

# COLD FUSION Developer's Journal

ColdFusionJournal.com

May 2006 Volume:8 Issue:5

**Writing an RSS  
Aggregator PART 1** 10

**Creating Object-Oriented  
Presentation Layers** 16

**Feed Your Site** 26

**Designing ColdFusion  
Applications for Deployment  
as EAR Files** 32

**Advanced XML  
Processing with StAX** 38

**Stopping Spam in  
Its Tracks...** 42

## When AJAX Happens to Old Browsers

20

**Using Objects in Your  
Adobe ColdFusion  
MX Applications** 46

Presorted  
Standard  
US Postage  
PAID  
St. Croix Press







## *ColdFusion Hosting is our Complete Focus*

### ➤ **POWERFUL HOSTING PLANS**

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

### ➤ **RELIABLE NETWORK**

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

### ➤ **FANTASTIC SUPPORT SERVICES**

24/7 support services | Knowledgeable phone support | We focus on your individual needs

# CFDynamics

866.233.9626 ➤ [CFDYNAMICS.COM](http://CFDYNAMICS.COM)

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

*Real service. Real satisfaction. Real value. Real support. Real Freedom.*





# One little box. A whole lot of power.

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

---

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



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

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



› **Make rapid J2EE development a reality.**

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



› **New mobile applications are a go.**

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

To learn more or take ColdFusion MX 7 for a test drive, go to:  
[macromedia.com/go/cfmx7\\_demo](http://macromedia.com/go/cfmx7_demo)





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

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

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

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

**By the Numbers:**

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



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

<http://edgewebhosting.net>



**2003 - 2006**

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



**editorial advisory board**

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

**editorial****editor-in-chief**

Simon Horwith [simon@sys-con.com](mailto:simon@sys-con.com)

**technical editor**

Raymond Camden [raymond@sys-con.com](mailto:raymond@sys-con.com)

**executive editor**

Nancy Valentine [nancy@sys-con.com](mailto:nancy@sys-con.com)

**research editor**

Bahadır Karuv, PhD [bahadir@sys-con.com](mailto:bahadir@sys-con.com)

**production****lead designer**

Abraham Addo [abraham@sys-con.com](mailto:abraham@sys-con.com)

**art director**

Alex Botero [alex@sys-con.com](mailto:alex@sys-con.com)

**associate art directors**

Louis F. Cuffari [louis@sys-con.com](mailto:louis@sys-con.com)  
Tami Beatty [tami@sys-con.com](mailto:tami@sys-con.com)

**editorial offices****SYS-CON MEDIA**

135 Chestnut Ridge Rd., Montvale, NJ 07645  
Telephone: 201 802-3000 Fax: 201 782-9638  
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)  
is published monthly (12 times a year)  
by SYS-CON Publications, Inc.

**postmaster: send address changes to:**

COLDFUSION DEVELOPER'S JOURNAL  
SYS-CON MEDIA  
135 Chestnut Ridge Rd., Montvale, NJ 07645

**@copyright**

Copyright © 2006 by SYS-CON Publications, Inc.  
All rights reserved. No part of this publication may  
be reproduced or transmitted in any form or by any means,  
electronic or mechanical, including photocopy  
or any information, storage and retrieval system,  
without written permission.

**Worldwide Newsstand Distribution**

Curtis Circulation Company, New Milford, NJ  
FOR LIST RENTAL INFORMATION:  
Kevin Collopy: 845 731-2684, [kevin.collopy@edithroman.com](mailto:kevin.collopy@edithroman.com)  
Frank Cipolla: 845 731-3832, [frank.cipolla@epostdirect.com](mailto:frank.cipolla@epostdirect.com)

For promotional reprints, contact reprint  
coordinator Megan Mussa, [megan@sys-con.com](mailto:megan@sys-con.com).  
SYS-CON Publications, Inc., reserves the right to  
revise, republish and authorize its readers to use  
the articles submitted for publication.  
All brand and product names used on these pages  
are trade names, service marks, or trademarks  
of their respective companies.

# Showing Commitment to the Community



By Simon Horwith

I've written a lot lately about the growing strength of the ColdFusion development community – shown by the onslaught

of frameworks and the rapid adoption and support for these frameworks, the growing number of bloggers, new conferences and the great success of the CFUnited conference, and by the commitment to the community being shown by Adobe.

Adobe has shown their support in so many ways including Adobe Labs, the CFEclipse RDS support and wizards, and the easy-to-use new Flex/CF integration, to name a few. This is in addition to the fact that at least one, if not several, of the members of the ColdFusion Server Team is in attendance at nearly every CF event throughout the year. This month, I thought I'd talk about some things that every developer can do to help support the community, and about what we at SYS-CON are also doing to help out.

Not too long ago, Ray Camden wrote in a blog entry that "you don't have to be a guru to contribute code to the community." I'll be honest; my first reaction was to rebut. Code that is written for the purpose of being reused by the community and that is made publicly available should be well written. The idea of novice developers opening files that are freely available in order to read the code and learn from it, mistakenly using this poorly written

code as an example to learn from, makes me nervous. For that reason I do disagree with Ray: if a developer is going to write an application, API, or other complex bit of code for others to reuse, they should not only be confident but competent as well. However, this does not mean that they cannot contribute.

As the community grows and the number of open source projects continues to increase, I envision some of these applications not only having a significantly large code base, but to be developed and managed in such a way that it is relatively easy for any somewhat decent developer to contribute to the project code base in some way. Some of the projects are organized this way, but not nearly as many as those with a single file that are really only editable by one person (presumably the original author) at a time. Whether this idea of many large projects that anyone should be able to contribute to comes to fruition or not, I suggest that writing and distributing project code should be the last form of contribution to the community on a developer's path from "newbie" to "guru." What then can a developer do

**About the Author**

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

# BALANCE

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

[www.communitymx.com](http://www.communitymx.com)



Visit [www.communitymx.com/trial/](http://www.communitymx.com/trial/) for your free 10 day trial.





## president & ceo

Fuat Kircaali fuat@sys-con.com

## group publisher

Jeremy Geelan jeremy@sys-con.com

## advertising

### senior vp, sales & marketing

Carmen Gonzalez carmen@sys-con.com

### vp, sales & marketing

Miles Silverman miles@sys-con.com

### advertising director

Robyn Forma robyn@sys-con.com

### advertising manager

Megan Mussa megan@sys-con.com

### associate sales manager

Kerry Mealia kerry@sys-con.com

Lauren Orsi lauren@sys-con.com

## sys-con events

### president, events

Grisha Davida grisha@sys-con.com

### national sales manager

Jim Hanchrow jimh@sys-con.com

## customer relations

### circulation service coordinator

Edna Earle Russell edna@sys-con.com

### manager, jdj store

Brunilda Staropoli bruni@sys-con.com

## sys-con.com

### vp, information systems

Robert Diamond robert@sys-con.com

### web designers

Stephen Kilmurray stephen@sys-con.com

Wayne Uffelman wayne@sys-con.com

### online editor

Roger Strukhoff roger@sys-con.com

## accounting

### financial analyst

Joan LaRose joan@sys-con.com

### accounts payable

Betty White betty@sys-con.com

### accounts receivable

Gail Naples gailn@sys-con.com

## subscriptions

### Subscribe@sys-con.com

Call 1-888-303-5282

For subscriptions and requests for bulk orders, please send your letters to Subscription Department

Cover Price: \$8.99/issue

Domestic: \$89.99/yr (12 issues)

Canada/Mexico: \$99.99/yr

All other countries \$129.99/yr

(U.S. Banks or Money Orders)

Back issues: \$12 U.S. \$15 all others

to contribute no matter what their level of expertise?

First, I suggest every novice developer do two things: absorb the wealth of information that's publicly available and practice, practice, practice! Absorbing information is best accomplished by subscribing to a popular e-mail list server such as CF-Talk and reading the posts, reading/subscribing to many of the excellent ColdFusion blogs that are out there, and by reading the ColdFusion documentation (available free on the Adobe site), *ColdFusion Developer's Journal*, and the articles and tech notes published on the adobe.com Website. Practicing is just that: spend free time or time in the office (if you can) figuring out how to do things with ColdFusion that you don't already know how to do. Most developers learn more through trial and error than any other medium. Start with built-in tags and functions that you don't know or don't know well, and then move on to practicing writing custom tags and ColdFusion components. ColdFusion is not rocket science and the more you read and practice, the better you will become at your trade... and in a very short amount of time, if you put effort into it, enjoy what you do.

As you begin to feel more confident in your skills, don't only read e-mail list server posts but also begin replying to them. If there are less knowledgeable developers in your office and you have the opportunity, let them know that you're available to help and mentor them whenever you have the chance. Begin reading other people's code and critiquing it – so much can be learned by looking at what others have written. Attend conferences and user group meetings (I'm sure there's a user group near you – if not, start one!) – you will learn so much from the speakers you encounter and will make many valuable connections by socializing at these events. If you are comfortable doing so, offer to present at an upcoming user group meeting. If you have given several user group presentations and feel ready for it, begin submitting topics to conferences. Send me an e-mail with a topic proposal for an article. Don't worry about grammar or content – our editors will clean up grammar and Ray or I will help with content suggestions. With a little effort, most if not all of these things can be accomplished within one year – even if it's your first year as a ColdFusion developer. Then, when the time is right, I suggest volunteering code to the community in the form of

an open source project, if that's what you want to do.

Is all of this necessary in order to contribute code to the community? Certainly not at all... but I receive a lot of e-mails and meet a lot of developers who want to get involved but clearly don't know where to begin. There are so many ways you can get involved in the community; I suggest starting slow by exploring them all from a safe distance and getting more involved in the means that best suit you. We don't all enjoy having our code analyzed, or standing in front of a crowded room to which we must educate and entertain for an hour or more, but we can each contribute in our own way.

At SYS-CON, we are constantly looking to improve our offerings to the community. We do this via enhancements to the SYS-CON and magazine Websites, by improving existing and adding new magazines, and by sponsoring and organizing events. In fact, I am writing this editorial from a train on my way to New York City for the SOA Web Services Edge Conference (organized and run by SYS-CON) where I am speaking on rapidly developing SOA-driven Web 2.0 applications with Flex 2.0 and ColdFusion MX. Not too long ago, in response to the AJAX and Web 2.0 craze, SYS-CON launched *AJAX Developer's Journal*, which is an excellent magazine (Rob Gonda, long time CF developer and AJAX nut, is the editor-in-chief). Most recently, *MX Developer's Journal* was renamed and somewhat refocused. Beginning last month, the magazine formerly known as *MXDJ* is now known as *Web Developer's & Designer's Journal*. The magazine is more focused on rich application development – particularly using the Flash, Flex, and Dreamweaver products from Adobe. "But what about our beloved CFDJ?"... I'm glad you asked.

*ColdFusion Developer's Journal* is now in its sixth year and going strong. Over the past couple of years, since assuming the role of editor-in-chief, we've experimented with several ideas about content and format, and have sought feedback from our readers. We have recently been making decisions about the future of the magazine and have found the feedback to be invaluable in the decision-making process. I thought I'd briefly share the decisions we've made so far, as well as some of the feedback that helped in making these decisions.

– continued on page 29

# TABLE OF CONTENTS



## When AJAX Happens to Old Browsers

By Jeremy Lund...20



## Designing ColdFusion Applications for Deployment as EAR Files

By Jochem van Dieten and Mark van Hedel...32



## Advanced XML Processing with StAX

By Jim Collins...38

### editorial

Showing Commitment to the Community

By Simon Horwith

5

### cf 101

Writing an RSS Aggregator PART 1

Building an application

By Jeffrey Houser

10

### foundations

Creating Object-Oriented Presentation Layers

All the coolest programmers are doing it

By Hal Helms

16

### rss feeds

Feed Your Site

Incorporating RSS and Atom feeds into your ColdFusion Site

By Jeremy Lund

26

### cfugs

ColdFusion User Groups

30

### open source

Stopping Spam in Its Tracks

... using CAPTCHA

By Brian Rinaldi

42

### cfmx

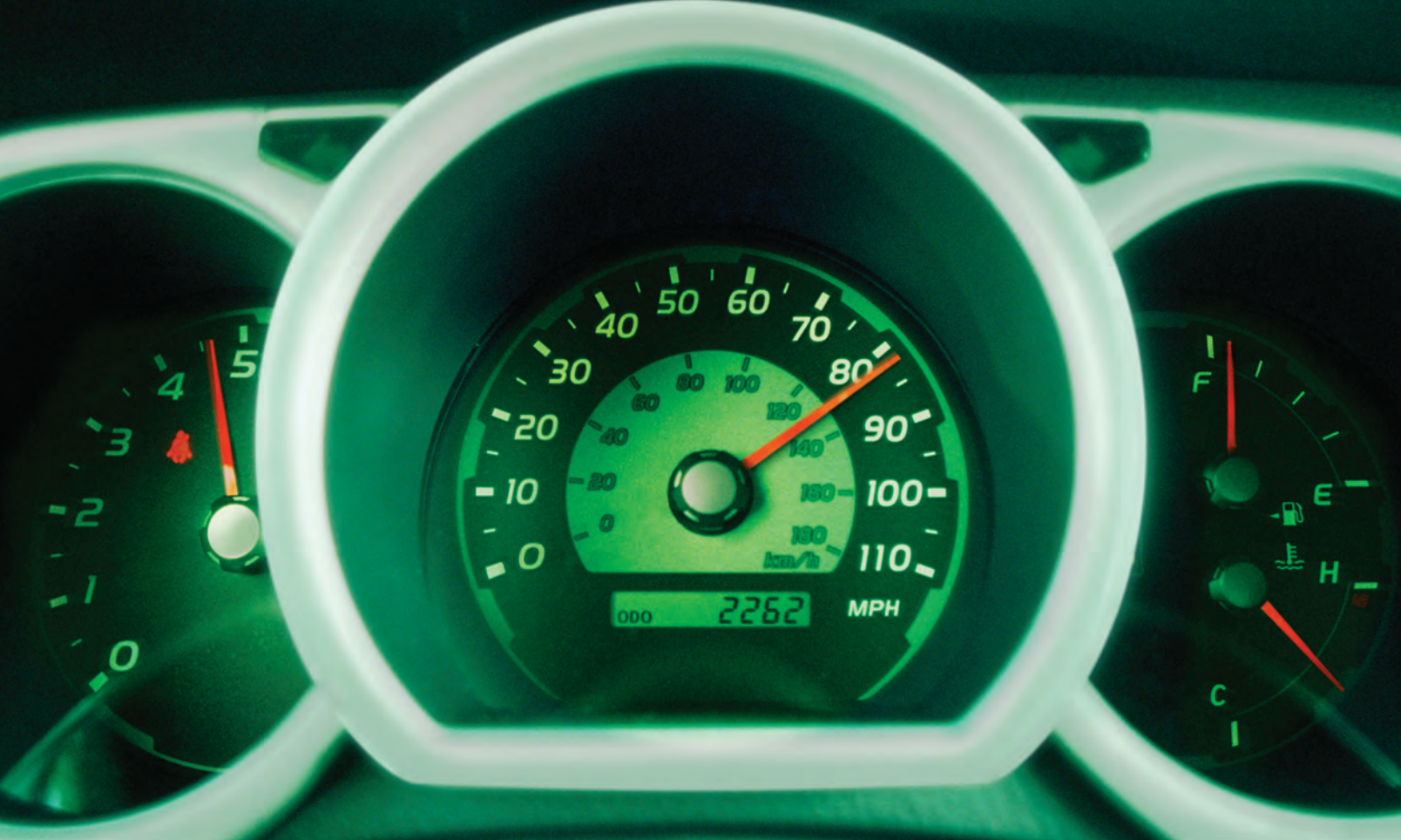
Using Objects in Your Adobe ColdFusion MX Applications

Developing reusable, maintainable code

By Nicholas Tunney

46





## Rev Up Your Flash Video

### Stay Ahead of the Competition With VitalStream and the Enhanced Video Features in Flash 8

With over two years of experience in delivering much of today's most popular media, VitalStream® is the first and most experienced Flash™ video streaming service provider.

#### Enhanced Flash 8 Video Features:

- New VP6 codec delivers higher quality video at the same bit rate
- 8-bit alpha channel transparency enables you to blend video with other elements
- Improved live video capabilities

#### VitalStream Complete Toolset for Flash:

- MediaConsole®
- MediaOps™ SDK
- Flash Authentication
- Reporting Dashboard



*Integrate Streaming Media  
Into Your Flash Projects*

Take Advantage of the Enhanced Video Features in Macromedia Flash 8  
Call (800) 254-7554 or Download Tutorials at [www.vitalstream.com/go/mxdj](http://www.vitalstream.com/go/mxdj)

# Writing an RSS Aggregator Part 1

## Building an application



By Jeffrey Houser

So often in this column I feel that I'm writing about basic concepts and using trivial examples.

It's often up to you, as the reader, to figure out how to apply these concepts to your development. I thought it might be a good

idea to take some space to try to bring a lot of the concepts together and build an application. I don't always take the space to do so, but thought I'd give it a try here.

I'm sure that most of you read (or at least have heard of) blogs. Blogs are a great way to keep up on the up-to-the-minute comings and going of whatever community you choose. There are tons of bloggers throughout the ColdFusion community, some blogging from inside Adobe. Many blogs provide an RSS feed for content. RSS stands for Really Simple Syndication and is an XML dialect. There are a handful of RSS aggregators in the ColdFusion community such as mxna (<http://weblogs.macromedia.com/mxna/>), Full As a Goog (<http://www.fullasagoog.com/>), and Feed Squirrel (<http://www.feed-squirrel.com/>). I've always thought it would be cool to create something like a LiveJournal friends list of my very own. In this series of articles, I'm going to step you through the thought process that I'd go through to create one. First I'll concentrate on designing the database and fleshing out the ColdFusion components.

### Designing the Back End

When developing an application, I normally start with the database design, and that is where we'll start here. If you need a refresher on database design techniques, you can review the database design article from this column at <http://coldfusion.sys-con.com/read/49177.htm>. What data do you want to store in an RSS Aggregator application? You can read all about the RSS 2.0 spec at <http://www.rssboard.org/rss-specification>. For purposes of this article, I'm only going to concentrate on required fields.

- **RSS Feeds:** The RSS feed seems like a good place to start. The app will have to store the name of the RSS feed and the URL

location of the feed.

- **Items:** Since we're aggregating feed data from a lot of different sources, we'll need to store the posts from each individual feed. It'll be more efficient to store them locally than to create the feed manually each time we want to display something. A post is put into the RSS XML document as an item. According to the spec, nothing is required in the item except for title or description; there's no guarantee that one will be there. For the sake of this article, we'll assume that title, link, and description will be there. Items have a one-to-many relationship with RSS feeds. Each blog can contain many posts, but each post belongs to only one blog. To implement this in the database, we put a primary key for the RSS feed into the items table.
- **Categories:** Within the app, I want to be able to categorize each feed, so I can filter the group of messages that I'm reading. We'll add a category table to the database. This data won't be coming from the RSS, but some identifier that we create.

The database structure is shown in Figure 1.

For the model portion of this app, I'm going to use ColdFusion components. You can read more about component basics in two of my articles at <http://coldfusion.sys-con.com/read/47203.htm> and <http://coldfusion.sys-con.com/read/47446.htm>. I'd create one CFC for each element: an RSS Feed, an Item, and a Category. You can see the model in Figure 2. You may notice that the CFCs are named in the singular tense, while the database tables are plural. This is because each CFC is designed to represent a single item, while the database tables contain many items. I left out individual getter and setter methods from the diagram, but the code will use the generic getter and setter methods from Hal Helm's BaseComponent.cfc <http://halhelms.com/webresources/BaseComponent.cfc>. Each component includes an init and a commit method. The init will load information out of the database, while the commit will create (or update) information in the database.



Figure 1 Database structure





**WEB ALL-STAR** seeks an Integrated Software Suite that's up for anything. Understand my need to create cool graphics one day, while tweaking CSS the next. Should be intuitive, multi-platform and not phased by my turbo-charged pace. High-maintenance is a no-no.



## Different people. Different needs. One suite solution.

With the latest versions of Macromedia Dreamweaver®, Flash® Professional, Fireworks®, Contribute™, and FlashPaper™, the new Studio 8 is quite a catch. To meet Studio 8 and find all the tools you need to design, develop and maintain online experiences, visit [www.macromedia.com/go/studio8\\_mxdj](http://www.macromedia.com/go/studio8_mxdj)

macromedia®  
**STUDIO 8**



The RSSFeed and RSSCategory components will be used for maintaining feeds and categories. You might notice that the RSSFeed component contains an RSSCategory component instead of the CategoryID foreign key. I thought about putting an array of item objects in the RSSFeed, but decided against it. The application won't need to load all the items from the feed in question; a gateway object can be used to handle this type of action when viewing the data (although that's beyond the scope of this article).

The RSSAggregator component is where most of the magic takes place. It will be the background process, run as part of a scheduled task. It will loop through each feed in the database, collect the latest data, and store the new items in the database. It will use the item component

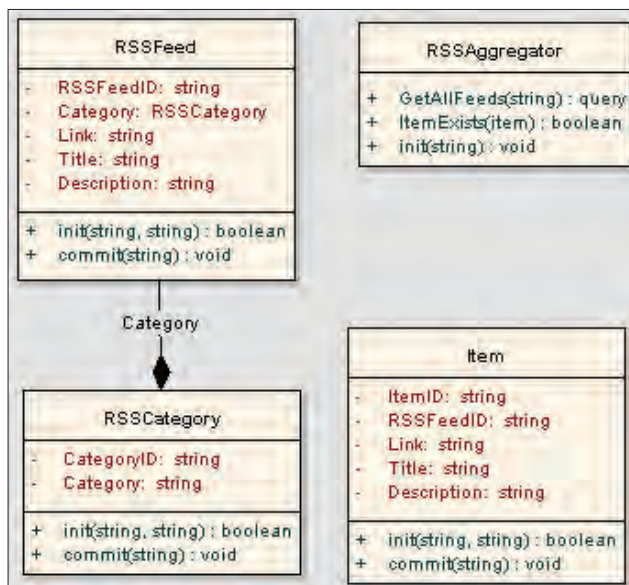


Figure 2: Object model

## Creating the RSS Feed Components

In real-world applications, I often like to start by building the administrator first. That way I don't have to worry about filling the tables with fake data to test the public-facing part of the site. Before we can start to process feeds on the back-end, we have to be able to enter the feeds into the database. We can start by fleshing out the RSSFeed and RSSCategory components. First, let's examine the RSSCategory component, as shown in Listing 1. Nothing in the RSSCategory.cfc should surprise you. It contains two instance variables, the CategoryID and Category. The CategoryID is initialized as a string because this app uses UUIDs as primary keys instead of auto-incrementing integers. The component has two methods (plus a few inherited from Hal's BaseComponent.cfc). There's an init method that accepts the CategoryID and the datasource. It queries the database and sets the instance variables, if applicable. The second method is a commit method. It accepts the datasource name as an argument. If the CategoryID is blank, a new category is created. If it isn't, an update is performed. This isn't much different, in concept, from the address.cfc I wrote about in one of my previous

component articles.

The RSSFeed component is shown in Listing 2. You'll notice many similarities to the RSSCategory component. There are more properties in RSSFeed than RSSCategory, but the concept is the same. You can notice that the category property of the component is an instance of the RSSCategory component. Instead of containing the category information directly, only the reference is stored. To access the category information, we can use the get, or set, methods off the component instance. The set methods are demonstrated in init; while the get methods are demonstrated in commit.


## Entering the Data

To enter the data, a simple HTML form is used, as shown in Listing 3. The first lines include a syslib.cfm file. This is just a utility library of functions. Two functions from the library are used in this component. The GetCategories function contains a query to get all categories. The "CreateSelectList" function is very similar to the cfselect tag for those who don't use cfform. (I'll make sure that all the code behind this article gets up on my blog; I left out the syslib.cfm for brevity).

The file creates an instance of the RSSFeed component. If an RSSFeedID is defined, it will run the init method; otherwise the base component is used. A form is created that asks for the category, and the http link of the RSS feed. The title and description are displayed on this form, but are actually populated from the RSS feed, so they're not editable.

The form submits onto Feedip.cfm, which is shown in Listing 4. The initialization code is the same as in Listing 3. Then the code enters a try block. To get the description and title, we need to call the RSS feed. If the cfhttp call times out, we know that the RSS feed is wrong. The URL could be invalid in other ways, but that's the only thing we check for in this code. XMLParse is called on cfhttp.filecontent. I wrote an article about getting and processing XML feeds this way, and you can read it at <http://coldfusion.sys-con.com/read/117667.htm>. The remainder of the code sets the instance variables and calls the commit. Congratulations, you've saved a feed in the database.

## What's Next

I've just about run out of space for this article, but this app is far from complete. In the next article I'll talk about the RSSAggregator and item components. It will create the scheduled task to call the aggregation code. Space permitting, I'll go into a gateway component along with some code that can be used to look at the aggregated data. If there isn't space in the next article for that, I'll fit it into some future article. Until next time, keep coding. 

## About the Author

Jeff Houser has been working with computers for over 20 years. He owns a DotComIt, a web consulting company, manages the CT Macromedia User Group, and routinely speaks and writes about development issues. You can find out what he's up to by checking his Blog at [www.jeffryhouser.com](http://www.jeffryhouser.com).

[jeff@instantcoldfusion.com](mailto:jeff@instantcoldfusion.com)



## Listing 1

```
<cfcomponent extends="basecomponent">
<cfscript>
    variables.instance.CategoryID = "";
    variables.instance.Category = "";
</cfscript>

<cffunction name="Init" access="public" returntype="Boolean">
    <cfargument name="CategoryID" type="string" required="true">
    <cfargument name="dsn" type="string" required="true">
    <cfset var qGetCat = "">
    <cfquery name="qGetCat" datasource="#arguments.dsn#">
        select *
        from RSSCategories
        where RSSCategories.CategoryID = <cfqueryparam value="#arguments.
CategoryID#" cfsqltype="cf_sql_varchar">
    </cfquery>
    <cfif qGetCat.recordcount is 0>
    <cfreturn false>
    </cfif>
</cfscript>
    variables.instance.CategoryID = qGetCat.CategoryID;
    variables.instance.Category = qGetCat.Category;
</cfscript>

<cfreturn true>

</cffunction>

<cffunction name="Commit" access="public" returntype="void">
    <cfargument name="dsn" type="string" required="true">
    <cfset var qUpdateCat = "">

    <cfif variables.instance.CategoryID is "">
    <cfset variables.instance.CategoryID = createuuid()>
    <cfquery name="qUpdateCat" datasource="#arguments.dsn#">
        insert into RSSCategories(CategoryID, Category)
        values (
            '#variables.instance.CategoryID#',
            <cfqueryparam value="#variables.instance.Category#"
cfsqltype="cf_sql_varchar"> )
    </cfquery>
    <cfelse>
    <cfquery name="qUpdateCat" datasource="#arguments.dsn#">
        update RSSCategories
        set Cate = <cfqueryparam value="#variables.instance.Cate#"
cfsqltype="cf_sql_varchar">
        where CategoryID = <cfqueryparam value="#variables.instance.
CategoryID#" cfsqltype="cf_sql_varchar">
    </cfquery>
    </cfif>
</cffunction>
```

```
</cffunction>
```

```
</cfcomponent>
```

## Listing 2:

```
<cfcomponent extends="basecomponent">

<cfscript>
    variables.instance.RSSFeedID = "";
    variables.instance.Category = createObject('component', 'RSSCategory
');
    variables.instance.Title = "";
    variables.instance.link = "";
    variables.instance.description = "";
</cfscript>

<cffunction name="Init" access="public" returntype="Boolean">
    <cfargument name="RSSFeedID" type="string" required="true">
    <cfargument name="dsn" type="string" required="true">
    <cfset var qGetRSSFeed = "">
    <cfquery name="qGetRSSFeed" datasource="#arguments.dsn#">
        select RSSFeeds.*, RSSCategories.*
        from RSSFeeds left outer join RSSCategories on (RSSFeeds.CategoryID
= RSSCategories.categoryID)
        where RSSFeeds.RSSFeedID = <cfqueryparam value="#arguments.RSS-
FeedID#" cfsqltype="cf_sql_varchar">
    </cfquery>
    <cfif qGetRSSFeed.recordcount is 0>
    <cfreturn false>
    </cfif>
</cfscript>
    variables.instance.RSSFeedID = qGetRSSFeed.RSSFeedID;
    variables.instance.Category.Set('categoryID', qGetRSSFeed.Catego-
ryID);
    variables.instance.Category.Set('category', qGetRSSFeed.Category);
    variables.instance.Title = qGetRSSFeed.Title;
    variables.instance.link = qGetRSSFeed.link;
    variables.instance.description = qGetRSSFeed.description;
</cfscript>
<cfreturn true>
</cffunction>

<cffunction name="Commit" access="public" returntype="void">
    <cfargument name="dsn" type="string" required="true">
    <cfset var qUpdateItem = "">
    <cfif variables.instance.RSSFeedID is "">
    <cfset variables.instance.RSSFeedID = createuuid()>
    <cfquery name="qUpdateItem" datasource="#arguments.dsn#">
        insert into RSSFeeds (RSSFeedID, CategoryID, Title, Link, Descrip-
tion)
        values (
```

```

    '#variables.instance.RSSFeedID#',
    <cfqueryparam value="#variables.instance.Category.
get('categoryID')#" cfsqltype="cf_sql_varchar">,
    <cfqueryparam value="#variables.instance.Title#" cfsqltype="cf_
sql_varchar">,
    <cfqueryparam value="#variables.instance.Link#" cfsqltype="cf_sql_
varchar">,
    <cfqueryparam value="#variables.instance.Description#"
cfsqltype="cf_sql_varchar"> )
</cfquery>
<cfelse>
<cfquery name="qUpdateItem" datasource="#arguments.dsn#">
update RSSFeeds
set CategoryID = <cfqueryparam value="#variables.instance.Category.
get('categoryID')#" cfsqltype="cf_sql_varchar">,
    Title = <cfqueryparam value="#variables.instance.Title#"
cfsqltype="cf_sql_varchar">,
    Link = <cfqueryparam value="#variables.instance.Link#"
cfsqltype="cf_sql_varchar">,
    Description = <cfqueryparam value="#variables.instance.Descrip-
tion#" cfsqltype="cf_sql_varchar">
    where RSSFeedID = <cfqueryparam value="#variables.instance.RSS-
FeedID#" cfsqltype="cf_sql_varchar">
</cfquery>
</cff>
</cffunction>
</cfcomponent>

```

### Listing 3

```

<cfinclude template="#request.cflibs#/syslib.cfm">

<cfscript>
variables.myFeed = CreateObject("component", "#request.ComponentLoc#.
RSSFeed");
if (IsDefined("RSSFeedID")){
    variables.myFeed.init(RSSFeedID, request.dsn);
}
</cfscript>

<cfoutput>
<form action="Feedip.cfm" method="post">
    <input type="hidden" name="RSSFeedID" value="#variables.myFeed.
get('RSSFeedID')#">
    <strong>Category</strong>:
    #CreateSelectList(getCategories(request.dsn), "CategoryID", "Category
", "CategoryID",
variables.myFeed.Get('Category').get('CategoryID'), "Select a Cat-
egory",
    "", "")#<br>

```

```

    <strong>Link</strong>: <input type="text" name="link"
value="#variables.myFeed.get('link')#"><br>
    <strong>Title</strong>: <cff variables.myFeed.get('title') is
"">Unknown, will be pulled out of Feed<cfelse>#variables.myFeed.
get('title')#</cff><br>
    <strong>Description</strong>: <cff variables.myFeed.
get('Description') is "">Unknown, will be pulled out of
Feed<cfelse>#variables.myFeed.get('Description')#</cff><br>
    <input type="Submit">
</form>
</cfoutput>

```

### Listing 4

```

<cfscript>
variables.myFeed = CreateObject("component", "#request.ComponentLoc#.
RSSFeed");
if (form.RSSFeedID NEQ ""){
    variables.myFeed.initbyID(RSSFeedID, request.dsn);
}
</cfscript>

<cftry>
<cfhttp url="#form.link#" timeout="30">
</cfhttp>

<cfcatch type="any">
    Warning, the link timed out.
<cfabort>
</cfcatch>

</cftry>

<cfscript>
MyXMLVar = xmlparse(cfhttp.filecontent);
variables.myFeed.set('link', form.link);
variables.myFeed.set('title', MyXMLVar.rss.channel.description.
xmltext);
variables.myFeed.set('description', MyXMLVar.rss.channel.title.
xmltext);
variables.myFeed.get('category').set('categoryID', form.categoryID);
variables.myFeed.commit(request.dsn);
</cfscript>

```

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



# Enter the premier ColdFusion conference

**Register  
Today!**

CFUNITED is the only conference of its kind that is run by developers, for developers. What this means is that speakers aren't dictated what they can and cannot talk about and all of the speakers are encouraged to base their sessions on real world experience. There are real users in the case studies so you can see what your colleagues at other organizations are doing. It also means that the marketing fluff stops at the door – so you can learn more from other developers!

**[www.cfunitied.com](http://www.cfunitied.com)**

"I attended CFUNITED along with a number of my colleagues from the university. Some were experienced developers, some were novices, some were IT managers or server admins. The consistent thread of all our post-conference conversations regarding CFUNITED was the number of ideas we came away with. Whether it was a new solution to an old problem or just having the possibilities of ColdFusion opened up before us, we all felt it was well worth the trip. Of course our blood sugar levels after eating that CF birthday cake might have contributed some to that feeling as well :-)"

- Bob Flynn, Indiana University MMUG

Accessibility/Usability - Section 508, CSS and disabled access

Advanced CF - Advanced ColdFusion topics

Boot Camp - Basic ColdFusion and Flash Topics

Deployment/Platform - Integrating with SQL Server, Windows and .NET

Flex - Learn Flex 2 by experts at Universal Mind & Adobe

Manager/Emp Programming - Fusebox and Project management topics



Charlie Arehart  
New Atlanta



John Paul Ashenfelter  
Transitionpoint



Selene Bainum  
INPUT



Simeon Bateman  
Simb Development



Thomas Burleson  
Universal Mind



Raymond Camden  
Mindseye



Sandy Clark  
Constella Group



Sean Corfield  
Adobe Systems, Inc.



Michael Dinowitz  
House of Fusion



Nahuel Foronda  
Universal Mind



Ben Forta  
Adobe Systems, Inc.



Shlomy Gantz  
Bluebrick Inc.



Rob Gonda  
Editorial Chief of Ajax  
Developer's Journal



Hal Helms  
Hal Helms Inc.



Simon Horwith  
Editor-in-Chief, CF  
Developer's Journal



Jeffrey Houser  
DotComIt



Jeremy Kadlec  
Edgewood Solutions



Kurtis D Leatham  
KomputerMan.com



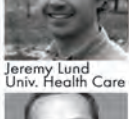
Adam Wayne Lehman  
US Department of State



Tom Link  
Universal Mind



Jeremy Lund  
Univ. Health Care



Nate Nelson  
Xententia, Inc.



Mike Nimer  
Adobe Systems Inc.



Maxim Porges  
CF/Westgate Resorts



Steve Rittler  
CounterMarch Systems



Andrew Schwab  
EXP Software /  
FusionDox



Michael Smith  
TeraTech



Matt Stevanus  
Universal Mind



Jeff Tapper  
Tapper.net Consulting



Kelly Tetterton  
Duo Consulting



Glenda Vigoreaux  
GVX Technology



Douglas Ward  
The Centech Group



Dave Watts  
Fingleaf Software

**8th Annual Event**  
**Over 1,000 attendees**

**Pre-Conference Classes: June 26th & 27th**

**Main Event: June 28th - July 1st**

**Washington DC Area**



**Adobe**



# Creating Object-Oriented Presentation Layers

**All the coolest programmers are doing it**



By Hal Helms

For the past several weeks, I've been immersed in writing a large application – so immersed, in fact, that I missed writing my column for last month! This application has been particularly interesting to me

because it makes such extensive use of AJAX – that combination of JavaScript, DOM manipulation, and the XMLHttpRequest object that has caught the attention of the general public. It's also been interesting because it has allowed me to experiment with the idea of creating an object-oriented (OO) presentation layer.

Why incorporate OO? Because all the coolest programmers are doing it! In fact, it seems to me that the accepted wisdom is rapidly becoming this: the only systems worth developing are OO systems and the only programmers worth their salt are OO programmers.

This puts great pressure on procedural programmers to “become” OO programmers. The problem is that designing OO systems is really hard. It's one thing to learn that polymorphism, encapsulation, and inheritance are key features of an OO language; it's quite another to fully grasp the design ramifications of the OO mindset. Perhaps it's time to reflect on Daniel Boorstin's observation that “the greatest obstacle to discovery is not ignorance – it is the illusion of knowledge.” What exactly does it mean to be an OO programmer?

One of the key principles of object orientation is that an object is a collection of services. The data an object stores (as instance variables) are meant to be used by that object in carry-

ing out requests for these services. Treating an object as a service provider rather than a data provider is one of the hardest adjustments for procedural programmers to make, so hard, in fact, that many never make the transition.

We might call this data-provider approach to objects pseudo-object orientation. To the pseudo-OO programmer, an object is first and foremost a holder of data. When pseudo-OOers attempt design, they do so by asking what properties a class has. Knowing that many real OO programmers use UML, pseudo-OOers often follow suit, but their pseudo-OO designs can be distinguished from the real thing at a glance: they're heavily weighted on expressing data and the services they expose (as methods) are primarily getters and setters, with some data validation added.

Let me give you an example. Suppose we are writing an e-commerce system that needs to account for taxes. Let's say that in our system, different products are taxed at different rates depending on their country of origin, while some products are entirely tax-free. Set a pseudo-OO programmer to work on this, and he'll likely come up with a UML class diagram for a product that looks something like Figure 1.

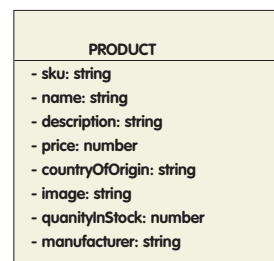


Figure 1

This analysis of a product fits nicely with what we might call the pseudo-OO motto: All Data, All the Time. The bottom portion of the class box, which should show the services the class can provide, is conveniently empty.



To the pseudo-OO coder, a class is really just a structure in which to hold data. Most often, an instance of a class (an object) is nothing more than an in-memory representation of a row in a database. Given this view, it's no surprise that our pseudo-OOer will immediately concern himself with synchronizing the in-memory representation of the row with the actual row in the database. Enter the data access object or DAO (that works with a single object) and its close cousin, the gateway (that works with collections of objects).

Of course, no pseudo-e-commerce system would be of much use without a pseudo-shopping cart. Stated slightly differently (but with almost no change in meaning), no data-centric e-commerce system would be of much use without a data-centric shopping cart, maybe something like Figure 2.

ShoppingCart
- products: array

Figure 2

The products array may be a two-dimensional array in which one index holds the Product object, another holds the quantity in the cart, and perhaps a third holds the SKU of the product (for quick lookups). When it's time to display the shopping cart, the code might look something like this:

```
<table class="cart">
```

```
<tr>
    <th>Quantity</th>
    <th>Product</th>
    <th>Description</th>
    <th>Price</th>
    <th>Extension</th>
</tr>
<!-- get array of products in cart -->
<cfset products = session.user.cart.getProducts()>
<cfset subtotal = 0>
<cfset taxes = 0>
<cfloop from="1" to="#ArrayLen(products)#" index="i">
<tr>
    <!-- quantity -->
    <td>#products[i][2]#</td>
    <!-- name -->
    <td>#products[i][1].getName()#</td>
    <!-- description -->
    <td>#products[i][1].getDescription()#</td>
    <!-- price -->
    <td>#DollarFormat(products[i][1].getPrice())#</td>
    <!-- extension -->
    <td>#DollarFormat(products[i][2] * products[i][1].get-
Price())#</td>
</tr>
<cfset subtotal = subtotal + products[i][2] * products[i][1].get-
```

## Efficient Web Content Management

**CommonSpot™ Content Server** is the ColdFusion developer's leading choice for efficient Web content management.

Our rich Web publishing framework empowers developers with out-of-the-box features such as template-driven pages, custom content objects, granular security, and powerful ColdFusion integration hooks (just to name a few), allowing you to rapidly build and efficiently deploy dynamic, high performance Web sites.

CommonSpot's open architecture and extensive APIs enable easy customization and integration of external applications. With CommonSpot, you can design and build exactly what you need. Best of all, CommonSpot puts content management in the hands of content owners. With non-technical users responsible for creating and managing Web content, developers are freed to focus on strategic application development.

## Evaluate CommonSpot today.

To see your site *running* under CommonSpot, call us at 1.800.940.3087.



fast  
easy  
affordable

## features.

- 100 % browser-based
- Content object architecture
- Template driven pages
- 50+ standard elements
- Content reuse
- Content scheduling
- Personalization
- Flexible workflow
- Granular security
- Mac-based authoring
- 508 compliance
- Full CSS support
- Custom metadata
- Taxonomy module
- Extensible via ColdFusion
- Web Services content API
- Custom authentication
- Replication
- Static site generation
- Multilanguage support

1.800.940.3087  
www.paperthin.com

Paper | Thin

© Copyright 2005 PaperThin, Inc. All rights reserved.

```

Price()>
    <cfset taxes = taxes + hmmm>
    </cfloop>

    <tr>
        <td colspan="5" align="right">Sub-total: #DollarFormat(subtotal)
    </td>
    </tr>

    <tr>
        <td colspan="5" align="right">Total: #DollarFormat(subtotal + hmmm)
    </td>
    </tr>
</table>

```

Notice that I've omitted all the logic for computing the taxes, replacing it with a generic "hmmm" variable. Of course, that won't really work, so perhaps our programmer will create a `computeTax(countryOfOrigin)` function that accepts a country of origin (a string) and returns the tax on the item. Add in a few DAOs and Gateway objects and you have yourself an object-oriented e-commerce system.

Except that it isn't. All that's happened is that an e-commerce system, solidly procedural to its core, has been tinkered with and baptized as an OO system. Instead of objects providing services, we have objects providing data that is then used for such tasks as calculating the subtotal, taxes, and total of the shopping cart.

What would a real OO system look like? We would first determine what we need a cart to do – what messages it should respond to, e.g., what services it should provide. How the shopping cart responds to these messages must, of course, be determined, but not now. Our first and most important job is to determine the entities needed by our system and the methods

ShoppingCart
<ul style="list-style-type: none"> <li>+ <code>addProduct(Product product): void</code></li> <li>+ <code>removeProduct(Product product): void</code></li> <li>+ <code>clearCart(): void</code></li> <li>+ <code>getSubtotal(): numeric</code></li> <li>+ <code>getTaxes(): numeric</code></li> <li>+ <code>getTotal(): numeric</code></li> </ul>

Figure 3

each entity will offer. These methods – or, at least, the publicly accessible ones – form an application programming interface (API) for each entity. The API informs us of the appropriate messages that can be sent to any object. When done correctly, an OO program resembles nothing so much as a conversation between

objects.

With this in mind, let's take another look at an OO design for a shopping cart. We start not with data, but with services to be provided. What do we want our cart to do? The answer to this will form the cart's API. Figure 3 shows my take on a Shopping-Cart class.

Now, all the focus is on services (methods), not on the data. I've gone so far in slighting the data as to completely ignore it, and this is exactly the right level to design at. OO programs are


not about data and functions; they are about determining the way that objects will communicate with each other. Our job as OO designers is to provide an environment in which safe, meaningful communication between objects can take place.

If we were working in the Java world, this emphasis on finding the right API would be reflected in the use of a Java construct called an interface, a fundamental entity in which only methods can be specified. Good design is done at the interface level. Only when the interfaces and their APIs have been determined is the class design (now including data) created.

We don't have interfaces in ColdFusion (nor am I anxious that they should be included in the language), but the essence of interfaces – the focus on services rather than data

– should be just as central to our designs in ColdFusion as it would be if we were designing in Java.

Once we have a class design done, of course, we must make the objects created from these classes persistent. DAOs and gateways are one way to do this, but we must make sure that they are subservient to an OO design that allows for communication between objects.

Next month, I'll share some experiments I've been working on that try to bring the ideas of object orientation to the presentation layer. I think the experiments may hold some promise, not in order to be fashionable, but to create programs that at every level (including the presentation level) are easy to maintain and adapt easily to change. Those are goals that will resonate with programmers of all types. 

## About the Author

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

[hal@halhelms.com](mailto:hal@halhelms.com)



# Welcome to the Future of Video on the Web!

REGISTER NOW!  
[www.iTVcon.com](http://www.iTVcon.com)

CALL FOR PAPERS NOW OPEN!

 **LIVE SIMULCAST!**  
AROUND THE WORLD ON [SYS-CON.TV](http://SYS-CON.TV)

**iTV CON.COM**  
INTERNET TV CONFERENCE & EXPO 2006

Coming in 2006 to New York City!

“Internet TV is wide open, it is global, and in true ‘Web 2.0’ spirit it is a direct-to-consumer opportunity!”



For More Information, Call 201-802-3023  
or Email [itvcon@sys-con.com](mailto:itvcon@sys-con.com)

## Welcome to the Future!

Did you already purchase your “.tv” domain name?

You can't afford not to add Internet TV to your Website in 2006!

2005 was the year of streaming video and the birth of **Internet TV**, the long-awaited convergence of television and the Internet. Now that broadband is available to more than 100 million households worldwide, every corporate Website and every media company must now provide video content to remain competitive, not to mention live and interactive video Webinars and on-demand Webcasts.

20 years ago the advent of desktop publishing tools opened the doors for the creation of some of today's well-known traditional print media companies as well as revolutionized corporate print communications. Today, with maturing digital video production, the advent of fully featured PVRs, and significant advances in streaming video technologies, **Internet TV** is here to stay and grow and will be a critical part of every Website and every business in the years to come.

It will also very rapidly become a huge challenge to network and cable television stations: **Internet TV** is about to change forever the \$300BN television industry, too.

The Internet killed most of print media (even though many publishers don't realize it yet), Google killed traditional advertising models, and **Internet TV** will revolutionize television the way we watch it today. You need to be part of this change!

**Jeremy Geelan**  
Conference Chair, [iTVCon.com](http://iTVCon.com)  
[jeremy@sys-con.com](mailto:jeremy@sys-con.com)

PRODUCED BY  
**SYS-CON**  
EVENTS

### List of Topics:

- > Advertising Models for Video-on-demand (VOD)
- > Internet TV Commercials
- > Mastering Adobe Flash Video
- > How to Harness Open Media Formats (DVB, etc)
- > Multicasting
- > Extending Internet TV to Windows CE-based Devices
- > Live Polling During Webcasts
- > Video Press Releases
- > Pay-Per-View
- > Screencasting
- > Video Search & Search Optimization
- > Syndication of Video Assets
- > V-Blogs & Videoblogging
- > Choosing Your PVR
- > Product Placement in Video Content
- > UK Perspective: BBC's "Dirac Project"
- > Case Study: SuperSun, Hong Kong

- |                |  |
|----------------|--|
| <b>Track 1</b> | Corporate marketing, advertising, product and brand managers   |
| <b>Track 2</b> | Software programmers, developers, Website owners and operators   |
| <b>Track 3</b> | Advertising agencies, advertisers and video content producers  |
| <b>Track 4</b> | Print and online content providers, representatives from traditional media companies, print and online magazine and newspaper publishers, network and cable television business managers |



# When A Happens Old Bro



By Jeremy Lund

One of the latest crazes in Web development is AJAX. Unless you've been living in a cave for the last year, you've heard of this old, yet currently popular, technique for making HTTP requests to a server

without refreshing the Web page. While claims of smaller bandwidth, faster response, and highly interactive user interfaces may intrigue you, one must ask, "Will it work for my user base?"

We've seen this same issue before. It's not an AJAX-specific issue; rather, it's a JavaScript issue. There are users that either by choice or mandate are stuck with an ancient Web browser. There are others who still don't have JavaScript enabled on their browsers. It's difficult to know exactly how many people don't have JavaScript turned on, but estimates range from 5%-10%.

Is 5%-10% a significant user population? It depends on who you ask. Some businesses can take a chance of alienating this user base at the prospect of wowing the remaining 90%, while for others, it's either necessary or desirable to accommodate all

users. It seems that you may be left with only two options when deciding whether to implement AJAX on your site:

- Leave the users of older browsers behind.
- Accommodate this small user base at the expense of lesser functionality for the remaining 90%.

I'd like to propose a third option:

- Accommodate both groups through effective planning and design.

"Accommodate both groups," you ask. "Doesn't that mean that I need to develop two separate Web applications?" With careful planning and design, you can develop a single application that provides essential functionality to all users, while providing enhanced functionality to those whose browsers can handle it.

## "Essential" versus "Enhanced" Functionality

Essential functionality means providing a user with the ability to complete a task or use a service. For instance, suppose one of the services on your Web site is a weekly newsletter. To subscribe to the newsletter, a user must complete a registration form. What happens when the user clicks the submit button at the end of the form? Few things are more frustrating than completing a large form only to have the submit button "shoot blanks" because it requires JavaScript to function. Essential

# AJAX to Browsers

**Building a single  
application that  
supports both new  
and old browsers**

functionality dictates that the submit button works regardless of whether the user is using Firefox, Internet Explorer, Safari, Lynx, or a screen reader.

Enhanced functionality means adding all of the bells and whistles that really set your application apart. While such features aren't essential, adding them to an application usually provides faster ways to accomplish tasks or easier ways of doing things. Enhanced functionality includes such things as:

- client-side form validation
- dynamic HTML effects
- AJAX

The strategy that I suggest for accommodating both essential and enhanced functionality is to add "bells and whistles" to a basic application. That is, we'll create an application that satisfies the essential functionality and then provide a way for enhanced functionality to be attached to the application only if the client's browser will support it.

## The Sample Application: A Contact Manager

The sample application accompanying this article is a simple contact manager (see Figure 1). The goal of this application is to provide users with the ability to:

- Add new contacts
- Edit existing contacts
- View all contacts
- Delete contacts

First we'll build the application that addresses essential functionality. In the basic application, all GET and POST operations will be full-page refreshes. Later, we will add some AJAX functionality to the application. To keep the example simple, it stores all contacts in a query object in the application scope.

## Coding the Basic Application

The basic application is in the /ajax/no-js/ directory of the code samples. One of the first things that I did when building the basic application was to separate queries, content sections, and controlling logic into separate templates. This is something that can be done now to prepare the application to work for both basic and advanced browsers. AJAX calls and other dynamic effects often need small segments of content, or data in XML, rather than HTML. Instead of duplicating code, we write each query and display template and include them in potentially multiple places.

Most of the business logic, or model layer, is in contactManager.cfc. It contains all of the operations needed to read and manipulate contacts. The contact listing table, contact entry and edit form, and delete confirmation page are all in separate templates. layout.cfm is a custom tag that contains the basic HTML skeleton to wrap the other content sections. Finally, index.cfm handles all of the possible actions from the browser in a <cfswitch> tag. Most of the action handlers look like the following saveRecord case:

```
<cfcase value="saveRecord">
<!---put the contact information in a struct --->
<cfinvoke returnvariable="contact" component="#request.manager#"
method="populateContact" argumentcollection="#form#"/>
<!--- store the contact in the database --->
<cfset request.manager.saveContact(contact)/>
<!--- retrieve an updated list of contacts --->
<cfset qContactListing = request.manager.getAll()/>
<!--- create an empty contact for the entry form --->
<cfset contact = request.manager.populateContact()/>
<cf_layout title="Contact Manager" message="Contact was successfully
saved.">
<cfinclude template="dsplisting.cfm"/>
```

	Name	Phone Number
<a href="#">edit</a> <a href="#">delete</a>	Cox, Stan	unlisted
<a href="#">edit</a> <a href="#">delete</a>	D'oh, John	555-1234
<a href="#">edit</a> <a href="#">delete</a>	Jenny, Jenny	867-5309

Last Name:	<input type="text"/>
First Name:	<input type="text"/>
Phone Number:	<input type="text"/>
<input type="button" value="Save"/>	

Figure 1



```
<cfinclude template="dspEntryForm.cfm"/>
</cf_layout>
</cfcase>
```

Each action handler case calls the business methods and includes the templates necessary to perform the action.

Finally, I made sure that the markup was clean and simple, and that all visual styling was placed in a stylesheet. Clean markup provides three benefits, regardless of whether the application will be using AJAX:

1. It's easier to work with the HTML structure using JavaScript DOM APIs, and stylesheets.
2. It usually means less data is sent back to the client at one time, since unnecessary elements and attributes are eliminated. When we send HTML segments to the client using AJAX, those responses will be even smaller.
3. The HTML is often easier to read and maintain.

Try out the sample application so you understand how it works. At this point, the basic application works reliably and provides essential functionality. Because, it's not very exciting, it's time to add some enhanced functionality.

## Grafting In the Enhancements

The enhanced application is in the /ajax/with-js/ directory of the code samples. In this version of the application, we're going to add some AJAX calls. This means replacing any of the pages exit points — the edit links, delete links, and save button — with JavaScript calls to actions in the ColdFusion application. A common way to do this is to make the necessary onclick event handler an attribute on the anchor tag or submit button like so:

```
<a href="index.cfm?action=editRecord..." onClick="lookupContact();">edit</a>
```

Rather than cluttering up the HTML with event handlers, I prefer to attach them to the elements using JavaScript calls in the header. The calls are usually invoked as part of the browser window's onload event. This keeps the HTML content and JavaScript code clearly separated. To make it easy to attach events using this method, let's make two quick changes to the code:

1. Add class attributes to the edit and delete links, and an id attribute to the submit button. This will make it easier to find the elements using JavaScript.
2. Move the HTML contact table to a separate template (dspListingContent.cfm) and include it in the original template using <cfinclude/>. The HTML table is a key page fragment that we'll want to return to the client using AJAX.

## DOM and AJAX the jQuery Way

While it might be tempting to write the JavaScript code necessary to attach events handlers to HTML elements to perform AJAX calls and manipulate the HTML document, there are plenty of JavaScript libraries available to use. For this application, I've chosen to use the jQuery library (<http://jquery.com>). jQuery includes an excellent interface for selecting elements us-

ing CSS rules. For instance, if I wanted to change all of the links in the contact table to red, all I have to write is the statement:

```
$("#contacts a").css("color","red");
```

I'll use this method of element selection to attach event handlers to the edit links, delete links, and submit button. The code to change the edit link is:

```
$('#contacts tbody a.edit').click(
function() {
$.getXML(this.href + "&isAjax=" + new Date().getTime(),
function(xml) {
var contact = new Contact();
contact.populateFromXml(xml);
contact.sendToForm($("#entryForm"));
});
return false;});
```

This adds an onclick handler to each edit link, taking advantage of the "edit" class that we added to the anchor tag. The function also returns false so the browser doesn't call the page in the href attribute of the link after handling the onclick event. You may notice that I've added something extra to the HTTP URL for the call. Instead of just using this.href I've added an isAjax attribute, with its value being the number of milliseconds since January 1, 1970. These additions address two issues:

1. I needed some way to indicate to the ColdFusion application whether this is an AJAX request.
2. Internet Explorer caches responses that have the same URL, even AJAX responses. By adding the current time, in milliseconds, to the URL, it effectively makes every request unique.

The next change is to our controller, index.cfm. Using the new isAjax attribute, the application determines whether the regular controller should handle the request or hand it off to a new set of handlers stored in ajaxActions.cfm. ajaxActions.cfm looks very similar to index.cfm, but the actions have been modified to return only the pieces of information that are needed to complete the request.

What do the AJAX calls do with the responses that they get? It depends on the action, but in the case of the editRecord action, it takes the XML that's returned from the application and populates the appropriate entry form fields. Other actions populate the message area with an action-appropriate message. After reviewing the application, try out the functionality. One of the things that you'll notice is that deleting a contact no longer requires a "Confirm Delete" page; rather, it uses the JavaScript confirm() dialog to do the same thing.

## Enhanced Application, Unplugged

What do you do once you've finished adding all of this nice functionality to the application? Turn it off, of course!

The ultimate test of the enhanced application is whether it will work when the JavaScript won't. Turn off the JavaScript in

your browser to see how the application behaves. Everything should work identically to how it worked in the basic application, which is exactly what we want it to do.

### Summary

It's possible to use JavaScript and AJAX in your Web applications without abandoning a portion of your user base. With some careful planning when designing the application, it's easy to build only one application that supports both new and old browsers. 

### About the Author

Jeremy Lund is the manager of the Web Resource Center at University Health Care in Salt Lake City, Utah (<http://uuhs.utah.edu/wrc/>). As the manager he coordinates development efforts, designs their Web architecture, and, in the time left, develops Web applications. He has worked with Internet technologies for over seven years, and enjoys working not only with ColdFusion, Java, and Perl, but is also an advocate of using XHTML and CSS. Jeremy received a bachelor's degree in computer engineering from the University of Utah, and is a Sun Certified Programmer for the Java 2 platform.

[Jeremy.Lund@hsc.utah.edu](mailto:Jeremy.Lund@hsc.utah.edu)

## Get more with Hot Banana!

Web Content Management + Active Marketing



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

**Call us today at**  
1-866-296-1803 to schedule a live 1-on-1 demo, or visit [hotbanana.com/cfdj](http://hotbanana.com/cfdj) to learn more.



# REPRINT IT!

## Once you're in it...



## ...reprint it!

- ColdFusion Developer's Journal
- Java Developer's Journal
- Web Services Developer's Journal
- Wireless Business & Technology
- XML Journal
- PowerBuilder Developer's Journal
- .NET Developer's Journal

Contact Dorothy Gil  
201 802-3024  
[dorothy@sys-con.com](mailto:dorothy@sys-con.com)

REprints

 SYS-CON MEDIA



# ***Rich Internet Applications: AJAX,***

[www.AjaxWorldExpo.com](http://www.AjaxWorldExpo.com)

# AJAXWORLD™ CONFERENCE & EXPO

## SAN JOSE SILICON VALLEY

### **SYS-CON Events is proud to announce the first-ever AjaxWorld Conference & Expo 2006!**

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

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

#### **CALL FOR PAPERS NOW OPEN!**

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

# Flash, Web 2.0 and Beyond...

REGISTER TODAY AND SAVE!

→ **October 3-4, 2006**

→ **Santa Clara Convention Center**  
Hyatt Regency Silicon Valley  
Santa Clara, CA

→ **To Register**  
Call 201-802-3020 or  
Visit [www.AjaxWorldExpo.com](http://www.AjaxWorldExpo.com)

→ **May 7-8, 2007**  
First International AjaxWorld Europe  
Amsterdam, Netherlands

**“Over the two information-packed days, delegates will receive four days’ worth of education!”**

## Early Bird\*

(Register Before August 31, 2006)  
..... **\$1,495\*\***  
See website or call for group discounts

## Special Discounts\*

(Register a Second Person)  
..... **\$1,395\*\***  
See website or call for group discounts

(5 Delegates from same Company)  
..... **\$1,295/ea.\*\***  
See website or call for group discounts

## On-Demand Online Access

(Any Event)  
..... **\$695**

\*Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

\*\*OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

**“It Was The Best AJAX Education Opportunity Anywhere in the World!”** —John Hamilton

## Topics Include...

### Themes:

- > Improving Web-based Customer
- > Interaction
- > AJAX for the Enterprise
- > RIA Best Practices
- > Web 2.0 – Why Does It Matter?
- > Emerging Standards
- > Open Source RIA Libraries
- > Leveraging Streaming Video

### Technologies:

- > AJAX
- > The Flash Platform
- > The Flex 2 Framework & Flex Builder 2
- > Microsoft’s approaches: ASP.NET, Atlas, XAML with Avalon
- > JackBe, openLaszlo
- > JavaServer Faces and AJAX
- > Nexaweb
- > TIBCO General Interface

### Verticals:

- > Education
- > Transport
- > Retail
- > Entertainment
- > Financial Sector
- > Homeland Security

**GROUP DISCOUNTS AVAILABLE:**  
— 5 Delegates from Same Company —  
for only \$995 (each)  
— Register a Second Person —  
for only \$1195



**Hurry! Limited Seating  
This Conference Will Sell-Out!**



**LIVE SIMULCAST!**  
AROUND THE WORLD ON SYS-CON TV

Receive **FREE**  
WebCast Archives  
of Entire Conference!

The best news for this year’s conference delegates is that your “Golden Pass” registration now gives you full access to all conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.



► This on-demand archives set is sold separately for \$995



For more great events visit [www.EVENTS.SYS-CON.com](http://www.EVENTS.SYS-CON.com)

VISIT [WWW.AJAXWORLDEXPO.COM](http://WWW.AJAXWORLDEXPO.COM) FOR THE MOST COMPLETE UP-TO-DATE INFORMATION



# Feed Your Site

## Incorporating RSS and Atom feeds into your ColdFusion Site



By Jeremy Lund

**C**ode reuse has always been an important principle in a software developer's toolbox.

While this principle often applies to source code, developers have also tried to apply the same formula to Web content.

By grabbing content from one Web site, the developer can reuse it in new and unique ways.

Traditionally, reusing content meant writing screen scrapers that could sift through a heavy table-filled HTML page to extract the few data elements of interest. If the source page's layout changed, the extraction code would usually have to be rewritten too.

All this changed when Web syndication feeds were introduced several years ago. A standard file format, such as RSS, meant that the same tools could be reused for many different sets of data. A Web designer could also rebuild his site at any time without worrying that someone else's extraction script would no longer function. While Web syndication files have been around for quite some time, the last couple of years they've really taken off. With so many news feeds available, now is an excellent time to take advantage of their power. The purpose of this article is to address how to use ColdFusion to consume Web syndication feeds, as well as how to generate your own news feeds.

### What Are RSS and Atom?

RSS stands for Rich-Site Summary or Really Simple Syndication depending on who you ask. It's a file format used in Web syndication and defined using XML. A RSS file contains metadata about the news feed or channel elements and a set of news items. Each news item usually contains the item's title, a brief summary of the news item, and a link to the full news story. Since RSS is nothing more than an XML file, there's a rich supply

of tools and APIs that can both generate and read RSS feeds. Atom is another Web syndication file format, with a primary purpose of addressing some of the limitations of the RSS 2.0 specification. The Atom format is different enough from RSS 2.0 that great care must be taken when trying to build a reader that consumes all kinds of feed formats. Thankfully, the file formats are comparable enough that the same parser can be used for most common uses.

You can read, or consume, a feed with one of the many freely available programs, but what's the fun in that? Let's find out just how easy it is to use ColdFusion to read news feeds.

### How to Read a Feed

As mentioned, one of the issues with trying to consume a feed is the differences in file formats among the different feed specifications. Because of the differences in both structure and element names between the different feeds, writing a feed parser can be difficult. Fortunately, someone has already done the work for you.

Roger Benningfield's RssAtom ColdFusion Component (CFC) will normalize the content of RSS 0.91, RSS 1.0, RSS 2.0, Atom 0.3, and Atom 1.0 feeds into a feed-neutral data structure. The primary function in this CFC, `normalize`, returns a structure. One element in the structure, `feed`, contains header elements and their values, while a second element, `items`, contains an array of items and their content.

A dump of this structure looks like Figure 1. This structure is much easier to work with than trying to account for the feed differences yourself. To run each of the examples accompanying this article, you'll have to get the latest version of `RssAtom.cfc` from Roger's site (<http://mxblogspace.journurl.com/users/admin/>).

`example1.cfm` reads one of the aggregated feeds from `ful-lasagoog.com` using the `<cfhttp>` tag and normalizes the feed using `RssAtom.cfc`. Finally, it displays some of the header information and iterates through each news item, showing title, link, and content for each item. Figure 2 has an example of what the results look like. The page's code weighs in at only 27 lines, but what if we wanted to include several feeds on one Web page? Because the repetitive code quickly adds up, let's create a custom tag to encapsulate the feed consumption and display code.

## Building a Custom Tag

Moving the existing code into a custom tag is easy, but let's add some additional functionality while we're at it. Let's also allow our tag user to specify the format in which he'd like to return

the feed results. rssdisplay.cfm contains the code for this custom tag. A call to this custom tag looks like this:

```
<cf_rssdisplay url="{feed url}" file="{or absolute path to feed}">
```

```
<format>
<header><![CDATA[<h2><a
href="{link}">{title}</a></h2><ul>]]></header>
<items><![CDATA[<li><a
href="{link}">{title}</a></li>]]></items>
<footer><![CDATA[</ul>]]></footer>
</format>
</cf_rssdisplay>
```

To format the feed, the tag will display the header once, replacing the variables, indicated as \${variable} with data from the feed header. It will then do the same thing for each item, except that it will use the item's format rule and replace each variable with item data. Finally, it will run the footer once, doing any necessary variable replacement. example2.cfm uses the custom tag and results in the same output as example1.cfm. The code in example2.cfm is not only shorter, but makes us flexible in how we present the feed data.

## Putting the Reader on a Diet

The custom tag makes it much easier to add feeds to any ColdFusion template that we care to add them to, but it suffers from one big problem. Each time the tag is called, it has to access the remote feed. This slows down how quickly the tag runs on your site and uses up someone else's bandwidth needlessly. It doesn't have to consume the feed on every page access, particularly since the feed most likely doesn't change every time that we access it. Let's add a couple of new attributes to the rssdisplay.cfm custom tag that will let us specify when and how to cache a feed.

cachedWithin will accept a time span that specifies how often the cache should be refreshed. name will give the cached feed a unique name to identify the cached feed. That way, if we want to display the same feed two different ways, both can use the same cached copy of the feed. rssdisplay2.cfm contains the modifications to implement caching. The caching code adds a little over 50 lines of code to the tag, with most of the changes between lines 29 and 70. I opted to cache the feeds in the server scope instead of the application scope or session scope so the tag could be called on any page, not just one that's part of an application. It's a trivial matter to change the caching to the application scope, if desired.

struct	
FEED	
struct	
AUTHOR	modius
AUTHOREMAIL	[empty string]
AUTHORURL	[empty string]
CONTRIBUTOR	[empty string]
CONTRIBUTOREMAIL	[empty string]
CONTRIBUTORURL	[empty string]
DATE	{ts '2006-04-11 14:14:03'}
DATEUPDATED	[empty string]
DESCRIPTION	Fullasagoog: Chock Full Of Rich Internet Application Goodness
ID	http://fullasagoog.com/
JOURNURL	struct
	EXPORT: false
LINK	http://fullasagoog.com/
LINKSELF	[empty string]
TITLE	fullasagoog.com ColdFusionMX blend
ITEMS	
array	
1	struct
AUTHOR	Mark Kruger
AUTHOREMAIL	[empty string]
AUTHORURL	[empty string]
CONTENT	This post may be one that very few of my readers will care about. But if you are the 1 MQSeries version 6 using coldfusion then this post may prove a life saver. You can b
CONTENTTYPE	html
CONTRIBUTOR	[empty string]
CONTRIBUTOREMAIL	[empty string]
CONTRIBUTORURL	[empty string]

Figure 1 Data dump from RssAtom.cfm

## RSS: Example 1

### [fullasagoog.com ColdFusionMX blend](#)

#### Fullasagoog: Chock Full Of Rich Internet Application Goodness

#### [Lvla CAPTCHA: Neat!](#)

Peter J. Farrell has taken my open source CAPTCHA component to the next level - You should check it out&nbsp;It&apos;s highly configurable (way more than mine is), and comes with a slew of new features. His documentation is also better than mine. I&apos;m still going to keep my open source captcha on my site, but if you are looking for something a bit above and beyond what i wrote, definatley check out what...

#### [ActiveX content does not load automatically in Microsoft Internet Explorer \(Windows\)](#)

TechNote: When you view a web page, Microsoft Internet Explorer for Windows displays a box around ActiveX content and a tooltip which reads "Click to activate and use this control." ActiveX content may or may not play as expected.

Figure 2 Displaying feed as HTML



Now that we've added the ability to cache feeds via the custom tag, all we have to figure out is how often this particular feed we're consuming is updated. Surely, the feed doesn't change that often, but is there a way to find out how frequently it changes?

Many feeds will include some information about how often they're updated. We can use this information to determine how often to retrieve a feed. RSS 2.0 feeds have a "time to live" element, or ttl, that indicates in minutes how often the feed is updated. The particular feed that we've been working with is a RSS 1.0 feed that includes the Syndication module. Using the three elements that comprise

the Syndication module, updatePeriod, updateFrequency, and updateBase, we can figure out how often the feed is updated. This particular feed is updated every hour. example3.cfm uses the updated <cf\_rssdisplay> tag, and caches the feed once an hour. Notice that it's more responsive than example2.cfm. Caching will help make your site run faster and will respect the source feed's update instructions.

## Feeding Ourselves

If the news items that you're interested in aren't currently available, perhaps it's time for you to generate your own news feed. Then too, there might be other

sources of news items that you might not consider news. While news feeds are primarily used for news articles and blog entries, there are plenty of other news sources. If news is nothing more than a time-sensitive event or entry, then each entry in a Web server error log is a piece of news. One thing that would be useful is a news feed that presents the most recent records from the ColdFusion server's application.log file. That way, I know what kind of errors are occurring on my server without logging into the ColdFusion administrator.

example4.cfm builds a feed that displays the 20 most recent entries in the application.log file. I decided to use the RSS 2.0 format for the feed simply because it's easy to work with and I'm most familiar with it. First the code builds the root and header portion of the feed using ColdFusion's XML functions. It then reads the last 20 records in the file, splits them into fields by column, and creates a new item using the column data. Figure 3 shows what the resulting XML feed looks like in Firefox.

## Consuming Our New Feed

Now that we've got this new feed, let's use it to build a simple administrator dashboard. Besides our new application news feed we'll also present the feeds for ColdFusion and JRun TechNotes. That way we have known issues at our fingertips if we come across a weird error in the ColdFusion logs. example5.cfm contains the code to build this little dashboard. Notice that it is only 34 lines long because the RssAtom.cfc and the custom tag do most of the work. In these few lines, we're retrieving, caching, and displaying three separate feeds, each feed having its own style. The results are shown in Figure 4.

## Summary

RSS feeds are useful for both sharing and displaying information, and more and more Web sites are providing news feeds of their data. ColdFusion, along with an excellent ColdFusion component, provides us with the tools to consume these news feeds easily. RSS feeds provide yet another channel to communicate information, and I encourage you to find ways in your daily management and development tasks to take advantage of this simple yet, powerful XML file format.

This XML file does not appear to have any style information associated with it. The document tree is shown below:

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>Application Log</title>
    <description>Application Log for CF Instance</description>
    <language>en-us</language>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <ttl>15</ttl>
    <lastBuildDate>Tue, 11 Apr 2006 22:41:57 -0600</lastBuildDate>
  </channel>
  <item>
    <title>(Error)</title>
    <description>
      javax.xml.transform.TransformerConfigurationException:
      javax.xml.transform.TransformerException:
      javax.xml.transform.TransformerException: Prefix must resolve to a namespace:
      namespaceA [Transformer] object cannot be created that satisfies the configuration
      requested. This could be due to a failure in compiling the [XSL] text.
      javax.xml.transform.TransformerConfigurationException:
```

Figure 3 A ColdFusion application.log RSS feed

## ColdFusion Status Page

Application Log CF TechNotes JRun TechNotes

### Application Log

(Error) (ts 2006-04-11 02:10:07)

```
javax.xml.transform.TransformerConfigurationException:
javax.xml.transform.TransformerException: javax.xml.transform.TransformerException:
Prefix must resolve to a namespace: namespaceA [Transformer] object cannot be created
that satisfies the configuration requested. This could be due to a failure in compiling the
[XSL] text. javax.xml.transform.TransformerConfigurationException:
javax.xml.transform.TransformerException: javax.xml.transform.TransformerException:
Prefix must resolve to a namespace: namespaceA The specific sequence of files included
or processed is: C:\var\cfmxapp\wwwroot\dev.lund.net\feeds\transformit.cfm, line: 13
```

(Error) (ts 2006-04-11 02:11:12)

```
The specific sequence of files included or processed is:
C:\var\cfmxapp\wwwroot\dev.lund.net\feeds\transformit.cfm
```

(Error) (ts 2006-04-11 02:12:15)

```
javax.xml.transform.TransformerConfigurationException:
javax.xml.transform.TransformerException: javax.xml.transform.TransformerException:
Could not find function: node-setA [Transformer] object cannot be created that satisfies
the configuration requested. This could be due to a failure in compiling the [XSL] text
```

Figure 4

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



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

## Showing Commitment to the Community


— continued from page 7

The deep focus issues that we ran for the first eight or nine months after CFMX 7 was released were very well received, as were some of the other focus issues (CFPetmarket in particular). It's difficult having the right content for a deep focus issue every month, so that will not be the regular monthly format moving forward. Due to the response we did get though, we will run a deep focus issue from time to time when it makes sense to do so, and we intend to follow every major release of ColdFusion with a series of issues that get our readers up to speed with the ins and outs of what's new immediately following each.

Over the past few months, I've written from time to time about wanting to give *CFDJ* significant Flex focus as well as the attention to CF. At one point I even went so far as to say the magazine might even be 50% CF and 50% Flex. Reader response to this was mixed – enough so that we've decided that *CFDJ* will indeed stay firmly rooted as the best printed resource for ColdFusion developers. Next issue will focus on Flex, since it is an important new product for CF developers, but after that Flex will receive occasional coverage, most likely closer to one article per month, based on our decision to move to a "repeated column" format. Our goal is to begin reaching out to meet the community's needs that, though often times silent, are certainly

legitimate.

The idea behind the repeated column format is that, in addition to special features on a variety of topics each month, *CFDJ* will run more regular monthly columns and multi-part articles. We will continue to offer the regular monthly columns already offered and will add to that. I can't get into specifics, but our goal is to achieve monthly blog coverage, CF Server optimization and troubleshooting advice, Q&A with a resident expert, Web 2.0 topics (Flex 2.0, Spry, and many other technologies), more attention to OOP and framework topics, and more tips/tricks/best practices articles pertaining to code and IDEs... each in a dedicated monthly column. Like I said, from time to time when appropriate, we'll diverge from this format and reintroduce the deep focus format in order to deliver intense coverage of special topics, but expect a more regular and more rounded issue each month going forward.

Expect to see this new format implemented over the next several months. I hope that you will find the new format and focus both informative and enjoyable. As always, now more than ever in fact, I am counting on our readers for feedback, advice, and of course for article submissions. You can e-mail me at [simon@horwith.com](mailto:simon@horwith.com). 

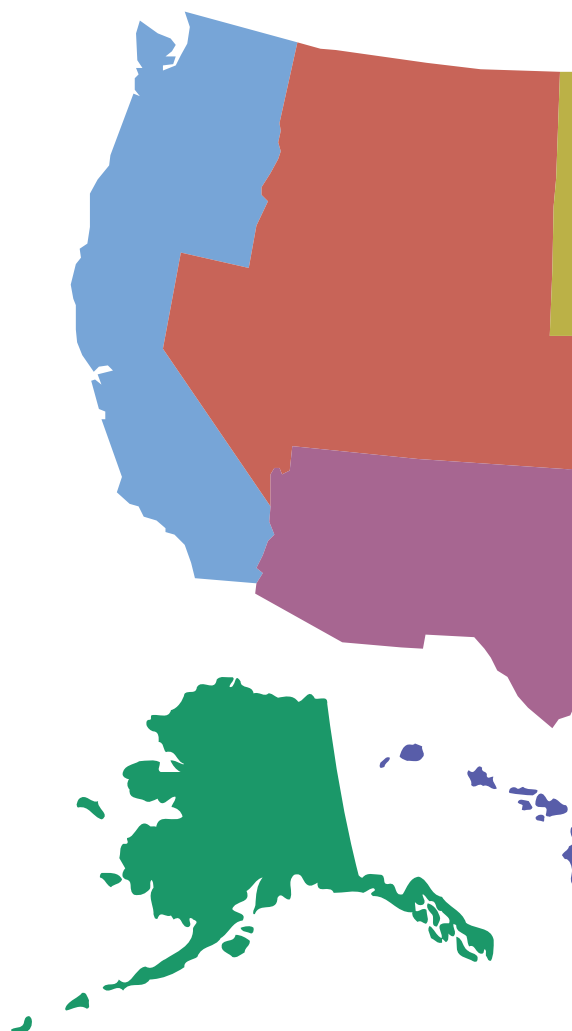


# ColdFusion

For more information go to...

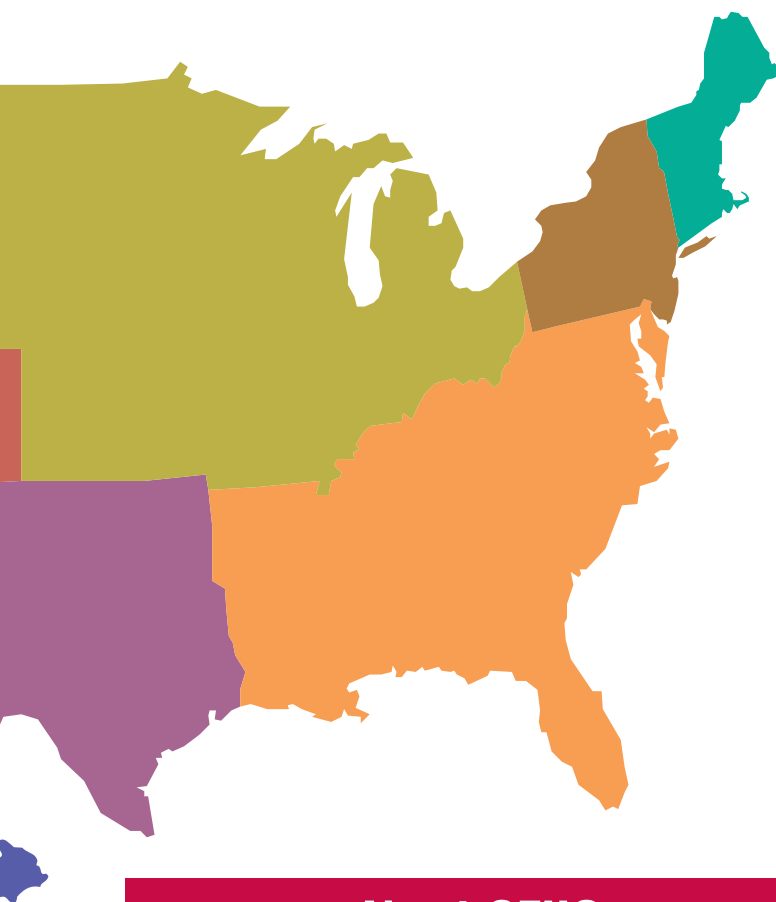
## U.S.

<b>Alabama</b> Huntsville, AL CFUG <a href="http://www.nacfulg.com">www.nacfulg.com</a>	<b>Louisiana</b> Lafayette, LA MMUG <a href="http://www.acadianammug.org/">http://www.acadianammug.org/</a>	<b>New York</b> Albany, NY CFUG <a href="http://www.anycfug.org">www.anycfug.org</a>
<b>Arizona</b> Phoenix, AZ CFUG <a href="http://www.azcfug.com">www.azcfug.com</a>	<b>Maryland</b> California, MD CFUG <a href="http://www.smdcfug.org">http://www.smdcfug.org</a>	<b>New York</b> New York, NY CFUG <a href="http://www.nycfug.org">www.nycfug.org</a>
<b>California</b> Bay Area CFUG <a href="http://www.bacfulg.net">www.bacfulg.net</a>	<b>Maryland</b> Maryland CFUG <a href="http://www.cfug-md.org">www.cfug-md.org</a>	<b>New York</b> Syracuse, NY CFUG <a href="http://www.cfugcny.org">www.cfugcny.org</a>
<b>California</b> Sacramento, CA CFUG <a href="http://www.sacfulg.com/">http://www.sacfulg.com/</a>	<b>Massachusetts</b> Boston CFUG <a href="http://bostoncfug.org/">http://bostoncfug.org/</a>	<b>North Carolina</b> Raleigh, NC CFUG <a href="http://tacfulg.org/">http://tacfulg.org/</a>
<b>California</b> San Diego, CA CFUG <a href="http://www.sdcfulg.org/">www.sdcfulg.org/</a>	<b>Massachusetts</b> Online CFUG <a href="http://coldfusion.meetup.com/17/">http://coldfusion.meetup.com/17/</a>	<b>Ohio</b> Cleveland CFUG <a href="http://www.clevelandcfug.org">http://www.clevelandcfug.org</a>
<b>Colorado</b> Denver CFUG <a href="http://www.denvercfug.org/">http://www.denvercfug.org/</a>	<b>Michigan</b> Detroit CFUG <a href="http://www.detcfulg.org/">http://www.detcfulg.org/</a>	<b>Oregon</b> Portland, OR CFUG <a href="http://www.pdxcfug.org">www.pdxcfug.org</a>
<b>Connecticut</b> SW CT CFUG <a href="http://www.cfugitives.com/">http://www.cfugitives.com/</a>	<b>Michigan</b> Mid Michigan CFUG <a href="http://www.coldfusion.org/pages/index.cfm">www.coldfusion.org/pages/index.cfm</a>	<b>Pennsylvania</b> Central Penn CFUG <a href="http://www.centralpenncfug.org">www.centralpenncfug.org</a>
<b>Connecticut</b> Hartford CFUG <a href="http://www.ctmug.com/">http://www.ctmug.com/</a>	<b>Minnesota</b> Southeastern MN CFUG <a href="http://www.bittercoldfusion.com">http://www.bittercoldfusion.com</a>	<b>Pennsylvania</b> Philadelphia, PA CFUG <a href="http://www.phillycfug.org/">http://www.phillycfug.org/</a>
<b>Delaware</b> Wilmington CFUG <a href="http://www.bvcfug.org/">http://www.bvcfug.org/</a>	<b>Minnesota</b> Twin Cities CFUG <a href="http://www.colderfusion.com">www.colderfusion.com</a>	<b>Pennsylvania</b> State College, PA CFUG <a href="http://www.mmug-sc.org/">www.mmug-sc.org/</a>
<b>Florida</b> Jacksonville CFUG <a href="http://www.jaxcfug.org/">http://www.jaxcfug.org/</a>	<b>Missouri</b> Kansas City, MO CFUG <a href="http://www.kcfug.org">www.kcfug.org</a>	<b>Tennessee</b> Nashville, TN CFUG <a href="http://www.ncfulg.com">http://www.ncfulg.com</a>
<b>Florida</b> South Florida CFUG <a href="http://www.cfug-sfl.org">www.cfug-sfl.org</a>	<b>Nebraska</b> Omaha, NE CFUG <a href="http://www.necfulg.com">www.necfulg.com</a>	<b>Tennessee</b> Memphis, TN CFUG <a href="http://mmug.mind-over-data.com">http://mmug.mind-over-data.com</a>
<b>Georgia</b> Atlanta, GA CFUG <a href="http://www.acfulg.org">www.acfulg.org</a>	<b>New Jersey</b> Central New Jersey CFUG <a href="http://www.cjcfug.us">http://www.cjcfug.us</a>	<b>Texas</b> Austin, TX CFUG <a href="http://cftexas.net/">http://cftexas.net/</a>
<b>Illinois</b> Chicago CFUG <a href="http://www.cccfulg.org">http://www.cccfulg.org</a>	<b>New Hampshire</b> UNH CFUG <a href="http://unhce.unh.edu/blogs/mmug/">http://unhce.unh.edu/blogs/mmug/</a>	<b>Texas</b> Dallas, TX CFUG <a href="http://www.dfwcfug.org/">www.dfwcfug.org/</a>
<b>Indiana</b> Indianapolis, IN CFUG <a href="http://www.hoosierfusion.com">www.hoosierfusion.com</a>	<b>New York</b> Rochester, NY CFUG <a href="http://rcfulg.org/">http://rcfulg.org/</a>	<b>Texas</b> Houston Area CFUG <a href="http://www.houcfug.org">http://www.houcfug.org</a>



# User Groups

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



## About CFUGs

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

## INTERNATIONAL



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

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

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

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

**Canada**  
Kingston, ON CFUG  
[www.kcfug.org](http://www.kcfug.org)

**Canada**  
Toronto, ON CFUG  
[www.cfugtoronto.org](http://www.cfugtoronto.org)

**Germany**  
Central Europe CFUG  
[www.cfug.de](http://www.cfug.de)

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

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

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

**Scotland**  
Scottish CFUG  
[www.scottishcfug.com](http://www.scottishcfug.com)

**South Africa**  
Joe-Burg, South Africa CFUG  
[www.mmug.co.za](http://www.mmug.co.za)

**South Africa**  
Cape Town, South Africa CFUG  
[www.mmug.co.za](http://www.mmug.co.za)

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

**Sweden**  
Gothenburg, Sweden CFUG  
[www.cfug-se.org](http://www.cfug-se.org)

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

**Turkey**  
Turkey CFUG  
[www.cftr.net](http://www.cftr.net)

**United Kingdom**  
UK CFUG  
[www.ukcfug.org](http://www.ukcfug.org)





# DESIGNING COLD FUSION APPLICATIONS FOR DEPLOYMENT AS EAR FILES

**There are benefits for everyone**

By Jochem van  
Dieten and  
Mark van Hedel

**E**nterprise Application aRchive (EAR) files are a standard and portable manner for packaging

applications to be deployed on J2EE application servers, such as Adobe JRun, BEA WebLogic, and IBM WebSphere. An EAR file is, in essence, just a big ZIP file with a special layout of the directory tree inside that contains a full application, see the following code. EAR files have a well-defined format and make Java bytecode truly portable. The same EAR file that we compile and build on our i386 Windows laptop can also run on a 64-CPU mainframe running commercial Unix.

## Directory Structure of an EAR file application.ear

- META-INF

This article originally appeared on the Adobe Developer Center. Reprinted with permission

- application.xml
- contextroot.war
- CFML source
- HTML sources
- CSS
- images
- WEB-INF
- web.xml
- ColdFusion runtime

As you can see from this directory structure, an EAR file deployment for a ColdFusion application can include both the CFML source code that you write and a complete ColdFusion server runtime. It's important to note that even when stripped of all documentation and images, an EAR file can be quite large – over 50 MB, not including your source code.

In this article you will learn why it is beneficial to deliver your ColdFusion applications using EAR files. We will also explain what to watch out for when you design applications that you are planning to deliver in an EAR file.

## Requirements

### ColdFusion MX 7 Enterprise

To try, go to [http://www.adobe.com/go/devcenter\\_cf\\_try](http://www.adobe.com/go/devcenter_cf_try), and to buy, go to: [http://www.adobe.com/go/devcenter\\_cf\\_buy](http://www.adobe.com/go/devcenter_cf_buy).

Note: You will need to install the ColdFusion multiserver installation. This option is available in the licensed Enterprise





Edition, the Developer Edition, or in the 30-day trial edition of ColdFusion MX 7. This requires that you download the trial or purchase the Enterprise, Evaluation, or Developer Edition. You must select “multiserver configuration” during the installation process.

#### Apache Ant

Download Apache Ant from <http://ant.apache.org/>.

#### Prerequisite Knowledge

- Basic understanding of ColdFusion
- Basic J2EE knowledge

### Advantages of EAR Files

There are many reasons for using EAR files to deploy ColdFusion applications for your customers. Not all are applicable to all customers, but there are generally benefits for everyone.

On the business side, J2EE is a standard technology that has a lot of traction. If your customers already have a J2EE environment, ColdFusion will fit in without changes to their infrastructure. If you want, you can even position ColdFusion as a 4GL language for rapid application development. All that the server administrator sees is an EAR file; it may be big but an EAR file is like any other EAR file. This advantage can open a whole new market that was previously the exclusive domain of Java shops.

On the technical side, EAR files can provide more than just a format for delivering your code. Because EAR files can include the complete ColdFusion runtime, you can also package additional components like database drivers and preconfigured ColdFusion

runtime settings. This means you can deliver your application to your customer with far fewer instructions for installation and configuration – greatly reducing the risk of a deployment error.

Combined with the J2EE underpinnings, using EAR files makes it easier to deploy your application in more than one instance on one or more servers. You also have the option to combine these instances into a cluster that provides fault-tolerance and increases the scalability of the application you are deploying.

From a support standpoint, we have all been there: a customer reports that the latest patch doesn't work. After several hours of frantic debugging, you discover that the issue is that the customer never installed a previous patch. However, if you deliver your application as an EAR file, you can ensure that the customer has installed the latest version of every file because each EAR file you deliver contains the entire application.

### Challenges to Deploying Applications as EAR Files

There are challenges to delivering your application as an EAR file. Some of the aforementioned advantages can, in other contexts, become disadvantages. For instance, because all the code is compiled and packaged in the EAR file, the customer cannot tinker with it. This also means that you cannot tinker with it after preparing it. If you frequently have to add last-minute, one-line adjustments to your code during deployment, you might find yourself at a disadvantage.

Likewise, creating EAR files requires that you own the ColdFusion Enterprise Edition. You can still develop and build your application using ColdFusion Developer Edition on your local host, but for deployment you must own an Enterprise Edition license for every two CPUs on which you deploy your application. EAR deployments will not work on ColdFusion Standard Edition.

The biggest challenge is how your application interacts with its environment. Because you are delivering a self-contained application, you must be sure to put everything in the EAR file – code, HTML, CSS, additional components, and so forth. Deploying the application should require as little configuration as possible, both to make it easier for your customer and to reduce the chance of something going wrong.

For interaction with the Web server, the most important issue is that you pick the right context-root. The context-root is the subdirectory under which the application is located. By selecting different context-roots, you can deploy multiple applications on one Website. Putting your application in a subdirectory also enables you to handle uploaded user content gracefully (such as avatars on a forum): you can upload them to a sibling directory of the application so the Web server can process them directly. Don't be greedy and use / as the context-root. Because you don't know if the person who deploys your application is going to use an empty J2EE server instance or not, you should develop your application to always run inside a context-root. All paths in your application should be relative to this context-root. When you create your application, choose a context-root so that the name is unique on the deployment server; don't use any common names, such as “admin.”

Some of the challenges require you to think about your application in a whole new way. When you package your application as an EAR file, the J2EE server has to expand the application before it can run it. Some J2EE servers, like JRun, do so automatically, which adds some overhead the first time it starts. Others, like WebSphere,



require the server administrator to expand the EAR file manually when deploying on it. J2EE servers that expand the EAR file automatically do so to a temporary directory and start the application from there.

One side-effect of this is that the application no longer knows where the EAR file is located; the application knows only the temporary directory to which it was extracted. As a result, when you change something using the ColdFusion Administrator, it ends up in the temporary directory and not in the EAR file. Therefore, if the server administrator moves the EAR file from one server to another, the application will lose any changes that you have made using the ColdFusion Administrator. Even worse, if the operating system deletes all temporary files upon reboot, you will lose all your changes to the ColdFusion Administrator just by rebooting the system. Luckily, this is largely a theoretical problem. Still, if possible, deploy your application as an expanded EAR file. Be mindful of this issue, because even when deploying an expanded EAR file, it will affect your settings when the server administrator moves the EAR or you release a new version.

Another way this becomes a problem is in the use of ColdFusion mappings. When ColdFusion starts from the temporary directory, it automatically updates its internal mapping to CFIDE to point to the temporary directory from which it is executed. All the other mappings will not be updated, however.

## Building an EAR File

Now that we are done reviewing the benefits and challenges of EAR files, let's get to work and build one.

### Building an EAR File Using the ColdFusion Administrator

The easiest way to create an EAR file is from the ColdFusion Administrator:

1. Open the ColdFusion Administrator.
2. Select the J2EE Archives (ear/war) menu option.
3. Fill out the two-page form and ColdFusion will create the EAR file.

You can find all of this in the ColdFusion LiveDocs: <http://livedocs.macromedia.com/>.

When you build an EAR file, consider carefully what exactly you want to include. The ColdFusion Administrator gives you some control over the basic settings.

You can include the Administrator, disable debugging, include the source code, and even include third-party components. For example, if you need to include the PostgreSQL JDBC

driver, you can include it in your EAR file simply by putting the JAR file in the `cfusion-ear/cfusion-war/web-inf/cfusion/lib` directory. It will be included automatically when ColdFusion generates the EAR file. Don't forget to include a copy of the license for every third-party component with your EAR file.

## Automatic Generation Through Apache Ant

Apache Ant is the de facto standard for automated building and deployment scripts in the Java world. You can use it to run your custom buildfiles to build an EAR file exactly as you need it. In fact, that is exactly what the ColdFusion Administrator does behind the scene. If you look in the `{jrunroot}/cfusion/cfusion-ear/cfusion-war/WEB-INF/cfusion/packages/compiletest` directory, you can find the Ant buildfile that was used by the ColdFusion Administrator when you built your EAR file from the ColdFusion Administrator.

## Preconfiguring EAR Files

As we explained in the challenges section, it is important to put as much configuration in your EAR file as possible. When you include everything that your application needs, it provides the easiest experience for your customers.

## Build-Time Configuration

When you use a custom build process through Ant, you can set many more options at the time you generate your EAR file by writing them to the appropriate XML file and including that XML file with your EAR file. If you look under the packages directory again, you will see that there are examples of XML files to include with your EAR file. Import them into your version control system and adapt them to your needs. Table 1 lists the XML files you need for the most common settings.

As you can see, some important categories of settings are missing, including data sources, mail servers, and mappings. That's because you typically cannot add them at build time; you must add them later.

## Deploy-Time Configuration

There are some settings that you cannot include at build time because you don't know the information yet. For instance, LDAP servers for authentication, SMTP servers for e-mail, and data sources differ from deployment to deployment. Configuration of those settings is part of the deployment process.

The easiest way to add configuration to the deployment process is to provide a wizard when the user deploys the EAR file to a J2EE server for the first time. The wizard asks for all the settings you need and uses that Admin API to configure ColdFusion. Obviously, you need to limit access to this wizard – for instance, to localhost and the IP address of the server itself. But as we explained in “Challenges to Deploying Applications as EAR Files,” the settings you change this way are not guaranteed to persist, because if ColdFusion is deployed as a compressed EAR file, the files will be written to a temporary directory. To make sure the settings persist, you must write them to another location.

This has always been a problem in the J2EE world. One of the ways to solve it is to write the settings to a `.properties` file in a location that is in the class path. To do so, you must ask the user

Setting	Location
Search Engine Safe URLs	web.xml
Debugging	neo-debug.xml
Requesting timeouts and threads	neo-runtime.xml
Logging	neo-logging.xml
Application variable settings	neo-runtime.xml
Session variable settings	neo-runtime.xml
Missing template handler	neo-runtime.xml
RDS security	neo-security.xml
Scriptprotect	neo-security.xml
Migration wizards	adminconfig.xml

Table 1. XML Configuration Files for Various Settings

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

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



**ONLY**  
**\$69<sup>99</sup>**  
ONE YEAR  
12 ISSUES

**Subscription Price Includes  
FREE JDJ Digital Edition!**

[www.JDJ.SYS-CON.com](http://www.JDJ.SYS-CON.com)

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



OFFER SUBJECT TO CHANGE WITHOUT NOTICE



running the wizard to enter a local folder in the classpath. With a cffile action, you can place a properties file in the classpath with the settings that a user has entered in the wizard. In that file, you can then save the configuration of the mail server, data sources, LDAP server, and so on.

By placing this properties file in your class path, you'll be able to read these properties later without knowing the final placement of the file. To do this, you need one Java class that reads the properties and passes them to your ColdFusion application. Java classes can access these properties files and read them. This way you can pass the properties to ColdFusion and use them in your application.

There are a few reasons why you would want to put the properties in the class path:

- To make sure the default settings will still be available when you deploy a new EAR file
- To make sure the settings for your ColdFusion Administrator can be restored when the application is restarted

Finally, your application must check if the properties are known to the JVM. If not, the application must run the wizard again and ask the user for those settings.



## Application Start-Time Configuration

As we explained earlier, some settings such as mappings can be set only at application start time because they may change until that moment. The way to do so is to set mappings on the fly in `onApplicationStart` in combination with the Admin API. Because the Admin API is in the CFIDE directory and the mapping to the CFIDE directory is updated automatically by ColdFusion, you can still access it.

If you want to use the Admin API from your code, however, you will need the password. This means you have to hard-code the Administrator password into your EAR file. Because you cannot change the password in the EAR file, you have a potential security leak: every copy of that EAR file will have the same hard-coded password. One way to protect against this is to leave the ColdFusion Administrator out of your EAR file and provide your own administrative interface using the Admin API, making it accessible only from localhost – and offering access only to those settings of ColdFusion that you want your customers to be able to change.

Obviously this is also the time to apply all the settings you wrote to the `.properties` file at deployment time. Since the `.properties` file is in the classpath, you can use a native Java class to read it.

## Where to Go from Here

As you can see, the decision to deliver your application as an EAR file requires careful thought. You have to design your application while keeping J2EE servers, context-roots, the locations of files, and how you are going to configure your application in mind. We have touched on the basics of a three-stage configuration method that allows configuration persistence through

application updates.

- In the first stage, customize the EAR file as much as possible with all the settings you know beforehand. These settings include debugging, timeouts, script protection, Java classes added to the classpath, and so forth.
- In the second stage, create a configuration wizard that will ask the administrator for settings regarding the mail server, data sources, scheduled tasks, and permanent storage when the application is started the first time. Although you can apply these settings immediately, you must write them to a `.properties` file in the classpath, outside the EAR file and temporary directory so the settings will persist.
- In the last stage, add settings that can be automatically derived to `onApplicationStart` so they will be re-created whenever the application starts. This typically includes everything that relates to physical paths on the server, like mappings and temp directories. At this stage you must also check for the existence of a `.properties` file with configuration settings previously saved in the second stage.

Our solution is not the only way to create and deploy EAR files and solve configuration issues. Most J2EE servers offer their own custom extensions for dealing with this problem. If you know beforehand the J2EE server to which you are developing, we suggest checking its documentation for best practices for that server.

## Resources

To learn more about deploying EAR files, see the following resources:

- *Managing Clusters and Packaging Apps Easily in ColdFusion MX 7*: <http://www.adobe.com/devnet/coldfusion/articles/clustering.html>
- *ColdFusion LiveDocs: Deploying ColdFusion Applications*: <http://livedocs.macromedia.com/coldfusion/7/html-docs/00001758.htm>
- *ColdFusion LiveDocs: Packaging Applications in J2EE Archive Files*: <http://livedocs.macromedia.com/coldfusion/7/html-docs/00001761.htm>
- *ColdFusion LiveDocs: Using the cfcompile Utility*: <http://livedocs.macromedia.com/coldfusion/7/html-docs/00001762.htm>



## About the Authors

*Jochem van Dielen is longtime ColdFusion developer and Adobe Community Expert (formerly Team Macromedia) member for ColdFusion. He is currently working on delivery automation and black-box testing at Prisma IT ([www.prisma-it.com](http://www.prisma-it.com)).*

*Mark van Hedel is an Advanced Certified ColdFusion MX7 Developer and MCI at Prisma IT. He is currently working on automating the ColdFusion development process and creating object-driven applications.*

[jochem@prisma-it.com](mailto:jochem@prisma-it.com)

[mark@prisma-it.com](mailto:mark@prisma-it.com)



# Visit the *New*

**www.SYS-CON.com**

# Website Today!

## The World's Leading i-Technology News and Information Source

# 24/7

### FREE NEWSLETTERS

Stay ahead of the i-Technology curve with E-mail updates on what's happening in your industry

### SYS-CON.TV

Watch video of breaking news, interviews with industry leaders, and how-to tutorials

### BLOG-N-PLAY!

Read web logs from the movers and shakers or create your own blog to be read by millions

### WEBCAST

Streaming video on today's i-Technology news, events, and webinars

### EDUCATION

The world's leading online i-Technology university

### RESEARCH

i-Technology data "and" analysis for business decision-makers

### MAGAZINES

View the current issue and past archives of your favorite i-Technology journal

### INTERNATIONAL SITES

Get all the news and information happening in other countries worldwide

## JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

*IT Solutions Guide*

*Information Storage+Security Journal*

*JDJ*

*Web Services Journal*

*.NET Developer's Journal*

*LinuxWorld Magazine*

*Linux Business News*

*Eclipse Developer's Journal*

*MX Developer's Journal*

*ColdFusion Developer's Journal*

*XML Journal*

*Wireless Business & Technology*

*Symbian Developer's Journal*

*WebSphere Journal*

*WLDJ*

*PowerBuilder Developer's Journal*



# Advanced XML Processing with StAX

**A powerful, fast, and efficient alternative  
to other ways of XML parsing**

By Jim Collins

Putting support for XML processing in ColdFusion 6.0 was regarded as a major feature upgrade. With the switch to Java, ColdFusion could leverage the existing Java functions in Jakarta Commons and add support for things like Web Services (Axis). However, binding itself to Java also bound ColdFusion to the limitations of the Java feature set.

When MX was released, the Xerxes XML parser was state-of-the-art. Java XML processing has advanced significantly since then. This article will discuss these new developments and show how you, the developer, can leverage them from ColdFusion. Some experience with creating and using Java objects with ColdFusion will help but isn't required.

## Conventional CFXML

Let's look at a trivial example of using ColdFusion XML functions to parse an XML object. ColdFusion Developers Journal (CFDJ) provides an RSS feed for published authors with a list of their articles. We will use the RSS feed for Nic Tunney, the developer of ObjectBreeze, as our sample XML. Some people are under the impression that XML has to be

stored in a file to be processed but this is not the case.

RSS, ATOM, and even well formed XHTML work quite well. The RSS feed is in XML and contains various elements. Like other RSS feeds, the root element is "rss," with XmlText, XmlAttributes, and the channel as elements under the rss root. These elements are shown in Figure 1.

The channel element contains the information that we're interested in. You will see that the channel element contains repeated "item" elements corresponding to each article Nic Tunney has written for CFDJ. This item element contains the article title, a URL to the article, a publication date, and a description. The ColdFusion code to get this information is simple and is in Listing 1. It results in the output shown in Figure 2

Here we've used the DOM to extract the information we want. This information could also be extracted using XPath or an XSLT stylesheet. So what's going on here? A CFDump of a ColdFusion XML document using `<cfdump var="#MyXML.getClass().getName()#"/>` will show that MyXML is a Java object of type `org.apache.xerces.dom.DeferredDocumentImpl`. ColdFusion has retrieved the entire RSS object and created a DOM (Document Object Model) representation of it in memory. A complete guide to the structure of a ColdFusion XML Object can be found at [http://livedocs.macromedia.com/coldfusion/7/htmldocs/wwhelp/wwhimpl/common/html/wwhelp.htm?context=ColdFusion\\_Documentation&file=00001510.htm#1119477](http://livedocs.macromedia.com/coldfusion/7/htmldocs/wwhelp/wwhimpl/common/html/wwhelp.htm?context=ColdFusion_Documentation&file=00001510.htm#1119477).

This is the result we would expect because ColdFusion uses the Xerxes parser (part of the Apache XML Project) to achieve this, and Xerces is a DOM parser.

What does this mean? The first method of parsing for XML was DOM parsing. In this model the entire XML file (or feed) is read and a DOM object is created in memory. For this reason, the DOM is referred to as a “tree-based API” because it creates an object resembling a tree in memory. Although this method is simple and straightforward, it has problems associated with it. This is an example of H.L. Mencken’s observation that “For every problem there is a solution which is simple, clean, and wrong.” Reading in the entire document is excess processing overhead, especially if we’re only interested in part of the document. To add insult to injury, the DOM object created by Xerces can be two to three times larger than the original XML document.

Another issue with this approach is that frequently the developer will need a different data structure than the one made available by the DOM. It’s very inefficient to build a DOM tree and then create a new data structure and discard the original. The DOM model fails when the XML source is very large and can crash ColdFusion. Reading and creating the DOM in memory is also...very...slow.

## SAX

As a solution to these problems, the xml-users mailing list under the leadership of David Megginson created something called SAX. SAX stands for Simple API for XML. SAX reads the XML feed line-by-line and fires off events based on the type of line read. For this reason SAX is referred to as an “event-based API.” Developing in SAX is somewhat odd to the newcomer used to calling functions in the API to achieve his objectives. SAX uses callback functions (your register is a ContentHandler with the parser) that are registered with the SAX parser. The XML processing logic is usually stored inside these callback functions. More specifically, the SAX developer will write a ContentHandler interface that contains methods such as startDocument(), endDocument(), startElement(), etc. corresponding to the events encountered by the SAX parser.

SAX has a very lightweight memory and processing footprint, and is very fast. Now, this is all well and good but there are still some problems. For one, the callback issue. There’s no way in ColdFusion to add a ColdFusion function as a called method. This isn’t a ColdFusion limitation. Java developers find using callback methods counter-intuitive. It’s like driving in reverse. SAX also still processes the entire document. There’s a way to stop processing by throwing an error, but that’s like stopping your car by ramming it into a crash barrier. It works but it’s probably not optimal. Another drawback is that the programmer must keep track of the current state of the document in the code each time he processes an XML document. SAX isn’t completely useless to ColdFusion developers, however. For instance, ColdFusion itself probably uses SAX to validate XML documents.

My primary reason for discussing SAX was to introduce you to some basic concepts that are used in another model, StAX, that are very useful to the ColdFusion developer.

## StAX

It’s commonly said, “The third time’s the charm.” Although I suspect this has more relevance for interpersonal relationships, it applies equally well to software development. Dissatisfaction with the limitations of SAX led to the development of a third XML processing approach called StAX, and this time the designers got it right. StAX stands for Streaming API for XML. Unlike SAX, StAX is a

“pull” parser, which means is that you control the rate at which the XML document is parsed, or if parsing should continue at all, once you’ve found the information you need. If you’ve done the operations that you want to do on the document, you can simply stop processing it.

One of the great things about StAX is it can process an XML source of any size. And, it’s very, very fast.

StAX is implemented by using a standard API, which is then implemented by the specific StAX parser. The StAX standard API is defined by JSR 173, which can be found at <http://www.jcp.org/en/jsr/detail?id=173>.

StAX is supported by a number of vendors such as Sun, Oracle, and BEA, each of which has released its own implementation. The Open Source community is also very active in StAX development, with Tatu Saloranta’s Woodstox being the leader. Woodstox will be

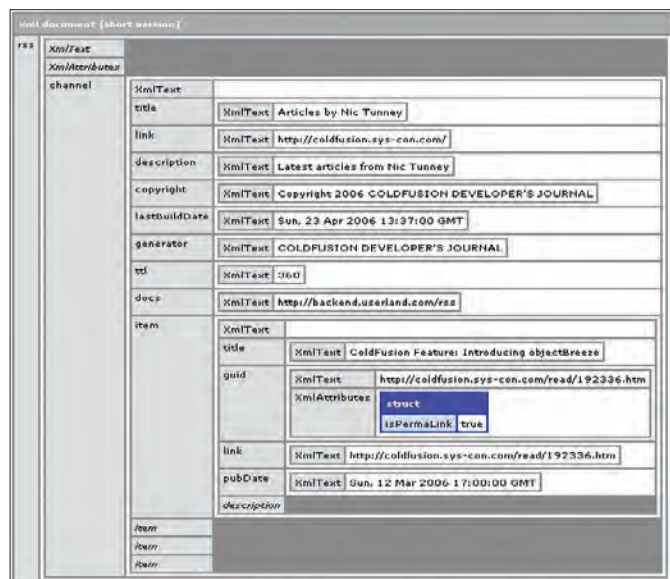


Figure 1

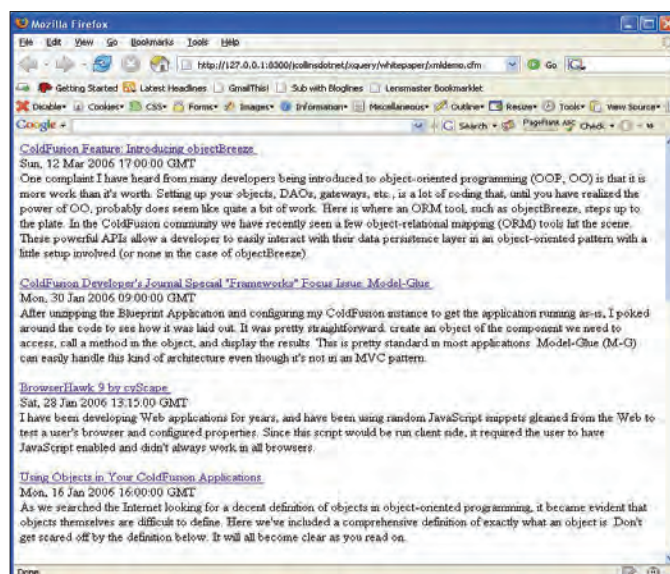


Figure 2



the implementation used in the examples that follow. Woodstox is available for download at <http://woodstox.codehaus.org/>. These examples will use a CFC that I've developed called CFStAX. The purpose of this wrapper is to make it easy to work with StAX even if you're uncomfortable working with Java APIs, and to simplify some of the setup required. CFStAX is available at <http://www.sourceforge.com/projects/cfsynergy>. CFStAX uses Woodstox and was developed with the time and help of Tatu Saloranta, its author.

StAX offers two models for processing XML, a Stream model and an Event model. These are also referred to as the cursor-style API and the Iterator-style API, respectively. In the Stream Model, the XML source is parsed using the XMLStreamReader object. The XMLStreamReader.next() method returns an integer value corresponding to the event type of the XML object encountered, i.e., start\_document, start\_element, etc. You can write a series of elseif statements to take various actions based on the event returned by XMLStreamReader.next(). The Stream model is the model most used in StAX primarily because of its simplicity.

Using the Event model, an XMLEventReader delivers XMLEvent objects using its next() method. Again, events of interest can be handled by a series of elseif statements. The XMLEventReader is particularly elegant for doing XML-to-XML transformations. There's a reason in both cases for using a series of elseif statements. A case statement is difficult to implement because of ColdFusion's lack of support for static variables. If we could do `cfswitch(XMLStreamReader.getEventType() )` then we could use switch statements.

## Examples

Using StAX is easy. The first step is to put the relevant jar files on the classpath. All StAX implementations consist of two files, a reference API that sets up the standard APIs for StAX, and an implementation file that contains how that particular implementation implements the API calls. If you're using Woodstox these files will be STAX2.JAR and wstx-asl-2.9.3.jar (ASL 2.0) or wstx-lgpl-2.9.3.jar (LGPL 2.1). The latter two files differ only in their licensing terms. I put mine on the path `C:\JRun4\servers\cfusion\cfusion-ear\cfusion-war\WEB-INF\lib` because I don't like to add to the core files. This lets me add and remove testing files from the classpath without having to worry about disturbing the core files.

Once the jar files have been added to the classpath, the jrun service must be restarted. Now that we have set up our system, we can go through some StAX usage examples and see how powerful this technique is.

### Listing 1

```
<!-- Get the RSS feed and convert into a ColdFusion XML Object -->
<cfset myXML = XMLParse("http://coldfusion.sys-con.com/author/ntunney.rss") />
<!-- Parse and output the object contents -->
<cfset alen = arraylen(MyXML.rss.channel.item) >
<cfloop from="1" to="#alen#" index="CurrentLink">
<cfoutput>
    <a href="#MyXML.rss.channel.item[CurrentLink].link.xmltext#">
        #MyXML.rss.channel.item[CurrentLink].title.xmltext#
```

## Reading XML with StAX

For these examples we'll return to the RSS feed we used before to power our CFStAX examples. They're in CFScript because CFScript is syntactically closer to the original Java code and is easier to port existing Java code to, for me anyway.

The way our original code looks using the StAX processing model with the Stream Model is in Listing 2. It's a raw implementation to show STaX processing. The CFStax equivalent would encapsulate much of this.

The Event model is similar but uses methods to determine the event properties. See Listing 3. Note that this EventStream code can cause a problem in certain cases, throwing a QName error. This is a known problem and will be addressed in a future release of CFStAX.

Writing and merging XML are other functions frequently used, and are very easy to do using StAX but are beyond the scope of this article. Full examples are available in the documentation provided with CFStAX.

## Conclusion

StAX is a powerful, fast, and efficient alternative to other methods of XML parsing and should be in the toolkit of anyone responsible for processing XML on a regular basis.

## References

Jeffrey Houser. "Using XML in ColdFusion." <http://coldfusion.sys-con.com/read/117667.htm>

Processing XML with Java (complete book online). <http://www.cafeconleche.org/books/xmljava/>

Elliotte Rusty Harold. "An Introduction to StAX." September 17, 2003. <http://www.xml.com/pub/a/2003/09/17/stax.html>

Lara D'Abreo. "StAX: DOM Ease with SAX Efficiency." January 11, 2006. <http://www.devx.com/Java/Article/30298> 

## About the Author

*Jim Collins is a ColdFusion Developer with 15 years of experience in IT and software development. His profile is available at <https://www.linkedin.com/in/jimcollins>. His blog, covering ColdFusion and Java integration is at <http://www.cfsynergy.com>.*

[jimcollins@gmail.com](mailto:jimcollins@gmail.com)

```
</a><br/>
    #MyXML.rss.channel.item[CurrentLink].pubDate.xmltext#
<br/>
    #MyXML.rss.channel.item[CurrentLink].description.xmltext#
<br/> <br/>
</cfoutput>
</cfloop>
```

### Listing 2

```
<cfscript>
```

```

// standard imports
URLObject = createObject("java","java.net.URL");
URLObject.init("http://coldfusion.sys-con.com/author/ntunney.rss");
CFStAX = CreateObject('component','CFStax').init();
XMLStreamConstantArray = CFStAX.getXMLStreamConstants();

// StAX imports
XMLInputFactory = CreateObject("java",
"org.codehaus.stax2.XMLInputFactory2").newInstance();
XMLStreamConstants = CreateObject("java",
"javax.xml.stream.XMLStreamConstants");
XMLStreamReader = CreateObject(
"java", "org.codehaus.stax2.XMLStreamReader2");

// Stuff to write output
XMLStreamReader =
XMLInputFactory.createXMLStreamReader(URLObject.OpenStream());

for (event = XmlStreamReader.next(); event NEQ
XMLStreamConstants.END_DOCUMENT; event = XmlStreamReader.next()) {
    if(event IS XMLStreamConstants.START_DOCUMENT)
    {
        WriteOutput("Event: " & XMLStreamConstantArray[event] & "<br
/>");
        WriteOutput("Start document: " & XmlStreamReader.getLocalName()
& "<br />");
    }
    else if (event IS XMLStreamConstants.START_ELEMENT) //1
    {
        WriteOutput("Event: " & XMLStreamConstantArray[event] & "<br
/>");
    }
    else if (event IS XMLStreamConstants.END_ELEMENT) //2
    {
        WriteOutput("Event: " & XMLStreamConstantArray[event] & "<br
/>");
    }
    else if (event IS XMLStreamConstants.CHARACTERS) //4
    {
        WriteOutput("Event: " & XMLStreamConstantArray[event] & "<br
/>");
        if (XmlStreamReader.hasText()) {
            WriteOutput("Element text: " & XmlStreamReader.getText() &
"<br />");
        }
    }
    else{
        WriteOutput("Event: " & XMLStreamConstantArray[event] & "<br
/>");
    }
}
}
</cfscript>

```

### Listing 3

```

<cfscript>
// standard imports
v = variables;
v.URLObject = createObject("java","java.net.
URL");
v.URLObject.init("http://coldfusion.sys-con.com/author/ntunney.
rss");

```

```

v.CFStAX = CreateObject("component","CFStax").
init();
v.XMLStreamConstantArray = v.CFStAX.getXMLStreamConstants();

// StAX imports
v.XMLInputFactory = CreateObject("java",
"org.codehaus.stax2.XMLInputFactory2").newInstance();
v.XMLStreamConstants = CreateObject("java",
"javax.xml.stream.XMLStreamConstants");
v.XMLEventReader = CreateObject(
"java", "org.codehaus.stax2.XMLEventReader2");

// Stuff to write output
v.XMLEventReader =
v.XMLInputFactory.createXMLEventReader(URLObject.OpenStream());

for (event = v.XmlEventReader.next(); XmlEventReader.hasNextE-
vent();
event = v.XmlEventReader.next()) {

    if(event.isStartDocument())
    {
        WriteOutput("Event: " &
v.XMLStreamConstantArray[event.getEventType()] & "<br />");
        WriteOutput("Start document: " & v.XMLEventReader.getLocal-
Name() & "<br />");
    }
    else if (event.isStartElement())
    {
        WriteOutput("Event: " & XmlEventReader.getElementText() & "<br
/>");
        WriteOutput("Event: " &
v.XMLStreamConstantArray[event.getEventType()] & "<br />");
    }
    else if (event.isEndElement())
    {
        WriteOutput("Event: " &
v.XMLStreamConstantArray[event.getEventType()] & "<br />");
    }
    else if (event.isCharacters())
    {
        WriteOutput("Event: " &
v.XMLStreamConstantArray[event.getEventType()] & "<br />");
        //if (v.event.hasText()) {
            WriteOutput("Element text x: " & v.event.asCharacters() &
"<br />");
        }
    }
    else{
        WriteOutput("Event: " &
v.XMLStreamConstantArray[event.getEventType()] & "<br />");
    }
}
}
</cfscript>

```

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

# Stopping Spam in Its Tracks . . .

## . . . using CAPTCHA

By Brian Rinaldi

Spammers come in many forms – e-mail spammers, search engine spammers, comment spammers, trackback spammers, message board spammers...

It seems as if there is no activity that can be done via the Internet that the lowly spammer is unwilling to intrude upon. Since spamming is a basically a numbers game, spammers spend a lot of time and effort automating the process of getting their spam to the widest possible audience. Using a CAPTCHA (Completely Automated Public Turing Test to tell Computers and humans apart) can go a long way toward preventing these automated spam attacks on your site.

In April I wrote a post on how you could easily add Peter Farrell's open source LylaCAPTCHA to Ray Camden's BlogCFC to prevent comment spam (<http://www.remotesynthesis.com/blog/index.cfm/2006/4/24/Adding-OpenSource-LylaCaptcha-to-BlogCFC>) and that very simple code ended up being rolled into the recently released BlogCFC v5. In this article, I want to expand upon that concept to show you not only how to integrate LylaCAPTCHA into any form, but also cover how we can use Rob Gonda's ajaxCFC to process the validation before the form is ever submitted.

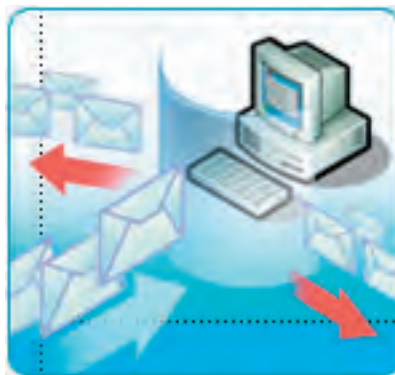
### Getting Started

Obviously the first thing you will need to do is download the appropriate project code from their respective sites. For the purposes of this example, all of my test code is within a folder called "captcha" directly under my site root. If you choose a different location, please be sure to adjust your paths appropriately throughout the example code. (The source code for this article can also be downloaded from the online version of this article at <http://coldfusion.sys-con.com>.)

You can download the latest version of LylaCAPTCHA from <http://lyla.maestropublishing.com/>. The project code consists

of two components and an XML configuration file, all of which I have placed within a subfolder called "lylacaptcha" within the example directory discussed previously. The only required change to the configuration XML that you will need is to modify the outputDirectory, which I have set to \* for the purposes of our test (i.e., the current directory – to our root example files directory). This configuration tells LylaCAPTCHA where to store the generated images, which will be deleted as soon as they are done displaying anyway. LylaCAPTCHA offers a large set of configuration options that you can also adjust if you choose – just be sure to refer to the provided documentation first. Last, as of the writing of this article, LylaCAPTCHA was at version 0.2Alpha and there was a known bug regarding the setColor method, which you will need to complete the quick fix listed under "Known Bugs or Problems" on the LylaCAPTCHA site.

Next, you'll need to download ajaxCFC, which can be done at <http://www.rob-gonda.com/blog/projects/ajaxcfc/>. I copied the necessary content into a subfolder of our examples directory named "ajaxcfc". The content you'll require includes the two ColdFusion components and the "js" directory that contains a set of six JavaScripts. No further configuration is necessary at this point for ajaxCFC.



### Building the Form

In this example, we'll be building an extremely simple form that collects a user's comments and summarily ignores them (sound like any sites you know?). What our form does care about is that you got the CAPTCHA correct, and it verifies this before the form is ever allowed to submit. First things first, let's get our application.cfm out of the way. The application file simply contains our cfapplication tag and our LylaCAPTCHA configuration code that should run only once when the application is initially configured. Obviously, if you are integrating this into your existing application, you simply need to incorporate lines 5 and 6 into your existing application startup code. Essentially, all you need to do is initialize LylaCAPTCHA with the XML file we edited above.

You can see the initialization code necessary for ajaxCFC on lines 17 through 22 of index.cfm. The first portion sets the appro-



# eclipse) developer's journal

**A DIGITAL PUBLICATION**

## Your One-Stop Guide to the Eclipse Ecosystem

The main purpose of *Eclipse Developer's Journal (EDJ)* is to educate and inform users of Eclipse and developers building plug-ins or Rich Client Platform (RCP) applications. With the help of great columnists, news you can use, and technical articles, *EDJ* is a great information source that you will be able to use to educate yourself on just about anything Eclipse related

### Subscribe Today!

**FOR YOUR DIGITAL  
PUBLICATION**

(AVAILABLE IN DIGITAL FORMAT ONLY)

**6 Issues for \$19.99**

Visit our site at [www.eclipse.sys-con.com](http://www.eclipse.sys-con.com) or  
call 1-888-303-5282 and subscribe today!

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



<http://eclipse.sys-con.com>

**SYS-CON.COM**

SYS-CON Media, the world's leading publisher of technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of Eclipse

appropriate component location, in this case `ajaxCFC/ajaxCaptcha.cfc`, which we'll discuss later, and the location of the JavaScript folder discussed in "Getting Started" above.

Overlooking the JavaScript for the moment, let's look at the form code on lines 49 through 57. There is nothing special to this form, except that there is a hidden form field that passes the hash necessary to decode the CAPTCHA response. We generate this hash per request on line 13 of `index.cfm`. It's important to note that the CAPTCHA image is actually referencing a `cfm` file and passing along the hash. The `cfm` file (`showCaptcha.cfm`) has only two lines of code. The first line generates the CAPTCHA image from the hash code we just passed, and the second serves this image up via a `cfcontent` tag that also takes care of removing the file for us as well.

Notice that I have overridden the standard form submission by adding `onsubmit="return false;"` to the form tag. This is because I call `validateCaptcha()` when the submit button is clicked, which will use `ajaxCFC` to verify that you have entered the correct CAPTCHA text by passing the value of the CAPTCHA text and hash form fields. Now let's take a look at the JavaScript necessary to perform the validation. Line 26 of `index.cfm` performs the `ajaxCFC` function call, which is invoking the `validateCaptcha` method on the `CFC` we defined in the initialization discussed previously (i.e., `ajaxCFC/ajaxCaptcha.cfc`). It also indicates that when the data is returned from the server, the `captchaValidated` JavaScript function should be called. Finally, we pass the `validateCaptcha` ColdFusion function the necessary CAPTCHA text and hash values.

Taking a look at the `validateCaptcha` function within `ajaxCaptcha.cfc` (lines 2–10), you'll see that all it does is pass back the value `LylaCAPTCHA`'s `validateCaptcha` function, which is a boolean. When this value is returned, the `captchaValidated` JavaScript function is called and passed the result (`index.cfm` lines 29–36). All `captchaValidated` does is alert the user if the test was failed or submit the form if the test was passed. Note that on lines 1–11 of `index.cfm`, I include some dummy form processing code, which does however re-validate the CAPTCHA response in case someone figures out a way to bypass our script. This probably isn't completely necessary, but is simple to do and a good precaution.


### Additional Functionality

Essentially, the code discussed above is all that is required. However, anyone who has encountered CAPTCHA knows that sometimes a particular CAPTCHA image may be extraordinarily difficult to read. Even the best CAPTCHA will be difficult to read a small percentage of the time. Well, we could refresh the page and get a new image, but then I might lose what I have typed, plus that seems counterintuitive. Wouldn't it be nice if we could just serve up a new image at the user's request? Well, we can't, so stop asking...wait, scratch that, we can.

What you need in order to generate a new image is a new hash code. We can expand our `ajaxCaptcha.cfc` with a function that will return a new hash code with all of three lines. The `newCaptcha` JavaScript function (`index.cfm` lines 38–40) calls the `getNewCaptchaHash` ColdFusion function (`ajaxCaptcha.cfc` lines 12–14), which takes no arguments and simply returns a new CAPTCHA hash reference. When the result is returned, the

`showNewCaptcha` JavaScript function is called, which first sets the CAPTCHA image source to the new CAPTCHA image, and then changes the hidden form field hash reference value. Easy right?...and you said it couldn't be done.

### Conclusion

I think at this point it should be obvious how easy it is to integrate either `LylaCAPTCHA` and `ajaxCFC`. Obviously, both projects can serve more purposes than we have covered here, and I recommend you check them out in greater detail. When you're wondering whatever happened to the treasurer of the Nigerian government, take a moment to thank the efforts of Peter Farrell and Rob Gonda for not only making superb products, but being generous enough to make them available to the rest of us free and open source. Last, if you are looking for other free and open source ColdFusion projects, check out my ColdFusion open source project list at <http://www.remotesynthesis.com/cfopen-sourcelist>. 

### About the Author

Brian Rinaldi is a Web developer at Hasbro. He is certified as an Advanced ColdFusion MX developer and has been developing in ColdFusion since 1999.

[brinaldi@remotesynthesis.com](mailto:brinaldi@remotesynthesis.com)

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

## Java for ColdFusion Programmers?



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

For class information and registration, come to [halhelms.com](http://halhelms.com).



# Subscribe Today!

## SAVE 16%

12 Issues for **\$89<sup>99</sup>**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



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

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

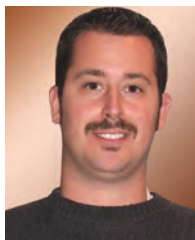
**ColdFusion** Developer's Journal





# Using Objects in Your Adobe ColdFusion MX Applications

## Developing reusable, maintainable code



By Nicholas Tunney

As ColdFusion has become a leading choice for large-scale quick-to-market application development, new CF server releases have been adapted to make it possible to integrate object classes into your Web ap-

plications. Since ColdFusion MX, developers have been able to architect applications using CFCs.

CFCs have enabled us to develop object classes, extend classes, and allow classes to inherit properties and methods from other CFCs in very short order, giving the CF development community a way to approach ColdFusion from a more object-oriented style. Using this line of thinking during your architectural process lets the development team quickly build applications that are highly reusable and maintainable.

### Prerequisites:

The reader is expected to have a solid grasp of the ColdFusion MX programming language. A basic understanding of core object-oriented principles will help with concept comprehension, but isn't required.

### Objects Defined

In object-oriented programming, an object is defined as "a container of related data (properties), and the methods that control inserting and retrieving that data. This process is termed encapsulation. Encapsulation, one of the three key properties of object-oriented programming (OOP), is the process by which access to data or procedures is limited to a specified interface. An object is an instance of

a class, which is a model instantiated to create a logical entity that handles storage and management of data (the object). The object can then be used as a single entity throughout your code."

To simplify, an object is a single instance of a class that contains data related to the object, and the methods that let data be set or retrieved from the object instance.

So how do objects help us achieve our goal of developing reusable, maintainable code?

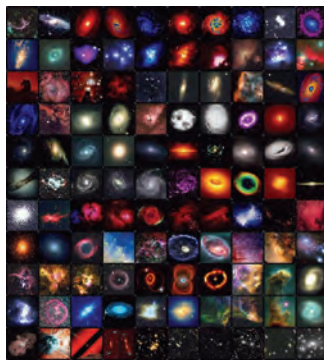
Here's a basic example: in CF5 (non-object-oriented), a developer would query the data persistence layer to return a record set. Each record in the set could, in turn, relate to an object record persisted in the database. The developer would loop over the record set and output each row with very little control. Basically, the record set was displayed in the order it was returned from the original query. If a record in the set had to be changed, the developer would have to write an update statement and supply property values one-by-one for that specific row. The initial query would then need to be rerun to get the new values before the record set was displayed again.

There is very little control over each record in the set. With the use of objects in CFMX, a developer can create a collection of objects (discussed below), and populate them from a single query. Each object in the collection can now be retrieved as a single entity, and the developer can retrieve or modify the values of each entity through accessor methods. He can then persist a single object from the collection (which may even self-validate) or the entire collection and keep using the

updated object in the request (we don't need to re-query since the object or collection already contains the updated properties). We can also persist the object or collection in a CF scope, such as the session scope, and use the object indefinitely throughout our application across requests. This is just one example as objects have many advantages over standard record sets.

### Creating a Class in ColdFusion MX

Now that we have a basic understanding as to why we would implement objects in ColdFu-



sion, we'll explain creating an object class. A class is nothing more than a standard ColdFusion component that contains private variables to store object properties, an `init()` method so the class can be instantiated, and accessor methods (sometimes referred to as getters and setters).

Below I'll create a Foo class that contains two properties: `fooName` and `fooDescription`. In a real-world application, we may use an Employee class that contains `firstName`, `lastName`, and `emailAddress`. Any ColdFusion class should begin with a `<cfcomponent>` tag:

```
<cfcomponent hint="I am the Foo class.">
```

Next, we define the object properties. Since in object-oriented programming we want to encapsulate the object properties, we use the ColdFusion VARIABLES scope to store the variables referring to the properties (see Figure 2). If we used the THIS scope instead, the properties would be available directly to a developer using the object. We want to require the developer to use the defined accessor methods to affect any object property:

```
<cfset variables.fooName = "" />
<cfset variables.fooDescription = "" />
```

Every class should contain an `init()` method. This method returns an initialized instance of the class (object). (See below.) Any default properties we want to set can also be included in the `init()` method. We could also pass in arguments to this method to use during object instantiation.

```
<cffunction name="init" access="public" returnType="Foo" output="false"
hint="I instantiate Foo">
<cfset variables.fooDescription = "none" />
<cfreturn this />
</cffunction>
```

Now that we have our properties encapsulated and have created a method that lets us instantiate an object of the class, we need to define the accessor methods that will let the developer interact with the private class properties. There are two types of accessor methods: one type sets a property value and one type returns a property value. These accessor methods are commonly referred to as "getters" and "setters." The class will define one getter method and one setter method per property (See Listing 1.). In our Foo example, both `fooName` and `fooDescription` are CF string types, so our getter return types and setter arguments are both strings.

The last step is to close our ColdFusion component definition:

```
</cfcomponent>
```

## Instantiating an Object in ColdFusion MX

To create a ColdFusion object, we have to create an instance of our class. This instance is the physical representation of the class and lets us interact with the object. There are a few ways to instantiate an object. I'll be discussing `createObject()`. As a reference, you may want to check the Adobe Livedocs for `<cfobject>`

and `<cfinvoke>`.

For the current example, `Foo.cfc` (the Foo class created above) resides in a directory off the Webroot named `cfc`. When referring to `Foo.cfc`, we use the dot notation "`cfc.Foo`." This tells ColdFusion where the file is located. In the same call, I've included the `init()` method so the object that's returned and set equal to `foo` is a fully instantiated class (see below). When running `createObject()` alone, only code that's outside the defined methods is evaluated.

```
<cfscript>
foo = createObject("component", "cfc.Foo").init();
</cfscript>
[OR]
<cfset foo = createObject("component", "cfc.Foo").init() />
```

Our object is now instantiated and can be affected through the use of the accessor methods. To learn more about the object, `<cfdump>` `foo` to see the object's structure. Note that you can't see any of the encapsulated properties in the variables scope. The accessor methods are used as displayed below.

```
<cfscript>
```

## CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
AJAXWORLD	WWW.AJAXSEMINAR.COM	201-802-3022	24
CFDYNAMICS	WWW.CFDYNAMICS.COM	866-233-9626	2
CFUNITED	WWW.CFUNITED.COM		15
CFDJ	HTTP://COLDFUSION.SYS-CON.COM/	800-303-5282	45
COMMUNITYMX	WWW.COMMUNITYMX.COM/TRIAL		6
ECLIPSE	HTTP://ECLIPSE.SYS-CON.COM	888-303-5282	43
EDGEWEBHOSTING	EDGEWEBHOSTING.NET	1-866-334-3932	4
HAL HELMS	HALHELMS.COM		44
HOSTMYSITE	WWW.HOSTMYSITE.COM	877-215-4678	29-52
HOTBANANA	HOTBANA.COM/CFDJ	866-296-1803	23
INTERAKT	WWW.INTERAKTONLINE.COM/		51
ITVCON.COM	WWW.ITVCON.COM	201-802-3023	19
JDJ	WWW.JDJ.SYS-CON.COM	888-303-5282	36
MACROMEDIA	WWW.MACROMEDIA.COM/GO/8_STUDIO8	415-252-2000	3-11
PAPERTHIN	WWW.PAPERTHIN.COM	800-940-3087	17
VITALSTREAM	WWW.VITALSTREAM.COM	800-254-7554	9

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. 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 Developer's 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. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.



```
// instantiate foo
foo = createObject("component", "cfc.Foo");
// set the fooName property in the object
foo.setFooName("myFoo");
</cfscript>
<!-- display fooName -->
<cfoutput>
#foo.getFooName()#
</cfoutput>
```

Note that the accessor methods can also be used just as easily without using `<cfscript>` (`<cfset foo.setFooName("myFoo") />`).

## Collections and Inheritance

A CF object collection is simply defined as a class containing an array of objects with methods defined for interacting with the objects on a collection level. The best way to approach collections is to have a general collection class that contains common methods for interacting with your object collection. Some of these methods can be `isEmpty()`, `add()`, `existsAt()`, `getAt()`, etc. This class can then be inherited by a class containing methods relating specifically to the type of objects that it contains. Listing 2 gives an example generic Collection class [base class derived from code written by Phill Nacelli, adapted for the objectBreeze ORM API ([www.objectBreeze.com](http://www.objectBreeze.com))]. You may see a need for additional methods.

As mentioned above, this component can be inherited by a specific object collection class. Inheritance is the process of passing class characteristics on to a child class. All of the parent's properties and methods are now available to the child class. By creating an object-specific collection and inheriting the class defined in Listing 2, methods relating specifically to the object itself can be introduced. For example, if we wanted a way to search a collection of Foo objects to find all the objects where the `fooName` property equals a certain string, we can define a `getByFooName()` method in our `FooCollection` class (see Listing 3).

Notice that inheritance is defined in the component using the "extends" attribute in the `<cfcomponent>` tag. As you'll see below, we are using the `count()` method as though it existed directly in the current class.

## Object Composition in ColdFusion MX

Simply put, object composition is the process of combining several simple objects into one complex object. This can be very powerful in your application since it allows for faster view creation and simplified object reads and commits. It's sometimes difficult to determine whether we should be inheriting from a class or compounding a class using composition. To help figure out which kind of relationship we should use, we can use "is a" and "has a" relationship definitions. When we used inheritance in our collection example, we used inheritance because the `fooCollection` "is a" Collection. Another example would be an Employee class. We could create a generic employee class and define child classes for Accounting, IT, and Sales. Each class would then have its own specific methods relating specifically to that class, such as `calculateVacation()`. Each child class could then implement its own specific formula for

calculating the vacation time derived for that specific department (there are further design patterns that should be applied here as well, but this is a basic example).

We use inheritance in this case because Sales "is a(n)" employee. When we apply composition, we do so because the object is in a "has a" relationship with another class. Each Employee has a job position. We want to keep our data normalized, and since people can change job titles throughout their career at the company, we create a Position table with a one-to-one relationship with Employee that contains all details regarding specific job positions. We can then say our object "has a" Position, and apply the relationship through composition. Each instance of Employee would contain a property that holds the Position object, and this is defined as an object composition.

To demonstrate composition in ColdFusion, let's revisit our Foo class. Foo "has a" widget. The Widget class is set up like the Foo class in that it has properties, `init()`, and accessor methods. For the example, assume that Widget has `widgetName` and `widgetDescription`. To compound widget into the Foo class, we have to add a property to hold Widget, and `getWidget()` and `setWidget()` accessor methods:

```
...
<cfset variables.widget = "" />
...
<cffunction name="getWidget" access="public" returnType="cfc.Widget"
output="false" hint="I get the widget
property">
<cfreturn variables.widget />
</cffunction>
<cffunction name="setWidget" access="public" returnType="void"
output="false" hint="I set the widget
property">
<cfargument name="widget" type="cfc.Widget" required="yes" hint="widget
value" />
<cfset variables.widget = arguments.widget />
</cffunction>
```

We now have an empty Widget property in Foo. To instantiate the Widget class in the object, the `init()` method has to invoke the widget class, and create a Widget object:

```
<cffunction name="init" access="public" returnType="Foo" output="false"
hint="I instantiate Foo">
<cfset variables.fooDescription = "none" />
<cfset variables.widget = createObject("component", "cfc.Widget").init()
/>
<cfreturn this />
</cffunction>
```

Variables.widget now contains an instance of the widget class. To access this class in our code, we use the accessor method that encapsulates widget. All of Widget's properties are available to us in the Widget property:

```
<cfscript>
foo = createObject("component", "cfc.Foo").init();
</cfscript>
```

```
<cfoutput>#foo.getWidget().getWidgetName()#</cfoutput>
[OR] (Instead of method chaining for the display)
<cfset widget = foo.getWidget()>
<cfoutput>#widget.getWidgetName()#</cfoutput>
```

Composition can go an infinite number of levels since there's no limit. Objects can also contain several other object compositions. Just keep in mind that objects are meant to make programming more maintainable, so decide during architecting when an object should be compounded or just included as a new object.

The same composition logic applies to collections as well. Going back to our Employee class, a single employee could have several e-mail addresses. To keep our data normalized, we create a one-to-many table called EmailAddress. Since Employee "has a(n)" e-mail address, we compound the objects. The Employee contains many EmailAddresses, so we define an EmailAddress-Collection that inherits from our Collection class. When we want to view a specific e-mail address, it would look like:

```
<cfoutput>
#employee.getEmailCollection().getAt(1).getEmail()#
</cfoutput>
```

A clearer method would be to loop the entire collection and set a variable equal to the current e-mail address for the duration of the loop:

```
<cfset emailCollection = employee.getEmailCollection() />
<cfoutput>
<cfloop from="1" to="#emailCollection.count()#" index="i">
<cfset email = emailCollection.getAt(i) />
#email.getEmail()#
</cfloop>
</cfoutput>
```

## Additional Topics and Summary

This article defined objects, object collections, inheritance, and composition in ColdFusion MX. This is just the tip of the iceberg. Persisting objects is a very simple process that allows for further reuse and eases application maintenance. To persist CF objects we use Data Access Objects (DAOs) and gateways. These components give us methods like commit() that let you save an entire composition to your persistence layer in one line of code. They also let you read data from your persistence layer into your objects. To learn more about DAOs and gateways (which puts the rest of the puzzle together) read the November 2005 *ColdFusion Developer's Journal* feature article entitled "Using Objects in your ColdFusion Applications."

### About the Author

Nicholas Tunney is a Macromedia Certified ColdFusion developer, and has been programming ColdFusion for over seven years. He's currently senior software architect for AboutWeb, a consulting firm located in Rockville, MD. To learn more about using objects in ColdFusion, visit Nic's blog at <http://www.nictunney.com>.

### Listing 1

```
<cffunction name="getFooName" access="public" returnType="string"
output="false" hint="I get the fooName
property">
<cfreturn variables.fooName />
</cffunction>
<cffunction name="setFooName" access="public" returnType="void"
output="false" hint="I set the fooName
property">
<cfargument name="fooName" type="string" required="yes" hint="fooName
value" />
<cfset variables.fooName = arguments.fooName />
</cffunction>
<cffunction name="getFooDescription" access="public"
returnType="string" output="false" hint="I get the
fooDescription property">
<cfreturn variables.fooDescription />
</cffunction>
<cffunction name="setFooDescription" access="public" returnType="void"
output="false" hint="I set the
fooDescription property">
<cfargument name="fooDescription" type="string" required="yes"
hint="fooDescription value" />
<cfset variables.fooDescription = arguments.fooDescription />
</cffunction>
```

### Listing 2

```
<cfcomponent name="Collection" hint="Collection of objects">
<cfset variables.container = arrayNew(1) />
<cfset variables.currentIndex = 0 />
<!--- constructor method --->
<cffunction name="init" access="public" returnType="Collection"
output="false" hint="initiates
instance of Collection">
<!--- initiate values --->
<cfset variables.container = arrayNew(1) />
<cfset variables.currentIndex = 0 />
<cfreturn this />
</cffunction>
<!--- public methods --->
<cffunction name="add" access="public" returnType="Void"
output="false" hint="adds new object to
```

```

collection">
<cfargument name="object" type="Struct" required="yes" />
<cfset arrayAppend(variables.container, arguments.object) />
<cfset variables.currentIndex = this.count() />
</cffunction>

<cffunction name="getAt" access="public" returnType="Struct"
output="false" hint="returns item at
given index">
<cfargument name="index" type="Numeric" required="yes"
hint="Collection (array) index
that contains an object" />
<cfreturn variables.container[arguments.index] />
</cffunction>

<cffunction name="existsAt" access="public" returnType="Boolean"
output="false" hint="I check if
item exists at a given position">
<cfargument name="index" type="Numeric" required="yes"
hint="Collection (array) index
that might contain an object" />
<cfset var retVal = "" />
<cfset var temp = "" />
<cftry>
<cfset temp = variables.container[arguments.index] />
<cfset retVal = true />
<cfcatch type="any">
<cfset retVal = false />
</cfcatch>
</cftry>
<cfreturn retVal />
</cffunction>

<cffunction name="removeAt" access="public" returnType="Boolean"
output="false" hint="removes
object at given index from collection">
<cfargument name="index" type="Numeric" required="yes"
hint="Collection (array) index
that might contain an object" />
<cfif arguments.index LTE this.count()>
<cfset arrayDeleteAt(variables.container,arguments.index) />
<cfreturn true />
<cfelse>
<cfreturn false />
</cfif>
</cffunction>

```

```

<cffunction name="count" access="public" returnType="numeric"
output="false" hint="returns
number of objects in collection">
<cfreturn arrayLen(VARIABLES.container) />
</cffunction>

<cffunction name="clearCollection" access="public" returnType="Void"
output="false" hint="I reset
the collection">
<cfset variables.container = arrayNew(1) />
<cfset variables.currentIndex = this.count() />
</cffunction>
</cfcomponent>

```

### Listing 3

```

<cfcomponent hint="FooCollection - Collection Object" extends="cfc.
Collection">
<cffunction name="init" access="public" returnType="FooCollection"
output="false" hint="I
instantiate FooCollection">
<cfscript>
super.init();
</cfscript>
<cfreturn this />
</cffunction>

<cffunction name="getByFooName" access="public" returnType="Any"
output="false" hint="I get a
foo object by fooName">
<cfargument name="fooName" type="numeric" required="yes" />
<cfset returnObj = "" />
<cfloop from="1" to="#count()#" index="i">
<cfif variables.container[i].getFooName() = arguments.fooName>
<cfset returnObj = variables.container[i] />
<cfbreak>
</cfif>
</cfloop>
<cfreturn returnObj />
</cffunction>
</cfcomponent>

```

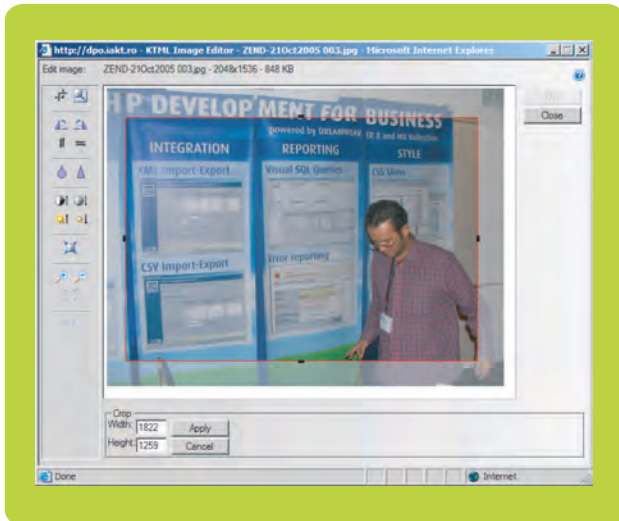
**Download the Code...**  
Go to <http://coldfusion.sys-con.com>



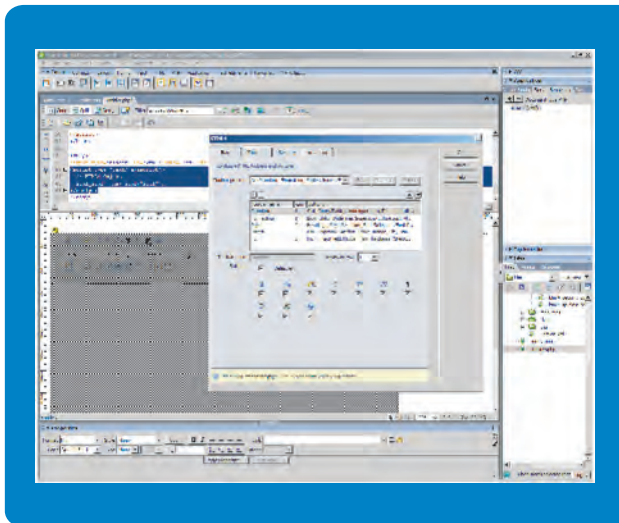
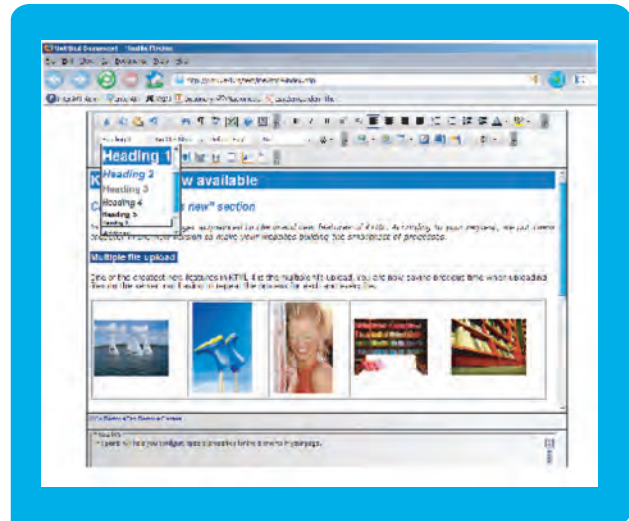
# KTML4 Word editing in browser

Along with the most affordable **UNLIMITED** licence

## Revolutionary Image Editor



## Word-like visual CSS styles



## Fast Dreamweaver integration

- Instant paste from Word
- Incredible speed
- Easy to use Word-like toolbars
- Improved CSS authoring
- Remote File Explorer
- XHTML 1.1 compliant

## Wide browser compatibility

- Multiple file upload at once
- HTML Table Editor
- Support for multimedia (Flash, Avi)
- Documents management (.doc, .pdf)
- Page templates
- WAI compliant

Please visit [www.interaktonline.com/ktml4/](http://www.interaktonline.com/ktml4/) for details

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

