as v_cheese
FROM dual
```

It looks just like the last example, doesn't it? And it almost is, except this select uses Oracle's "dual" table, a sort of psuedotable that's created by default in every Oracle installation. Without belaboring the point, this little baby enables you to call functions, your own or standard Oracle functions, in a standalone manner.

As you can see, there are loads of things you can do inside a <CFQUERY> tag, and it doesn't stop here, either. You can concatenate columns with columns, and even with strings, thus:

```
SELECT 'my Name is ' || first-
name || lastname as full_name
FROM table
```

Or overcome those pesky date problems by outputting dates in Julian format for ordering, thus:

```
SELECT TO_CHAR(start_date, 'J')
as julian_date
FROM table
```

I've covered barely a fraction of the power available within Oracle, but hopefully I've pointed you in the right direction or hinted at a hidden gem or two. Moreover, I hope I've shown that, in general, the leap to Oracle from any other database isn't that great, and you can gain very impressive results with minimal effort.

@ NSROBERTS@HOTMAIL.COM

ABOUT THE AUTHOR

Neil Roberts is an analyst programmer with a large international law firm based in London. His focus is on developing a range of Web sites and Web-based applications using both ColdFusion and Oracle. He is also an active member of the UK ColdFusion user group.

ColdFusion Feature

By Kevin Towes

Building the Development Team in Flash MX and ColdFusion MX

Two cool technologies + Two types of developers

As a ColdFusion developer you're in a unique position to harness the power of ColdFusion in a rich end-user experience. Don't be afraid of Flash. Once you have the hang of it, it's a fun environment to work in, and the results are always exciting – even inspiring sometimes.

The possibilities of Flash are as overwhelming as the authoring environment. But this doesn't excuse your responsibility for good development methodologies employing collaborative teams while maintaining code integrity. ColdFusion developers have a knack for trying new things and new techniques – that's what makes us stand out from the crowd.

As coders you are not and should not be required to become high-end animators or Flash designers. You should, however, have a basic understanding of how Flash works and what it can do. As Macromedia ColdFusion developers you're in a unique position, especially with the introduction of Macromedia Flash Remoting. ColdFusion MX has found a virtual sail (Flash) that puts it (and you) in an awesome place and time. CFMX is the platform of choice for exposing Flash applications with server-side controls and data management. There's a hunger and demand in the Flash world to extend Flash with the power and services offered by ColdFusion. The ColdFusion developer can also offer unique insights into code management and team deployment that's tough to find in Flash circles. CF developers will take Flash to a world and market it has never been to before, such as data management and delivery.

The biggest question right now is how to efficiently deploy your team's resources on Flash-based projects. Everyone's biggest fear is the dependency on that one person who "knows" Flash, ColdFusion, databases, design, and animation. This is not only dangerous, but an insane amount of pressure on a single person.

Think of this article as a primer on Flash. Its focus is on some techniques we've used to apply a strategy for collaborative development cycles within Flash. I hope it invokes thoughts and ideas concerning a new open development standard for Flash applications. As a familiar reference, I'll examine similarities to the Fusebox open standard (www.Fusebox.org). You already know how to prepare data to be loaded, so I'll focus on building your workflow environment.

Context: Why Flash?

Let's build the case for even considering Flash. You may think the first thing I'll say is the Macromedia hype: "You can build rich applications using Flash and ColdFusion." What does that mean? You can have animation? Tell me, how many programmers care about animation? What Macromedia is really saying here is that Flash provides an authoring environment for rapidly developing applications used to manage data efficiently. The biggest benefit to you, the developer who's often responsible for developing the user form controls, is that you no longer need to be concerned about issues of platform, OS, browser, or even device. The user interface is processed identically across everything. This alone will save you time previously spent in debugging complex JavaScript, DHTML, and HTML scripts.

I'll go out on a limb and say that a Flash-based data management interface is not only more comprehensive for your end user, but will be faster in development and deployment. So here's a thought. Flash is as much a rapid development client-side technology as ColdFusion is a rapid development platform for server-side technology. It's really the second piece in the puzzle that will have a dramatic effect on the development world as we now know it.

Let's start by identifying your team resources so we can understand the roles each person can and should play. Following that, I'll introduce the server-side coder to the uniqueness of the Flash MX authoring environment. We'll look briefly at the Flash UI components and server connection strategies. After that we'll deconstruct your Flash movie into separate movies that allow more than one person to contribute to the development. To finish off, we'll review variable and object strategies. This will be a top-level discussion, and we won't go into too much detail on the guts of the technologies.

Build the Team

Before beginning any project, a full understanding of the project's scope must be the first item on your agenda. The second should be to review the creative and technical resources available to you. Many projects fail or aren't completed on time due to inadequate understanding of the full scope of a project and its technical details.

Assuming your project is under control at the top level, develop a strategy to allow a number of people to work on it at the same time. It's obvious that we need to separate the application into many files to leverage your team's individual skill sets and existing code bits, objects, or components built in other projects. *Note:* Flash (FLA) files can't be opened by more than one developer at a time.

The "best world" dream team will consist of 10 roles. I say *roles* because in many cases people take on multiple roles, and in most cases development companies don't have 10 people to put on a single project.

The production team is divided to satisfy two key areas: creative and technical. The *creative group*, led by the creative director, works the art, design, and animation. The *technical group*, led by the chief architect, works the data requirements and server-side processing requirements.

The management team will consist of:

- ***Project manager*** (production leader)
- ***Creative director***
- ***Chief architect***

The development team will consist of:

- ***Flash constructor*** (interface between creative, technical, QA teams)
- ***Interface designer*** (responsible for user interface design and strategy)
- ***Animator*** (responsible for screen transitions, animated art)
- ***Flash coder*** (responsible for objects, functions, server-side communication, internal data management)
- ***Server-side coder*** (responsible for preparing data for transport via XML, Flash Remoting, or name/value pairs)
- ***Database administrator*** (responsible for the database model, integrity and permissions strategy)

The QA team assumes the tenth role in the production. They will ensure that animations work and data is accessible, and that the Flash movie is quick and stable across required platforms and devices.

The layout of the team structure in Figure 1 omits one team, the *content team* – not because they aren't important, but content is content, and the role should be no different in Flash than in ColdFusion/HTML.

Splitting the Movie

Before you can apply the team strategy just detailed, you need a file management strategy, as defined by the Flash constructor. This person will know the scope of the application and the team's strengths and skills. If you're familiar with development standards such as Fusebox, you understand that particular code objects are separated into different files. Business logic and database queries are separated from display logic to allow for maximum reuse of code and to reduce debugging time. When you're building Flash applications that are required to interact with a server, you have to practice these methodologies – you don't really have a choice.

A side benefit of this approach is that your visitor doesn't need to load the entire Flash application at once. This affects download time! A second benefit is that you can track a user's activities in Flash. Once a user has loaded a Flash movie, there are no callbacks to the server and therefore no activity logging. The more you call back to the server, the more impressions you make in the server log files, which you can analyze using standard third-party tools – another side benefit.

How do you determine where to "split" the Flash movie? One approach is to review pages that are similar, mostly in layout and content. Splitting your movie by section is another approach. This way you can assign a team member to work on the "portfolio section" or the "company info" section (explained in detail later).

Another thing to consider is that Flash has the ability to share libraries across multiple external movies. The Flash library is a collection of the symbols (graphics, buttons, fonts, movie clips, UI components) you're using in your movie. If you separate the Flash movie, this feature helps you keep everything in one place and under some control, as well as lightens the download requirements for the user. We won't go into the guts of shared libraries here, but it's important to be aware of it.

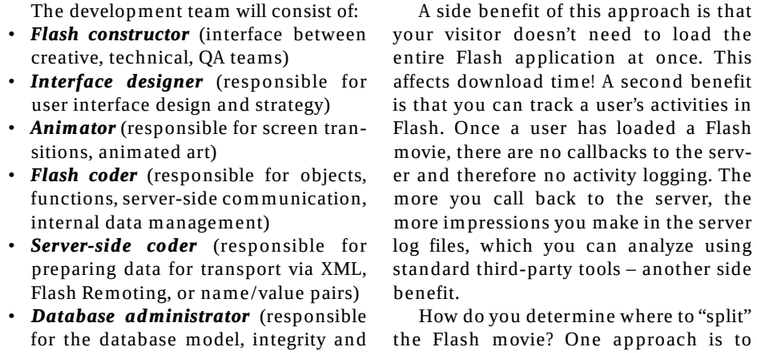


FIGURE 1: Team structure

Assembling the Movie

In ColdFusion CFML scripts are "assembled" at runtime using tags like CFINCLUDE and CFMODULE. The Flash ActionScript equivalents are loadMovie and #include. These two methods assemble external templates and scripts in a single movie. This process promotes proper code reuse.

The only similarity between #include and loadMovie is that they import external files. Other than that, they're very different. The loadMovie assembly occurs at runtime in the Flash Player. The #include method includes ActionScript saved in text files (.as) – not at runtime, but when the movie is published inside Flash MX – so the .as files must be available to Flash MX locally.

The loadMovie function requires two parameters: the SWF filename with the relative location, and where to place it.

The syntax for loadMovie is:

```
loadMovie (URL [,target/level [,
"get"/"post"]])
```

Two options are available to place the SWF file: target and level. A *target* is the Movie Clip or object that must exist before the ActionScript is executed. A *level* is much like an overlay or underlay for the entire template on top of the caller SWF.

Each level or target can contain only one movie. The benefit of targets over levels is that, using ActionScript, you have more control over them. For example, you can set a Movie Clip's visibility property. The method most practiced when loading a movie is to put it in a target clip. There is no "official" Macromedia recommendation to use levels or targets.

With loadMovie the SWF files must be accessible by the Flash Player, or requested within the same server domain (e.g., www.myDomain.com). The loadMovie ActionScript can exist anywhere ActionScript can be placed – in a frame or event handler, or from a clip, button, function, or Flash UI component.

Loading an SWF Movie into a Movie Clip

An external SWF file loaded into a movie clip will inherit all clip parameters and properties at the time of load. This includes position, size, rotation, and instance name. Variables and objects existing before the SWF movie is loaded will be lost.

Once an SWF has been loaded, ActionScript can control the properties of the clip containing the new SWF file. For example, to control the timeline of the loaded file, access it by its Movie Clip name.

To move the play head of the loaded movie named "myMovieClip" we would script:

```
_root.myMovieClip.gotoAndPlay(10);
```

In ActionScript you have much more control over a Movie Clip than a level. For example, you can adjust a Movie Clip's visibility or alpha:

```
root.myMovieClip._visibility=true/
false;
```

Loading an SWF Movie into a Level

The external SWF file that's loaded into a level will keep its own localized application state. Unlike loading into a Movie Clip, a target Movie Clip isn't required. When you load an SWF file into a level, it will either overlay or underlay the caller file. Any transparent sections of the loaded file will reveal the file(s) below. Similar to Movie Clips, when a movie is loaded into a level, it will "replace" all objects and variables in that level.

Conceptually, you're stacking files (see Figure 2). The loaded file adopts the stage size of the file on _level0, and objects on the stage beyond it are cropped out. Unlike Movie Clips, levels have limited properties and can't be manipulated to the extent Movie Clips can. To control the timeline of the loaded file, access it first by its level, then by its root. For example:

```
_level1_root.gotoAndPlay(10);
```

Unloading a File

Movies that were loaded can be removed (unloaded). Unloading a movie deletes all objects and variables, and stops all timelines within that file. The syntax for unloading a movie is:

```
unloadMovieNum ([target/level])
```

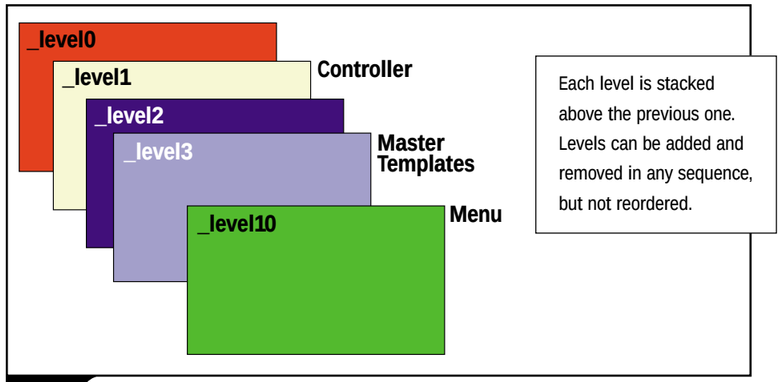


FIGURE 2: Stacked files

Establishing the Structure of Your Application

Figure 3 offers a suggestion for breaking out your application. As you can see, the application is sectioned similarly to an HTML- or CFML-based Web site. Each page (or screen) is within its own Flash movie. The Core file (FlashCore.swf) essentially loads the pages into it when requested. In this example it's doing all the data loading from XML documents generated by ColdFusion. As you already know, your options for loading data include LoadVariables, LoadVars, XML, and Flash Remoting.

This particular Flash application will have few calls to the ColdFusion server for data. This is the traditional approach to Flash development. If you plan to use Macromedia Flash Remoting technology, your application model will look a little different – it's easier and more efficient to make frequent calls to the server. For more on Flash Remoting visit www.macromedia.com/software/flash/flashremoting/.

The .as files (in yellow) are a collection of three classes:

- ***NetServices:*** Flash Remoting class required when accessing the NetServices functions (Flash Remoting)
- ***MyFunctionLibrary.as:*** Collection of ActionScript functions recycled from other projects (code reuse)
- ***FlashMovieControl.as:*** Series of project-unique functions required to run application

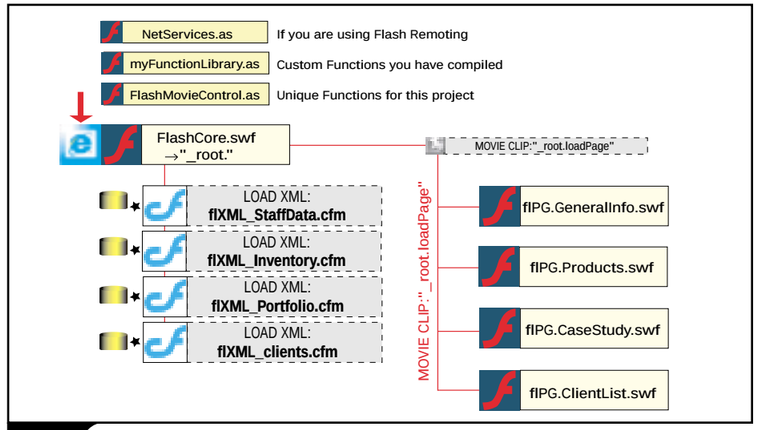


FIGURE 3: Suggested application structure

Filenames

Sanity is important for all coders, and in the interest of good organization the practice of consistent and contextual file naming will have a positive effect on your development environment. We've been successful with the expression flPG.GeneralInfo.fla as syntax, where flPG represents the type of movie this is. "GeneralInfo" is the contextual title for the contents of the file.

Other naming syntax we employ: flGet, flPost, flXML. For example, flGet.MyDataCall.cfm is accessed in ActionScript as:

```
ServerData = new LoadVars();
ServerData.load("flGet.MyDataCall.cfm");
```

flPost.MyServerAction.cfm is accessed in ActionScript as:

```
ServerData = new LoadVars();
ServerData.sendAndLoad("flPost.MyServer
  Action.cfm", this, "post");
```

flXML.MyXMLData.cfm is accessed in ActionScript as:

```
ServerData = new XML();
ServerData.load("flXML.
  MyXMLData.cfm", this, "post");
```

Keeping the filenames of your ColdFusion templates to a standard will help you and your team recognize where particular methods and functions are happening. The filename is also self-documenting.

Separating the ActionScripts

ActionScripts play a critical part in the success of an application. To get the most out of Flash, consider wrapping your code into ActionScript functions or go one step further and start writing your own classes with their own properties and methods. That's right – Flash isn't just animation. ActionScript, if you didn't know, is based on the European Computer Manufacturers Association (ECMA-262 at www.ecma.ch), and serves as the international standard for the JavaScript language.

Thus, if you've spent all your time developing these classes with properties and methods, you don't want to have them corrupted during the other phases of production. Securing the ActionScript logic inside Flash is important. When animators, designers, and junior developers work on the file containing your scripts, you need to guard against accidental deletion or corruption. You also need to provide the instructions for your development team to deploy these objects and methods you create. The best way is to physically remove the scripts and include them when the movie is published.

The function #include filename will call the external file (.as) and include it inline in the ActionScript.

The included file is inserted inline when the FLA file is published to SWF. Custom functions, objects, and complex ActionScript should be pulled out from the FLA file and included to secure the code. Also, if there's a problem, the coder could adjust the code without opening Flash! Yes, you can edit ActionScript in ColdFusion Studio. If you include .as extensions in the JavaScript color code setting, your ActionScript will turn into something a little more recognizable. Remember, ActionScript = JavaScript! This is a great technique to stockpile code snips for reuse in other projects or to exchange with other developers.

A Note About Movie Clips and Objects

To a person interested in animation and design, a Movie Clip is something that can contain a separate timeline, ActionScript, animation, anything. To coders, the Movie Clip is an instance of the Movie Clip object, which Macromedia has developed with a series of parameters and methods. That's what allows you to control it using ActionScript. With that understood, it's not a reach when I say that variables inside a Movie Clip are local only to that Movie Clip instance. So a Movie Clip is an instance of the Movie Clip object, or class; variable structures can be created inside that object.

You can create variable structures in ActionScript by creating a new Object(). For example:

```
MyStructure= new Object();
MyStructure.MyVar1 = Kevin;
MyStructure.MyVar2 = Towes;
```

This example created a new object instance, but it doesn't contain any visuals; it's used primarily as a container for data. Flash MX introduced the LoadVars() object. This new object, or class, has a number of functions that allow for a much more efficient way to connect with the ColdFusion server. It essentially replaces the LoadVariables method, which wasn't an object and had no additional functionality. LoadVars() has methods that allow you to monitor the load and control the data flow between server and client. Using LoadVariables, that option was available only to advanced developers.

Okay, let's get back on track here.

Accessing Data

Much like the JavaScript DOM, you can access any variable or object from

anywhere in any timeline, including external movies on different levels. The dot notation is simple: "_root" represents the root timeline on any particular level; ".movie ClipName" represents the instance name of any Movie Clip (or object) on the root timeline. We can prefix the _root object with _level1 to reference a _root timeline on a different level (an external SWF file that's been loaded).

_level1_root.myMovieClip.myVariable would reference a local variable called "myVariable" on a Movie Clip instance named "myMovieClip" sitting on the root timeline of an external movie loaded into level1. Does this look similar to a variable structure to you?

Back to the Methodology

Poor variable management is something we've all been guilty of at one point or another. Rapid development does that. Quick fixes are due in part to time pressures and lack of planning. The results are redundant variables; poor use of arrays, structures, looping; database calls; and so on. With Flash you can get really out of control. When complex data enters the picture, a clear variable and scripting strategy will make your job much easier and the team working on the project will love you for it.

Now that we have multiple files, how do you control and manage the data? Flash is an environment in which the user state is persistent. This is a radically different concept from ColdFusion development, which emulates state using persistent cookies and server-level variables. In any traditional Web application the "state" of your application is lost as soon as it's returned to the client. In applications such as ColdFusion, state is maintained virtually, by matching variables with a value stored in a client cookie.

In Flash, once a variable is set (internally or through a server request), it exists until it is overwritten or deleted, or until the object instance (MovieClip) is no longer available (unloaded). If used correctly, this becomes very powerful and will reduce the number of hits you need to make back to the server.

In an application each section (General Info, Products, Case Study, Client List, etc.) is developed as a separate file and loaded into a Movie Clip on an event (onMouseClick, onEnter Frame, etc.). When a new section is required, a new movie replaces the SWF file. All variables and objects that were set are also removed.

If you have a large database call in the recently unloaded file, another server request is required to access your data. A simple strategy might be to store all variables in a common place (timeline) that is persistent. Any level or Movie Clip that won't be unloaded will do – just be consistent.

Establish a clear strategy for variable management. Use data objects to store variable structures, and avoid setting _root variables. This way your team will know where to access the data, thus increasing productivity and chances for success.

Here are some examples:

```
/* establish objects in your
  core movie (level0) that will
  manage data*/
_root.ServerData = new
  Object();
_root.ServerData.GeneralInfo =
  new LoadVars();
_root.ServerData.Products = new
  XML();
_root.ServerData.CaseStudy =
  new Object();
```

_root.ServerData.GeneralInfo will contain all data retrieved from the ColdFusion server for the General Info section. When the Movie Clip, flPG.GeneralInfo.swf, is accessed, the data will already be available. GeneralInfo is a LoadVars() object, and can access any object method available, including .load and .sendAndLoad.

Let's assume that your ColdFusion file returned an ActionScript name and value pair string like this one:

```
WelcomeText=Welcome to the
  Pangaea NewMedia
web site&CorporateEmail=ktowes
  @pangaeaNewMedia.ca&WebURL
  =www.PangaeanewMedia.ca
```

Essentially, you now have three variables available in your ServerData.GeneralInfo object. To access that data:

```
ServerData.GeneralInfo.WelcomeText;
ServerData.GeneralInfo.CorporateEmail
  ;
ServerData.GeneralInfo.WebURL;
```

_root.ServerData.Products will contain XML data returned from the ColdFusion server or straight XML document. Like the .GeneralInfo object above, the .Products object will have access to all the XML methods defined in Flash's XML object.

_root.ServerData.CaseStudy will contain a structured record set returned from a ColdFusion MX Web service call. At this point CaseStudy is set as a straight object. It will be populated with a record set array when the service function has returned data.

The Rest of the Project

While you've been developing the data model, and possibly the ColdFusion resources, the rest of the team has been working on user interface design, animation concepts, and engineering sound, each independent of the ColdFusion developer and/or the Flash coder. This was possible because you had a methodology and a development strategy.

Remember that each team is responsible for a specific part of the project. Animation is as important as interface design, which is as important as data management.

I like to test a production process against the goals of FuseBox. If your production cycle can meet the following criteria, you have an efficient and productive development environment. Thanks to Hal Helms for supplying me with this list:

- Code is separated (Queries, Display, Actions).
- Code becomes much more abstract and easier to reuse.
- The learning curve for new developers is short.
- The methodology standard is a collective effort.
- Tools and training can be leveraged across the entire development team.
- Standardized documentation assists in the maintenance of existing code.
- Project risk is reduced by reducing the time the project is in the dark.
- Stress and burnout are reduced on developers and managers alike…

Conclusion

There's so much to cover when you discuss the marriage of two very cool technologies and two types of developers. Flash is a very sexy approach to building rich user interfaces, and the ColdFusion developer has so much to offer the technology and the process. ColdFusion, as Ben Forta preaches, is the glue that binds Flash with Web services and data objects.

We've reviewed how to apply programming principles to enable your development team to collectively develop powerful Flash applications. Separating your FLA file into external files will allow you to incorporate a large team to develop awesome technical and creative solutions. Separating ActionScript code will secure your scripts and promote code reuse. Maintaining a consistent strategy for Flash development is the only way to move forward with the marriage of Flash and ColdFusion.

About the Author

Kevin Towes, cofounder of Pangaea NewMedia (www.PangaeaNewMedia.ca) in Toronto and a certified ColdFusion developer, holds advisory board appointments at the Canadian Film Centre and Macromedia. Manager of one of the world's largest ColdFusion user groups, Kevin is the organizational lead behind Toronto's annual CFNorth Developers Conference. A frequent speaker and author of numerous articles on ColdFusion and Flash, Kevin contributed to the upcoming book Inside Flash MX *(New Riders).*

@ KTOWES@PANGAEANEWMEDIA.CA

Ask the **Training Staff**

A source for your CF-related questions

BY
BRUCE VAN HORN

Wow! Can you believe it's June already? The last few months have been blurred by the flurry of excitement around the new MX products announced and released by Macromedia.

As you'll see from the first question, I'm very excited about the new CFMX product. I know you'll like it once you give it a spin. I'd like to hear your comments and questions about it.

Q: *I've been hearing a lot of talk about the new CFMX. Have you had a chance to play with it? If so, what do you like about it? And do you recommend upgrading?*

A: There has been a tremendous amount of talk about CFMX and all the new MX products from Macromedia. At the time of this writing the preview editions are available for downloading – they're very exciting. I've been participating in the beta program and have gotten to test-drive most of the new features. Bottom line: I love it! If you don't plan to use any of the new code features, you should still upgrade. Your existing code will run much faster than it does on ColdFusion 5.0. I ran a simple test on a server running both CF5 and CFMX. The same code took an average of 321ms to run on CF5 but only 62ms on CFMX. Definitely get a copy and try the new features and speed for yourself.

Q: *Is there a limit to the number of scheduled tasks we can run on a CF server? Currently we have nearly 20 different scheduled tasks and are trying to figure out whether we should get a separate server dedicated only to our scheduled tasks.*

A: No, there's no limit. CF uses very little overhead to check whether a scheduled task should be executed. The scheduler simply initiates an HTTP request for the specified page, just as a real user would. If your system can handle 20 simultaneous users, it can handle 20 scheduled tasks. Your application server, however, may feel the strain if all of your tasks are scheduled to execute at the same time. To spread out the server load, try to avoid having all of your tasks execute at the same time.

Q: *I'm trying to build an application that emulates the way a human interacts with a page, and I need to create pauses between page selections. How can I make CF wait 10 seconds before processing the next section of code?*

A: There's a right way and a wrong way to do this. The easiest thing is to create a conditional loop that simply loops until 10 seconds has passed (see listing below for an example). This isn't the best method, however, because it's very resource intensive. Essentially, every time this code runs, it will use 100% of your CPU resources. If the wait isn't long and there aren't many users on your server, this may be okay, but I don't recommend it. For the best solution go to Macromedia's Developer Exchange (http://devex.macromedia.com/developer/gallery/) and download either CFX_Sleep or CFX_Wait. Both of these custom tags do the same thing and use very few system resources.

```
<!--- set a variable equal to
the now() function --->
<cfset StartWait = Now()>
<!--- create a conditional loop
that evaluates the
difference in seconds between
now() and the value
set in the StartWait variable--->
<cfloop condition="DateDiff('s',
Variables.StartWait, now()) lt
10"/>
```

Q: *I have a problem with clearing session variables. I use the automatically generated Session.URLToken variable on all my URLs so I don't have to use cookies for session management. When a user logs out, I clear the session scope using StructClear(Session). This, however, deletes CFID, CFTOKEN, and URLTOKEN. Is there an easier way to clear out a user's session variables without deleting these built-in variables?*

A: Yes, you have three options here. First, you could create a new structure in the session scope, like Session.UserData (see listing below), and add all of the variables you were storing directly in the session scope to this new structure. If you're using a variable like Session.LoggedIn to check a user's login status, you'd simply change your code to check for Session.UserData.LoggedIn. When your user logs out, simply delete the UserData structure from the session scope using StructDelete(Session, "UserData"). This erases the data associated with that user without deleting the other session variables the CF created automatically. This is probably more code than you want to write, however, since you have to change every reference from Session.LoggedIn to Session.UserData.LoggedIn.

```
<!--- Where you normally ini-
tialize your session
variables, create a new con-
tainer called UserData --->
<cfset Session.UserData =
StructNew()>

<!--- Then add your normal
session vars to this new
UserData container --->
<cfset Session.UserData.LoggedIn = 1>
```


> **"You should upgrade to CFMX. Your existing code will run much faster"**

```
<!--- At logout, delete the
UserData container --->
<cfset tmp =
StructDelete(Session,"UserData")>
```

Perhaps a better way is to simply change the code you run at logout. Instead of clearing out the entire session scope using StructClear(Session) – which is certainly short and easy – you could loop over your session variables to delete all but the ones CF created (see listing below). While this is more code than you currently have in your logout process, it's less than you need to modify all your session variable references as outlined above.

```
<!--- Code to run when user
logs out of application --->
<!--- Create a list of CF-
Generated Session Variables --->
<cfset CF_Variables =
"CFID,CFTOKEN,SESSIONID,URLTO-
KEN">

<!--- Loop over the Session
scope --->
<cfloop collection="#Session#"
item="VarName">
 <!--- if the session variable
is NOT in the CF_Variables list,
delete it --->
 <cfif not
ListFindNoCase(CF_Variables,
VarName)>
  <cfset tmp =
StructDelete(Session,VarName)>
 </cfif>
</cfloop>
```

For my money, however, there's an easier method. If your login process always reinitializes the session variables for each user, it's usually sufficient to delete only the one variable you're using for login validation. For example, if you check for Session.LoggedIn in your application pages, all you have to do is delete that one variable. Your other session variables will be overwritten or erased when the user logs in again, and Session.URL Token will remain intact. Instead of StructClear(Session), simply use StructDelete(Session,"LoggedIn").

• • •

Please send your questions about ColdFusion (CFML, CF Server, or CF Studio) to AskCFDJ@sys-con.com. And please visit our archive site at www.NetsiteDynamics.com/AskCFDJ.

ABOUT THE AUTHOR
Bruce Van Horn, president of Netsite Dynamics, LLC, is a certified Macromedia developer/instructor and a member of the CFDJ International Advisory Board.

@ BRUCE@NETSITEDYNAMICS.COM

New Possibility in CFMX: **Server-Side Redirects**

Doors are opening that may offer interesting rewards

BY CHARLES AREHART

Now that we can talk freely about ColdFusion MX, I want to share something new that you likely won't hear much about...

...but that I think stands to be an important feature: server-side redirects, or "forwards." It's something CF has long been missing, and an improvement over CFLOCATION. The thing about a forward is that it's quite different from, and in most ways much better than, a CFLOCATION. We've all used CFLOCATION to transfer control from one CF page to another. But did you know that it performs a client-side redirection? The user doesn't see it happen, and maybe you never noticed it. It's done by CF sending a special HTTP header to the browser. Maybe it never mattered to you, but this behavior has some negative implications.

The most important one is that data that's passed into the calling page isn't available to the called page. In other words, if template1.cfm is called as a form action page and is passed several form variables, when you do some processing and then use CFLOCATION to transfer to template2.cfm, you won't have access to those form variables in template2.cfm (see Figure 1).

Many have tried to get around this challenge by passing the form variables on the query string in the CFLOCATION's URL, or by setting the variables into the session scope. Still another approach is to CFINCLUDE template2.cfm instead of using CFLOCATION to call it, or perhaps call it as a custom tag. Each has its place. But there may be times when for modularity, encapsulation, or other purposes, you do indeed want to transfer control to the next program, but also pass along all the data that was sent to the caller. Further, you may want to pass along any data created in the calling page (template1.cfm in our example), such as queries that were executed or the result of any other operations (such as a CFHTTP, CFLDAP, etc.). Again, this data won't be made available to the called page with a CFLOCATION. But it can be made available in a server-side redirect.

The Power of Server-Side Redirects

Server-side redirects solve this problem. Data in the calling page can be made available to the called page. ASP added this feature with their server.transfer() method (whereas their response.redirect() acted like our CFLOCATION). And servlets and JSPs have long had the concept of a "forward."

Indeed, because CFMX is based on an underlying servlet/JSP engine, we now have exposed to us the same capability. To do it, we simply leverage a new function in CFMX, which is in fact exposing to us the servlet PageContext that allows servlet and JSP developers to use a forward. In fact, the syntax to do so may look a lot alike, but it's all CFML:

```
<CFSCRIPT>

getPageContext().forward("relativeURL");
</CFSCRIPT>
```

This doesn't have to be done in a CFSCRIPT, but it works a little more easily this way. Still, for those who don't want to remember all this detail, I've created a simple custom tag called CF_FORWARD that can often be used in place of CFLOCATION. You can find it in the Macromedia Developer Exchange, at http://devex.macromedia.com/developer/gallery/info.cfm?ID=C0DB78BB-617C-11D6-840300508B94F380&method=Full.

An example of using it might be:

```
<CF_FORWARD URL="relativeURL">
```

Note that in both examples the relative URL can be any CF template (and even a JSP template if it's been placed within the CFMX directories where your CF templates are located). Note also that you can pass a query string on the URL, just as you can in CFLOCATION.

Sharing Data Between Pages

I said earlier that one of the key new features is that you can share data between the calling and called pages. The only trick is that the variables to be shared must be placed in the REQUEST scope. This scope has been available to us in CF for a while and was used previously for sharing data with custom tags.

Now, if you place some variables into the request scope, or perhaps create a query on the calling page with NAME="request.queryname", then those same REQUEST variables will be available (using the same REQUEST scope and variable names) in the called page. In fact, one feature I added to my custom tag is that if you use the LOCALVARSCOPIED="yes" attribute, it will automatically copy local variables (such as queries created on the page with no specific scope or the VARIABLES scope) into the REQUEST scope before calling the forward.

Note, too, that this means that CF and JSPs/servlets can also transfer control between each other and access each other's REQUEST scopes (although the REQUEST scope in JSPs and servlets means a bit more than it does in CFMX).

This ability to do server-side redirects is also the key to doing true Model-View-Controller (MVC) style design of your applications. That's a subject beyond the scope of this article, but for those who understand it, it opens very interesting doors to us. I hope more people in the CF world will investigate this possibility now that it's closer to a reality.

Not Perfect

Sadly, the forward method as currently implemented (in the preview release candidate, or RC) isn't perfect for a couple of reasons.

First, and perhaps challenging to those with servlet/JSP experience, I've found that unlike JSPs and servlets (for those familiar with them), in CFMX the REQUEST scope of the caller doesn't automatically include the FORM and URL variables that might have been passed to it. The good news, however, is that any URL variables available on the calling page are also available on the called page. In this respect it works like custom tags.

The second problem is more serious. I've found (again, in the RC) that if I try to do a forward on a page (or within the custom tag called from a page) that is itself a form action page (has "form." variables available), the forward fails with an "err.io.short_read" error. That's strange and a real bummer. Until that's resolved, you won't be able to forward from a form action page. I've got the custom tag detecting and stopping before that error.

This is real tragic, though, for using MVC style design for form action pages. I hope it's either fixed in a later RC or can be fixed before the final production release.

A Bright Spot

Despite the bad news, there's a piece of really good news. Many have surely been hassled by the fact that if you set a cookie before doing a CFLOCATION, it never gets set on the client. But I've found that if you set a cookie before doing a forward, it is indeed set to the client. It seems like magic, but JRun's documentation suggests that the JSP:forward tag has always had this behavior. So it's just another benefit we leverage since CF is based on the underlying JSP/servlet engine.

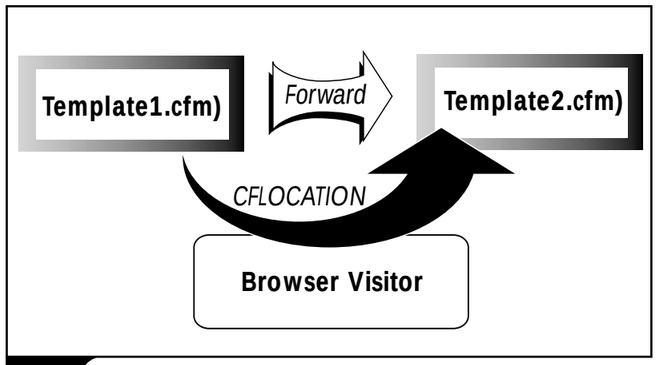


FIGURE 1: Client-side vs server-side redirection

Other Considerations

Finally, there are some things to be aware of with using a forward. When you do a server-side redirect, since by design it doesn't send a redirect through the browser, the URL shown to the user will still be that of the page they requested that did the forward, not the page to which control was redirected. In our example above, the URL will show template1.cfm. This has several ramifications (challenges if they bookmark it, possibly unexpected results for you or them if they hit the refresh button).

These are issues well discussed in the literature regarding the use of forward in JSPs and servlets, and I imagine ASPs since the introduction of server.transfer(). Over time, we'll all come to learn the same lessons and find the strengths and weaknesses of this approach.

It may also be interesting to see if the Fusebox community adopts this new approach in their design, given that it already attempts the kind of encapsulation and decoupling that is a hallmark of MVC design.

Of course, all this shouldn't challenge the Macromedia mantra that in CFMX one need not learn Java. No, one need not. I'm just pointing out that doors are opening that may offer interesting rewards for those who explore them – and perhaps eventually those discoveries may trickle into more abstractions for the CFMX masses.

CAREHART@SYSTEMANAGE.COM

ABOUT THE AUTHOR

Charlie Arehart is a certified Macromedia trainer/developer and CTO of SysteManage. He contributes to several CF resources, is a frequent speaker at user groups throughout the country, and provides training, coaching, and consultation services. Charlie is also a columnist for Java Developer's Journal.

By Dennis Baldwin

Flash Up Your Forms With Components

The future is bright for Flash/CF integration

FLASH MX HAS HIT THE STREETS, and some amazing applications are already under development.

Macromedia has gone above and beyond by providing us with this fully functional Web development tool. Now with drag-and-drop components such as scrollbars, list boxes, calendars, and much more, developers can harness the power of reusable code. This will dramatically enhance the development process as well as cut down the time it takes to maintain and debug Flash applications.

No longer "just an animation tool," as it has been labeled in the past, the new Flash has given rise to more full-blown Web applications that utilize Flash as the front end with a powerful application server like ColdFusion as the back end. Long gone (I'm hoping) are the days of useless intros, so let's make a conscious effort to start pushing Flash MX application development to its limits!

In Flash5 we were introduced to an incredible scripting environment that just keeps getting better. ActionScript allows the developer to harness the power of modular, object-oriented code that can control all aspects of a Flash movie. Reusable code is a staple in today's development projects and will save time and frustration in the long run. We'll take a look at a Flash application that utilizes components, gathers user input, and passes this information to be handled by CF. Let's get started.

Example Application

The purpose of this application is to become familiar with some new Flash MX components along with a few of their methods. Not only that, we'll see how simple it is to send information back and forth between Flash and CF. Since forms are used on practically every Web site, this will be a good example of how components can be used to enhance development. Flash MX has several built-in components that we'll get to know a little better.

The Macromedia development team has provided us with some handy form components that mimic HTML-based form elements. You'll need to download and install the Flash UI Components Set 2 from the Macromedia Exchange (http://dynamic.macromedia.com/bin/MM/exchange/main.jsp?product=flash). Installing these components requires the Extension Manager, which can also be downloaded from the exchange.

The Flash form will gather information from the user and insert it into the database. The key to handling information provided by the user resides in the ActionScript code. Almost every event has a corresponding method, or handler, that gathers the data and places it into a data object, which is then sent to CF upon submission.

Digging Into the Code

Open "form_components.fla" (this and other files referenced in this article are downloadable at www.flashcfm.com/cfdj/form_components.zip) and you'll see the code in frame one of the AS layer. This is all the code necessary to run the application. The components were dragged and dropped onto the stage and assigned the corresponding methods listed in the AS code. The init method is the key to kick-starting the application by setting focus to the "first_name" field, creating the data objects, initializing a few variables, and loading the states list from CF (see Listing 1).

The data object is simply a container that will gather data from the user and send it to the server once the form is submitted. The receive object will accept a response from the server and display whether the transfer was successful or not. The LoadVars object, new to Flash MX, adds increased functionality when sending and loading data, similar to the XML object. As with loading the states list from CF, an event handler that will fire once the data has finished loading can be specified. In our application the statesLoaded method is called and the data is handled accordingly.

```
function statesLoaded(success) {
 if(success) {
  var states = this.state.split(",");
  state_mc.setDataProvider(states);
 } else {
  trace("error loading data");
 }
}
```

If the load is successful, the states are split into an array and used to populate the combo box component, (see Figure 1), which has been assigned an instance name of "state_mc". The setDataProvider method is available to all combo box components and provides an efficient way of populating the combo box with data.

> To learn more about the methods of Flash MX components, be sure to view the ActionScript reference panel (Window > Reference). This panel comes in very handy when using components and learning how to interact with them.

The CF code in get_states.cfm is a simple database query that pulls the states from a table and places them in a comma-delimited list. Be sure to take a look at the states table in the users.mdb database. If a problem occurs when loading the states from CF, an error message is output to the debug window. It's a good idea to use a fully qualified URL, set in the "base_href" variable, so the CF data can be parsed and loaded into Flash within the IDE. If a relative URL is used, your CF code will be loaded into Flash and won't be very helpful for testing and debugging your application. Otherwise you'll have to publish and test your movie through the browser every time, which can be an extremely monotonous process.

The application has been initialized – what next? We need to start gathering data from the user and building our data object to send to CF. The input text fields such as first_name, last_name, city, and e-mail simply sit on the stage and wait for user input. These field values will be set in our data object once the send button is pressed; we'll look into this shortly.

The calendar component is a powerful one and in our application serves as a means to let users specify their birth date. The calendar carries an instance name of birthDate_mc and is assigned a change handler of getBirthDate. Each time a selection is made on the calendar the getBirthDate handler is called (see Listing 2).

When getBirthDate is called, we grab the date value using the getSelectedItem method. This value is set in our "birth_date" variable, and we use a little ActionScript to take care of the rest. Once we have the selected date, we need to format it in a manner compatible with CF. To do this we simply take advantage of the Flash Date object and create a new date from "birth_date". Now that we've formatted "birth_date" into a valid date object, we can call different methods of this object. Once these methods are called we can build our date string in the format of YYYY-MM-DD. After it's been built we set it in our dataObj, which will be sent to the server.

We'll look at the next three components at the same time since they access a similar method. The state (combo box), gender (radio button), and notify (check box) components use the getValue method to specify what selection the user has made. The methods in Listing 3 are assigned as change handlers to set the state, gender, and notification variables in our data object. To specify a change handler, select the component on the stage and go to the parameters tab in the properties panel (see Figure 2).

The comments field, a multiline text field that is set to wrap, ia assigned an instance name of "comments". The purpose of the instance name is to provide a way for the scrollbar component to reference the text field. Once the scrollbar is dragged onto the stage it needs to be assigned a target text field. As with all components, this is specified in the properties panel. After assigning this target (in our case it's "comments"), the scrollbar is ready to perform. Any text that goes beyond the viewable area will trigger the scrollbar, which has very advanced functionality. Play around with it and see if it doesn't remind you of a typical Windows scrollbar.

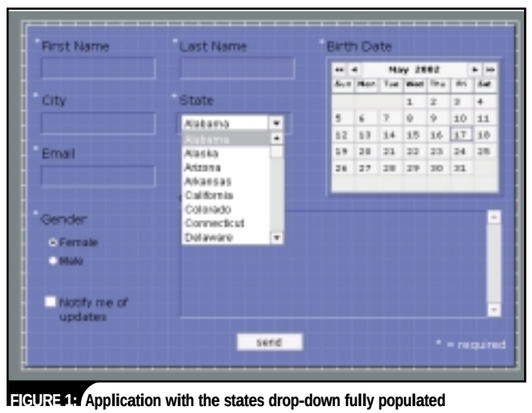


FIGURE 1: Application with the states drop-down fully populated

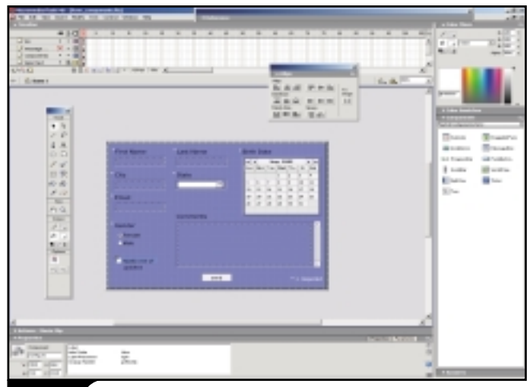


FIGURE 2: Properties panel with parameters set

So our user has completely filled out the form and clicks the send button – now what? The send button is actually a push-button component and is assigned the sendData method (see Listing 4) as its click handler.

Once the method is called, we assign our input text values to the data object. Since the data object was constructed as a LoadVars object, we have access to the sendAndLoad method, new to Flash MX. This method takes dataObj and sends it to the specified CF template, in our case "data_insert.cfm", to be handled by the server. The data is received by CF as name value pairs and by default is sent via the "POST" method. We can access these variables in CF as #form.first_name#, #form.last_name#, and so on.

The second parameter of the sendAndLoad method is a result function that can be used to handle the data sent back from CF.

The receive object was initialized at the beginning of the application when "init()" was called. This object waits for the response from CF and calls the "receiveHandler()" method when loaded (see Listing 5).

If the user's information is recorded successfully, we display the message box component (also part of the Flash UI Components Set 2) with the corresponding message. The send push button is disabled, via the setEnabled method, to prevent the user from accidentally resubmitting the form. If there's an error, we display the message clip with an error message. The user can acknowledge the error, correct the form, and submit it again.

Summing Up

Flash and ColdFusion integration may seem daunting at first, but with a little time and effort the knowledge you gain from building this basic form can be leveraged into building full-blown, integrated applications. Components are your friend, and they're here to save you time, money, and frustration. To develop this form from scratch, using components and writing the necessary code, could take from four to five hours. Without these components you'd be looking at three to four days to build a basic calendar…and you'd just be getting started! Take some time and get to know the component methods and how to access them. Learn how Flash and ColdFusion talk to each other and challenge yourself to build on this application. Try adding e-mail functionality where the user receives an e-mail after the form is submitted. The sky really is the limit when building integrated applications, so enjoy.

About the Author

Dennis Baldwin is the lead developer for Eternal Media, a Web/multimedia firm that provides technology solutions for ministries and nonprofit organizations (www.eternal-media.com). He also maintains a couple of online resources for Flash and ColdFusion developers at www.flashcfm.com and www.devmx.com.

@ DENNIS@FLASHCFM.COM

Listing 1

```
// method to initialize the application
function init() {
 // url where the application is running - change this to
suit your needs
 base_href = "http://localhost/form_components/";
 // set focus to the first_name fields
 Selection.setFocus("first_name");
 // initialize the dataObj which will hold the data passed
 // to CF via the sendData method
 dataObj = new LoadVars();
 // initialize the receive object that will handle the
response from the server
 receiveData = new LoadVars();
 // once the response comes back the receive handler will
be triggered
 receiveData.onLoad = receiveHandler;
 // set some default variables in dataObj
 dataObj.notify = false;
 dataObj.gender = "female";
 // create the loadVars object that will handle loading
 // the states from get_states.cfm
 var statesObj = new LoadVars();
 // once the states are loaded call the statesLoaded method
 statesObj.onLoad = statesHandler;
 // load the states
 statesObj.load(base_href + "get_states.cfm");
}
```

Listing 2

```
// method to handle the birth date once it's selected
function getBirthDate() {
 // call the getSelectedItem method of the calendar component
 // given the instance name of birthDate_mc
 birth_date = birthDate_mc.getSelectedItem();
 // create a valid date object from the selected date
 birth_date = new Date(birth_date);
 // set the birth year
 birth_year = birth_date.getFullYear();
 // set the birth month, add 1 since the month is zero indexed
 birth_month = birth_date.getMonth() + 1;
 // set the birth day
 birth_day = birth_date.getDate();
 // format the birth_date as YYYY-MM-DD so we can pass this
to CF
 birth_date = birth_year + "-" + birth_month + "-" +
birth_day;
 // set the birth_date variable in dataObj
 dataObj.birth_date = birth_date;
}
```

Listing 3

```
// method to set the selected state, from the combo box,
in dataObj
function getState() {
 dataObj.state = state_mc.getValue();
}

// method to set the selected gender, from the radio buttons,
in dataObj
function getGender() {
 dataObj.gender = gender_radio.getValue();
}

// method to set the notify flag, from the checkbox, in
dataObj
function getNotify() {
 dataObj.notify = notify_mc.getValue();
}
```

Listing 4

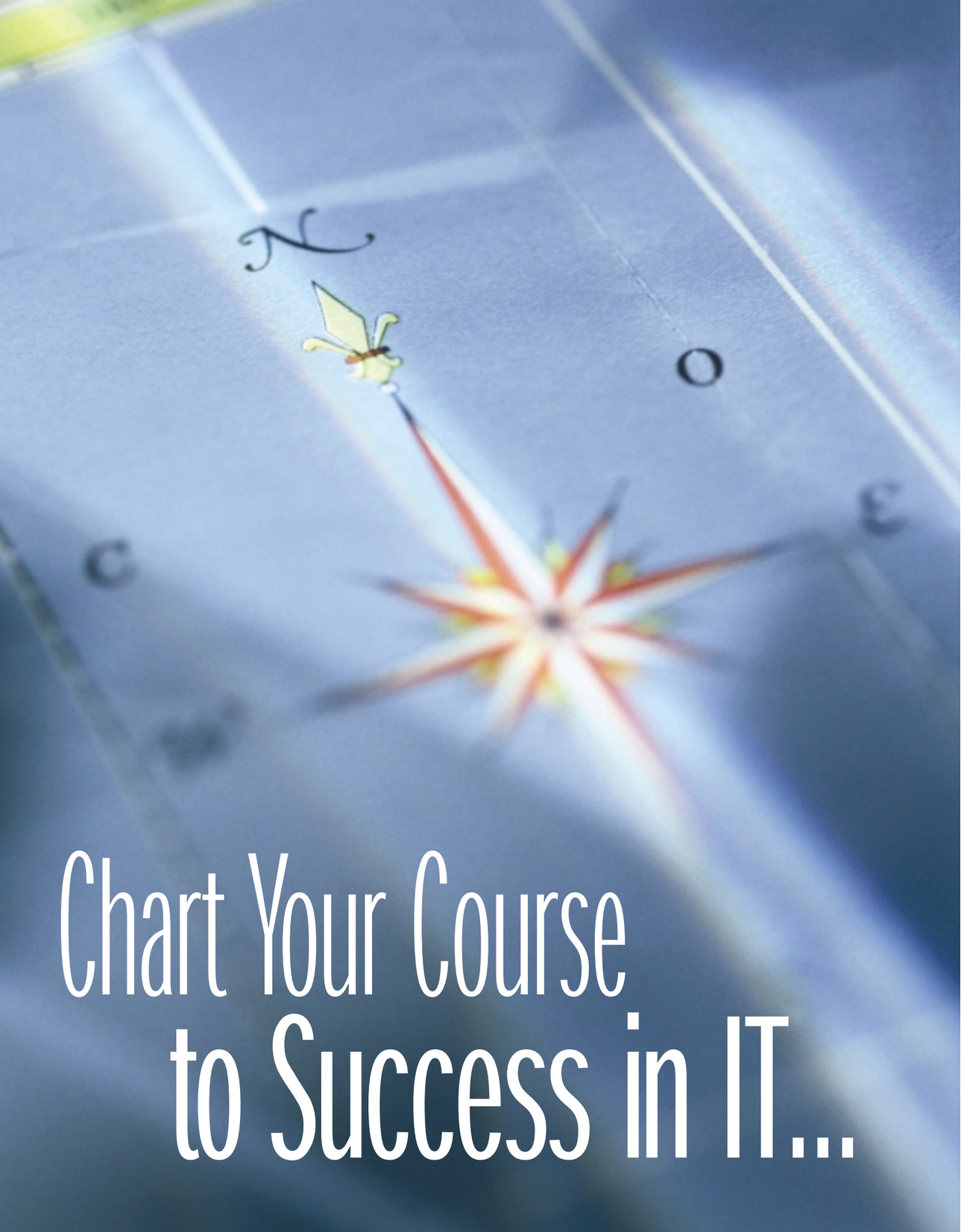
```
// method to send the data to CF
function sendData() {
 // set input text values into our data object
 dataObj.first_name = first_name;
 dataObj.last_name = last_name;
 dataObj.city = city;
 dataObj.email = email
 // since the comments field is assigned an instance name
 // we access the text of the field via comments.text
 dataObj.comments = comments.txt;
 dataObj.sendAndLoad(base_href + "data_insert.cfm",
receiveData);
}
```

Listing 5

```
// handles the response from the server after the informa-
tion has been submitted
function receiveHandler(success) {
 // if the transfer was successful display a success mes-
sage
 if(success) {
 statusMessage_mc.setMessage("Thank you for filling out our
form. Your information has been recorded successfully.");
 statusMessage_mc._visible = 1;
 // disable the send clip so they can't send the informa-
tion again
 send_mc.setEnabled(false);
 // if the transfer was not successful display an error
message
 } else {
 statusMessage_mc.setMessage("Please be sure to fill out
all required fields.");
 statusMessage_mc._visible = 1;
 }
}
```

CODE LISTING

The code listing for this article is also located at www.sys-con.com/coldfusion/sourcec.cfm

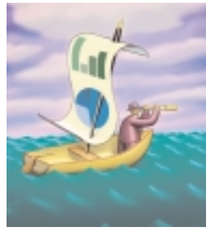


...order your copy of

Java Trends: 2003

Available July 1*

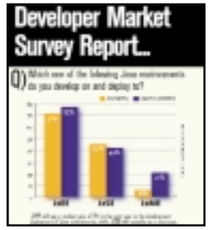
Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?



EnginData Research has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.

EnginData's research is invaluable to leading IT vendors, integrators, Fortune 500 companies, trade-show organizers, publishers, and e-business organizations worldwide.



Here's just a sample of the critical data points the Java report delivers...

- ✔ Current and projected usage of languages and platforms
- ✔ Multiple rankings of hundreds of software vendors and tools
- ✔ Types of applications being developed
- ✔ Databases being used
- ✔ Purchasing patterns
- ✔ Company size
- ✔ Development and purchasing timelines
- ✔ Perceptions and usage of Web services
- ✔ Preferred Java IDE
- ✔ J2EE, J2ME, J2SE usage comparisons

As an IT specialist, **EnginData Research** focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- ✔ Time frame for developing Web services – enabled apps
- ✔ Percentage of apps with Web services today
- ✔ Sourcing and hosting Web services
- ✔ Perceived leaders in Web services tools and solutions

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of **EnginData Research**. Order your copy of the 2002–2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.

engindata RESEARCH

www.engindata.com

*A limited number of preview copies will be available at our booth (#624) at JDJEdge International Java Developer's Conference & Expo, June 24–27, at the Jacob Javits Convention Center in New York.

CF Community

Tales from the **List**

Welcome MX!

BY **SIMON HORWITH**

ABOUT THE AUTHOR

Simon Horwith, a senior developer and Macromedia-certified ColdFusion instructor at Fig Leaf Software in Washington, DC, has been using ColdFusion since version 1.5. He is a contributing author to Professional ColdFusion 5.0 *(WROX) as well as technical editor of* The ColdFusion 5.0 Certification Study Guide *(Syngress).*

Developers were given their first glimpse of the next generation of ColdFusion at last year's Developers Conference, and periodic discussion/speculation has made the **CFDJList** ever since. In May Macromedia finally lifted the nondisclosure agreement on ColdFusion MX and the list was all abuzz. List subscribers could finally stop speculating and get solid answers to their questions from the many list members who have been playing with the various alpha and beta releases. Formerly bound by the agreement, these developers were finally able to have a public exchange of tales of "experience-thus-far" with the new server technology.

So what was the consensus? Overwhelmingly, developers working with the new server software for the past several months feel this is the most significant major release of ColdFusion to date. Among those chiming in were Ray Camden, Jeffrey Houser, and Ray Thompson, to name a few. And me, of course.

The questions varied, though most concerned CFCs – ColdFusion Components. The thread began with Patrick Whittingham providing a link to a Macromedia press release on MX technology. Ray Camden then let everyone know (with delight) that he and the other beta testers are now allowed to publicly discuss ColdFusion MX, as it had gone to public release candidate version that very morning. Patrick added that with Flash and CFMX technology he can create applications rather than HTML pages. He then provided analyst excerpts to back up his statement that "Macromedia is 'worlds' above Microsoft/IBM/Sun on the new rich-client Internet-based browser."

CFCs are an object-oriented implementation framework for ColdFusion developers. Plain and simple, they're going to radically change the way developers architect ColdFusion applications. A CFC may expose itself as an object with methods and properties that are callable from other ColdFusion templates, other servers, forms, and Flash MX applications. Because component functions can be easily written to be SOAP and .NET compliant (by specifying in a single attribute that they are for "remote" use), Flash movies, ColdFusion templates, and other Web-enabled applications can easily invoke their functionality (via RPC).

Macromedia has made the process of "talking" between Flash MX and ColdFusion MX applications all but trivial, not just with components but with server-side ActionScript as well. The ability, finally, to really use ColdFusion for object-oriented design and implementation has taken development on the platform to a new level. The ability to parse and work with XML has been addressed, and the new XML functionality in CFML is awesome. There's lots of buzz about new tags such as <CFFUNCTION>, <CFCOMPONENT>, and <CFXML>, and many new XML functions. Truly, CFMX will continue to be a hot topic of discussion for at least the next several months. The wonderful thing about CFMX is that these "popular topics" of discussion, while probably the most significant additions to CFML, are just a few features among dozens of significant enhancements and additions.

I'm happy to say that if it hadn't been true before, ColdFusion is now all grown-up. For the most part, developers already familiar with CFML don't have to forget what they've learned, but there will be a transition period as we reinvent the way we work and learn to harness the full potential of this new breed of CFML.

In addition to CFMX, there was another hot topic, one that may surprise some readers – Dreamweaver! While little controversy surrounds the new CFMX Server, the announcement that the development community shouldn't expect any new versions of ColdFusion Studio, but rather a new "coder-friendly" version of Dreamweaver MX, wasn't embraced as widely by the community. This is to be expected, as most CF developers write their code with ColdFusion Studio, know ColdFusion Studio, and like the Studio environment. After several years of working with one interface, it's going to take some time for them to become comfortable with Dreamweaver. The majority of the Dreamweaver MX list discussions involved (mostly) negative comments about Dreamweaver or speculative remarks about the new "HomeSite+" application that promises to ship with Dreamweaver at final release.

I've played around with Dreamweaver MX for a while now, and I must say that the new version hasn't been given a fair shake by most members of the community. As time goes on I'm confident that this will change and that developers will come to rely with confidence on Dreamweaver MX just as they did when Studio was first introduced several years ago. In Dreamweaver we're giving some things up, but we're gaining many things as well. That's what change is all about. All in all, it has some amazing features that I wish we'd had long ago.

That said, I too am curious about HomeSite+…and it's sure to be a future hot topic of discussion on the list.

• • •

If you aren't yet a subscriber to the CFDJList, I urge you to join now. With the new release of ColdFusion and the entire MX suite of products comes the need to learn. Contributing to and following the discussion threads on the list will likely prove invaluable. And don't be surprised to find the next several installments of "Tales from the List" to be devoted entirely to this topic.

@ SHORWITH@FIGLEAF.COM

CFDJnews

Macromedia Introduces CFMX

(San Francisco) – Macromedia, Inc., has made it official: Macromedia ColdFusion MX is finally here. Previously code-named *Neo*, CFMX reportedly brings the ease of use and productivity of ColdFusion to the highly scalable, standards-based Java technology architecture. CFMX offers innovations for creating rich Internet applications and working with XML, Web services, and Microsoft's .NET framework.

Built on a breakthrough new architecture that delivers the scalability, reliability, and power of the Java platform without the complexity, customers can deploy CFMX as a standalone server or on top of Java application servers.

CFMX is an integral member of the new Macromedia MX product family that also includes Macromedia Flash MX and Dreamweaver MX. www.macromedia.com

More on Dreamweaver MX

(San Francisco) – As announced by Macromedia, Inc., Macromedia Dreamweaver MX is a revolutionary product that enables developers to design as well as code the full spectrum of solutions from Web sites to Internet applications

Dreamweaver MX combines the visual layout tools of Dreamweaver, the Web application development features of Dreamweaver UltraDev, and the extensive code editing support of Macromedia HomeSite into one approachable, powerful development environment. It also offers complete support for CFMX features, XML, and Web services. www.macromedia.com

Studio MX Speeds Rich Internet Apps

(San Francisco) – Macromedia is shipping Macromedia Studio MX, a suite of integrated tools for developing the full spectrum of Internet solutions.

Studio MX includes Dreamweaver MX, Flash MX, Fireworks MX, FreeHand 10, and a developer edition of Macromedia ColdFusion MX.

For more information, visit www.macromedia.com/go/studiomx/. Information on upgrading from one or more Macromedia products is available at www.macromedia.com/go/studioupgrade/.

Macromedia MX Seminar Tour

Easy. Powerful. Open. Come see firsthand how Macromedia Studio MX has everything you need to create the entire spectrum of Web solutions, from Web sites to rich Internet applications – all in one powerful, approachable, completely integrated solution. This three-day tour consisting of six different events is visiting cities across the U.S. Find out more at www.macromedia.com/go/q103cfdj.

CFDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
ACTIVE PDF	WWW.ACTIVEPDF.COM	949.582.9002	4
CFADVISOR	WWW.CFADVISOR.COM		49
CFDYNAMICS	WWW.CFDYNAMICS.COM	800.422.7957	23
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX.HOST	33
COLD FUSION DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	21
CTIA	WWW.CTIASHOW.COM		3
ENGINDATA RESEARCH	WWW.ENGINDATA.COM	201.802.3082	47
E-ZONE MEDIA	WWW.FUSETALK.COM	866.477.7542	37
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	39
INTERLAND	WWW.INTERLAND.COM	866.253.0827	2
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	52
JAVA DEVELOPER'S JOURNAL	WWW.SYS-CON.COM	800.513.7111	19
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	35
MACROMEDIA	WWW.VUE.COM/MACROMEDIA	877.460.8679	13
MACROMEDIA	WWW.MACROMEDIA.COM/GO/USERGROUPS	877.460.8679	17
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CERTIFICATION	877.460.8679	51
NEW ATLANTA	WWW.NEWATLANTA.COM		11
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	41
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	15
RACKSHACK	WWW.RACKSHACK.NET	800.504.SURF	9
WEB SERVICES EDGE	WWW.SYS-CON.COM	201.802.3069	26, 27
WEB SERVICES JOURNAL	WWW.SYS-CON.COM	800.513.7111	36

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of ***ColdFusion Developers Journal***. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in ***ColdFusion Developers Journal***. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.

Next Month...

Don't miss the July issue!

Macromedia's MX Family

By Kevin Lynch

What the new features will mean to you

MX and Web Services

By Ronald West

How to connect your CF app to the world

Palm Apps and CF Web Agents

By Joshua Oster-Morris

Linking external data with Web agents using CFHTTP and CF's sweet suite of string parsing functions

ColdFusion and XML

By Kevin Schmidt

A quick and easy Web service with CFMX

ColdFusion 5.0 and Microsoft's SOAP SDK

By Andrew Stopford

Create and consume a Web service with CF and the SOAP SDK

ColdFusion Developer's Journal