SYS-CON

THE MYSTERY DESIGN PATTERN PG 7

ColdFusion Developer's Journal

ColdFusionJournal.com

January 2007 Volume:9 Issue: 1

Presorted
Standard
US Postage
PAID
St. Croix Press

YUM!
INTERSPECTACULAR

Adobe® Creative Suite® 2.3

Say hello to the next generation.

It's found in the world's most inspirational places. It's pushing the boundaries of art direction and design. Introducing Adobe® Creative Suite® 2.3. Today's most talented designers and art directors are working smarter and faster because of it. Just ask the masterminds behind INTERspectacular—they rely on the Creative Suite to help bring their ideas into the world. See how they do it at adobe.com/creativemind. It's everything but the idea. Better by Adobe.™

Luis Blanco and Michael Uman,
INTERspectacular

adobe.com/creativemind

Adobe

For answers call us at 1-866-EDGEWEB

3 3 4 3 9 3 2

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

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

By the Numbers:

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

EDGE WEB HOSTING

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

http://edgewebhosting.net

CFDJ READERS' CHOICE AWARD · MXDJ READERS' CHOICE AWARD

2003 - 2006

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

COLDFUSION Developer's Journal

editorial

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

executive editor
Nancy Valentine nancy@sys-con.com

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

production

lead designer
Abraham Addo abraham@sys-con.com

art director
Alex Botero alex@sys-con.com

associate art directors
Louis F. Cuffari louis@sys-con.com
Tami Beatty tami@sys-con.com

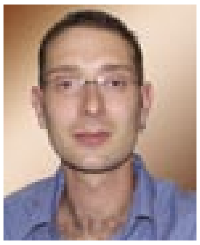
editorial offices

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

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

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

ColdFusion: So Easy, Even a Caveman Can Do It



By Simon Horwith

While recently channel surfing, one of the "Geico caveman" commercials came on the tube to remind me that Geico.com is so easy to use, even a caveman can do it.

The marketing campaign is very clever; watching it, I couldn't help thinking what a shame it was that Allaire didn't think of the slogan first. After all, ease of use certainly is one of ColdFusion's greatest strengths. That night I couldn't stop thinking about the caveman commercial and how applicable it is to CF. Questions kept entering my mind like spam forcing its way into my inbox. Is being very easy to learn and use a good thing for a programming language? Does being easy make ColdFusion a less viable business solution? Are ColdFusion developers illegitimate in the programming world? It is no secret that Java, .NET, and other developers sometimes look down on ColdFusion and ColdFusion developers. Is it a simple case of ignorance? ...or jealousy? ...or is this perception justified? I'll preempt addressing these questions with a statement that makes my own perspective and interests clear. Like many other developers, I do what I do because of a real love for my trade. Like all other developers, I also do what I do for profit, i.e., to make money. For a very long time I've been an extremely active member of the ColdFusion development community – I enjoy being active in the CF community and I love working with ColdFusion. For the past five or six months, however, I've been pretty quiet. The reason for this is that rather than focusing on engagements where I largely end up "preaching to the choir," I've been focusing on speaking directly with business owners and important decision makers in large businesses that, in reality, are the folks who can really make ColdFusion a quantifiable success. Speaking with these business-decision makers and advising them on technical and architectural issues has influenced my perception of how ColdFusion fits into the larger IT picture, and it has shaped how I choose to respond to these questions.
I'll begin with the fundamental question as to whether or not there is such a thing as a programming language that is too easy. It's not such an easy question to answer. Developing software is largely all about solving problems – and I for one will take all the help I can. CFML and the features built in to the ColdFusion Application Server don't eliminate every problem I face as a developer, but they do make many otherwise difficult tasks trivial. Wanting development to be easier is something that programmers in all development environments strive for. Any Java or .NET developer who tells you they don't want their jobs to be any easier than they already are is either lying or ignorant. Java frameworks like JUnit, Aspect-J, and Struts to name a few were all developed to make developers' lives easier. .NET developers don't use Visual Studio because it looks pretty – they use it because it rapidly speeds up development. There are thousands of other examples of frameworks, IDEs, APIs, and other tools that developers use to make their jobs and their lives easier. Be happy to admit that you use ColdFusion because it makes your life easier.

– CONTINUED ON PAGE 34

About the Author
Simon Horwith is the editor-in-chief of Cold-Fusion 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.*
simon@horwith.com

Hot Banana Web Sites Are Built Fully Loaded!

Why settle for building and managing a *vanilla* Web site when you can savor a full featured Hot Banana Web site?

Hot Banana is a fully-loaded Web content Management System with all the Web site optimization and eMarketing campaign tools your Marketing department craves. And, it's easy-to-use, search engine friendly and fine tuned for peak performance.

So what's the cherry on top? It's ColdFusion of course - you'll love how easy Hot Banana is to implement, integrate and customize.

Schedule your Free Taste Trial Today!

Features...

- 100% browser-based
- Updated RTE
- Powerful DAM
- XHTML & CSS compliant
- Multilingual support
- Granular security
- Flexible workflow
- Press release manager
- Keyword analysis
- Web analytics center
- A/B Testing
- Powerful DAM
- Form builder
- RSS & blogs
- SEO tools
- Content reuse & scheduling
- Lead source tagging & tracking
- Custom metadata
- Event registration
- Email manager

Contact Us Now:
hotbanana.com/cfdj-demo
1.866.296.1803
sales@hotbanana.com

hotbanana®
Web Content Management For Marketing

SYS-CON MEDIA

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

group publisher
Jeremy Geelan jeremy@sys-con.com

advertising

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

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

advertising sales director
Megan Mussa megan@sys-con.com

advertising sales manager
Andrew Peralta andrew@sys-con.com

associate sales manager
Corinna Melcon corinna@sys-con.com
Lauren Orsi lauren@sys-con.com

sys-con events

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

events manager
Lauren Orsi lauren@sys-con.com

customer relations

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

sys-con.com

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

web designers
Stephen Kilmurray stephen@sys-con.com
Richard Walter richard@sys-con.com

accounting

financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

subscriptions

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

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

Data Access Objects

The mystery design pattern

By Jeffry Houser

It seems that there's a lot of talk in the ColdFusion community about data access objects and data gateway design patterns. Everyone talks about how great they are and why everyone should be using these patterns.

Unfortunately, there seems to be little talk about what exactly they are and how you would implement them in a ColdFusion application. At one point I did a lot of searching for resources on these two patterns and came up with nothing. These patterns aren't discussed in the famous "Gang of Four" book, nor are they covered in Head First Design Patterns. A Web search also came up empty. So where does one go to learn? Hopefully I can shed some light on these patterns. It was only after talking to different people about them that I started to understand their purpose and how you could implement them. In this article I'll explore the Data Access Object (DAO) in detail. Next month I'll examine the gateway pattern.

What Is a Design Pattern?

A design pattern is just another name for a best practice. There are usually certain tradeoffs you make for choosing one approach over another. I recommend you try to learn as many patterns as you can, and then you'll be in a better position to decide what will work best for your current situation.

When talking about design patterns, most ColdFusion developers are usually talking about ways to structure the code that makes their business model. Design patterns in the model are a way to structure code in such a way that allows the most flexibility long-term. Software applications spend most of their life in a maintenance change as I'm sure you've already experienced.

Design patterns aren't limited to your business model, though. Model-View-Controller is a design pattern many CF developers are familiar with that's not related to how to structure your model; it's designed to help separate your model from your view code. Yahoo released a series of design patterns that are related to the user interface. You can read about them at http://developer.yahoo.com/ypatterns/index.php. When you implement DAOs or gateway objects you'll be doing so in your model.

What Is a Data Access Object?

Data Access Objects are a design pattern that lets you separate your data access from any business logic. This lets you easily change your data storage mechanism with minimal code changes. Perhaps you want to move from a MySQL database to a SQL Server database. Or perhaps you're ditching local data storage in favor of using a Web Service mechanism? Maybe you want to move some data out of the database and into XML files? Or perhaps you'll want to move them out of XML files into a database. If your application is implemented using DAOs then you'll just have to write a new DAO objects for the new data mechanism and then tell your application to use the new one instead of the old one.

In most cases, a DAO will have four methods in it, one for inserting data, one for updating data, one for selecting data, and one for deleting data. Your business model components will access the DAO objects as needed to maintain the data properly. This probably sounds harder than it is, so let me demonstrate with an example.

MyFriends RSSCategory Component

To illustrate a DAO object, I want to take a

TABLE OF CONTENTS

look at the RSSCategory.cfc from the MyFriends RSS aggregator. You can download the aggregator code from the software pod on my Web site at www.jeffryhouser.com. RSS feeds that are entered into the system can be categorized. This component is used for creating or updating one of those categories. It was originally built in the interest of speed without much thought to database portability.

The RSSCategory component has two properties, a CategoryID and a Category. It has two methods, an init method and a commit method. The init method contains a select statement. The commit method contains an update statement and an insert statement. If we change our data storage mechanism, all the SQL statements would have to be changed inside the component. This could, potentially, mean changing every component in our Model. This is about as close to a full rewrite as you can get. The intent in creating a DAO is to move the SQL statements outside of the main component and into a separate component. That way, you only have to change the DAO, leaving the rest of your app untouched.

Writing the Data Access Object

The Data Access Object won't have any properties, just the methods for selecting, inserting, updating, and deleting. (For the sake of brevity, I won't provide the delete method.). A CFC without any local properties is like a function library, and that is exactly what we're using it for.

Listing 1 shows the select method. The method name is select and it returns a query. It accepts two arguments, the CategoryID and the datasource name. The CategoryID is the primary key for the category you want to retrieve. The query name is defined as a local variable then the select query is run. I chose RSSCategory.cfc because of the simplicity of the queries. The resultant query, even if no data is returned, is passed out of the function. Easy as pie.

Listing 2 shows the insert method. This method is named insert, but returns void. It accepts three arguments, the CategoryID, the Category, and the datasource. This is different than the select method, which only needs the primary key. Since we're creating a new category from scratch, we need all the associated data for the query. The query variable is defined as a local variable on the next line. Then comes the query. In the case of the MyFriends project, UUIDs are used as primary keys so they'll be created outside of the component and passed in. If you were using other methods for primary key creation, such as an auto-incrementing integer, then you may not want to pass that value into the method. You may want to get it from the database after the insert and return it out of the method. These decisions are application-specific decisions, though, not data storage-specific. The update method, shown in Listing 3 is identical in form to the insert method and I don't have anything else to add about the code.

Modifying the RSSCategory Component

With the Data Access Object created, the next step is to modify the RSSCategory component to use the methods in the DAO instead of directly executing SQL. First, we want to create an instance variable to contain the DAO object. We can add this line as part of the pseudo constructor code:

```
variables.instance.DAO = "";
```

The modified init method is shown in Listing 4. It adds a third argument, which is the type of DAO you want to use. Most likely this will be a global setting somewhere in your app. So I don't have to change any already-written code, I made this argument optional and added a default value, which refers to the Data Access Object I just created. First I define a local query variable. Then the code creates an instance of the DAO object. In the old version of the component, the query was located next. Here we call the select method and assign the results to the query variable. The remainder of the method sets the instance variables based on the query data.

The modified commit method is shown in Listing 5. The commit method remains largely unchanged. The method signature is the same. There is one argument for the data source. A local query variable is defined. If the primary key is an empty string then the new primary key is created and the insert method executed. Otherwise, the update method is run.

Why Use a Data Access Object?

With the example behind you, you might ask yourself why would you care about DAOs? I can honestly say that I rarely use them. Most of my applications are custom built for clients. How often do companies change their database? Yes, it happens, but it's rare. In my early days I converted a lot of Microsoft Access databases to SQL Server, but it's rare to find an Access-built app. It'd be even rarer to see an Access-built Web app that uses DAOs or any advanced design concepts.

DAOs really start being a benefit if you're going to build an application that will be deployed in multiple environments and those environments are unknown. Perhaps you're building a killer CMS? Or free blogging software? If so then you're going to want to use DAOs in your development. You don't know if the next deployment will be on SQL Server, Oracle, or even XML files.

Final Thoughts

You should now have an understanding of what DAOs are and how to use them in your ColdFusion applications. I wish I could give you a list of resources to learn about gateways, but my research has always come up empty. I can think of many alternate implementations of this pattern that could achieve the same affect. I'd love to hear how you implement DAOs. Let me know!

About the Author

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

jeff@instantcoldfusion.com

Listing 1 The select method

```
<cffunction name="Select" access="public" returntype="Query">
 <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_var-
char">
 </cfquery>

 <cfreturn qGetCat>
</cffunction>
```

Listing 2 The insert method

```
<cffunction name="Insert" access="public" returntype="Void">
 <cfargument name="CategoryID" type="string" required="true">
 <cfargument name="Category" type="string" required="true">
 <cfargument name="dsn" type="string" required="true">

 <cfset var qUpdateCat = "">

 <cfquery name="qUpdateCat" datasource="#arguments.dsn#">
  insert into RSSCategories(CategoryID, Category)
  values (
   <cfqueryparam value="#arguments.CategoryID#" cfsqltype="cf_sql_var-
char">,
   <cfqueryparam value="#arguments.Category#" cfsqltype="cf_sql_var-
char"> )
 </cfquery>

</cffunction>
```

Listing 3 The update method

```
<cffunction name="Update" access="public" returntype="void">
 <cfargument name="CategoryID" type="string" required="true">
 <cfargument name="Category" type="string" required="true">
 <cfargument name="dsn" type="string" required="true">

 <cfset var qUpdateCat = "">

 <cfquery name="qUpdateCat" datasource="#arguments.dsn#">
  update RSSCategories
  set Category =
        <cfqueryparam value="#arguments.Category#" cfsqltype="cf_sql_
varchar">
  where CategoryID =
             <cfqueryparam value="#arguments.CategoryID#" cfsqltype="cf_
sql_varchar">
 </cfquery>

</cffunction>
```

Listing 4 The RSSCategory init method

```
<cffunction name="Init" access="public" returntype="Boolean">
 <cfargument name="CategoryID" type="string" required="true">
 <cfargument name="dsn" type="string" required="true">
 <cfargument name="DAOType" type="String" required="false"
                     default="RSSCategory_DAO_SQLServer">

 <cfset var qGetCat = "">

 <cfset variables.instance.DAO =  createobject('component', '#arguments.
DAOType#')>

 <cfset qGetCat = variables.instance.DAO.select(arguments.CategoryID ,

       arguments.dsn)>

 <cfif qGetCat.recordcount is 0>
 <cfreturn false>
 </cfif>

 <cfscript>
  variables.instance.CategoryID = qGetCat.CategoryID;
  variables.instance.Category = qGetCat.Category;
</cfscript>

<cfreturn true>

</cffunction>
```

Listing 5 The RSSCategory commit method

```
<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()>
  <cfset variables.instance.DAO.insert(variables.instance.CategoryID,
                                                              vari-
ables.instance.Category,
                                                              argu-
ments.dsn)>

 <cfelse>
  <cfset variables.instance.DAO.update(variables.instance.CategoryID,
                                                               vari-
ables.instance.Category,
                                                               argu-
ments.dsn)>
</cfif>

</cffunction>
```

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

www.frameworksconference.com

FRAMEWORKS 2007 CONFERENCE

Washington D.C.

February 1st & 2nd

NEW TOPICS **GREAT SPEAKERS** **NETWORK** **LATEST TECHNOLOGIES**

The Frameworks 2007 conference will be located in the Washington DC area, on February 2007. We will have lots of great speakers like last year. This year in addition to Fusebox and FLiP we will have sessions on other frameworks such as Mach-ii, Model Glue, Ruby on Rails, ColdSpring and Struts, and other methodologies such as XP and test-driven development.

We will be looking at frameworks for several web-based languages including ColdFusion, Java and .Net. There will be sessions for beginning and advanced developers, with lots of opportunities for learning from "foreign frameworks" and cross-pollination.

WHY SHOULD YOU COME TO THE FRAMEWORKS CONFERENCE?

"Frameworks is FOR developers, put on BY developers. So the information is useful and there is a ton of it... Are you into "Networking", how would you like to talk to the people who created Fusebox, FLiP, or Fusedoc's? Would you like to meet leaders in our industry using these tools? Well, you can, and that is why you should come."
Eric R. L.

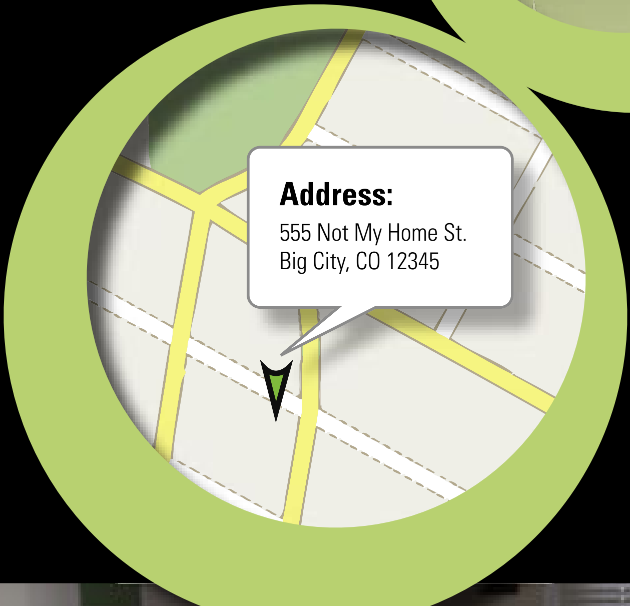
"I'm coming to the conference because our core applications are far behind the Fusebox times–we're running on Fusebox version 2! So I'm interested in learning about other frameworks that are out there, as well as ways of migrating our existing apps smoothly."
Troy B.

"To stay current with the latest fusebox and frameworks conference."
Anne S.

TeraTech Programming
www.teratech.com

Scary Question.

Exactly who is developing your next app?



Your App Starts Here.

We are the leaders in RIA development services.

INCREDIBLE APPLICATIONS

PASSIONATE USERS

PROVEN SUCCESS

CYNERGY

Unlock your potential
with the help of industry leaders in
Rich Internet Application development.
10 Years. 1400+ Customers.
Your app starts here.

CYNERGYSYSTEMS.COM

Adobe Solution PARTNER

Spry for ColdFusion Developers

AJAX made simple

Andrew Powell

AJAX (Asynchronous JavaScript and XML) is nothing new. The technologies behind AJAX have been around for quite a while. Jesse James Garrett just gave the amalgamation of XML, DOM, and JavaScript a catchy new name. Many CF developers hear buzzwords like AJAX and Web 2.0 and simply tune out because they think it's too much to comprehend.

If they'd simply stop and take a look at what AJAX can deliver, they'd see that it offers a unique opportunity to deliver a user experience that seemed beyond reach only a short time ago. With AJAX, a developer can deliver an application that requires only a fraction of the page refreshes of the typical Web application, thereby significantly enhancing the user experience. Delivering this wonderful new experience, however, is another matter. A developer must make use of the XMLHTTPRequest object, which isn't implemented the same way in all browsers. A developer must also have a pretty good understanding of JavaScript. Not all CF developers are strong JavaScript developers or even have JavaScript experience, save a few. What was needed for CF developers was a lightweight, easy-to-use way to leverage the power of AJAX. A number of frameworks for AJAX are available, but none of them are terribly easy to learn and get started with right away. There was no simple solution for AJAX that CF developers could embrace as a community.

This is where Spry comes in. Early in 2006, Adobe started releasing pre-release previews of Spry. Spry is an AJAX framework, a simplified way to make the power of AJAX available to the masses. There are three main components to Spry: XML Datasets, Widgets, and Effects. XML Datasets are, as the name implies, datasets based on an XML document. Widgets are a combination of JavaScript, HTML, and CSS that provide some form of advanced functionality in a self-contained block of code. Effects are simply that, effects that can be applied to HTML elements in a page. Spry also leverages the Google XPath library to facilitate parsing the retrieved XML document into easily accessible datasets. Currently, Spry is still in pre-release status. Adobe's intention with Spry is to make AJAX easy and accessible for Web designers who have little or no JavaScript experience. It will be an integral part of the next release of Dreamweaver. Something that Adobe didn't count on was that one of its existing user communities would latch on to this framework so quickly: the ColdFusion community.

The link between ColdFusion and Spry is almost natural. ColdFusion gives developers the ability to generate XML on the fly, which can then be fed to Spry. Some RDBMSes, such as Microsoft SQL Server (2000+) and Oracle, can even return query results as XML. All that's needed from ColdFusion is some simple formatting and a mechanism to pass the XML to the browser. XML doesn't have to be generated by the database itself though; functionality to convert any data type into XML can be written in ColdFusion by the developer. See Listing 1.

It's important to note that when creating XML to return to Spry with ColdFusion, it's not necessary to return a ColdFusion XML object. In fact, Spry won't understand what's returned if the XML object is returned because ColdFusion will serialize it into a WDDX packet. XML must be returned to Spry in its text format, as an XML document, not its ColdFusion XML object format. Using the ColdFusion XML object when generating the XML document can actually create more overhead in a high-load application because it takes up more space in memory than a simple string would in memory. Best practices for generating XML to be delivered to the browser or AJAX shouldn't involve the use of the ColdFusion XML object.

To return the XML document to the client, ColdFusion needs to set the proper MIME type. If the proper MIME type isn't set, the server will tell the client to expect HTML and not XML. To return the XML document to the client, the process is as simple as using CFCONTENT and outputting the variable. (See Listing 2.) This will ensure that the client expects XML to come back from the server, not HTML. This is key when dealing with Spry because not returning XML will throw an error when loading the XML dataset.

Spry doesn't care if its data comes from a flat XML file, ColdFusion, PHP, or wherever. All it cares about is the MIME type being "text/xml." The syntax for using Spry doesn't change with dynamic XML coming from ColdFusion. This is the beauty,

Name	Position
Wayne Rooney	Forward
Ryan Giggs	Midfield
Gary Neville	Right Back

Table 1: Players

and commonality, of Spry. If the XML is a static document, or a dynamic document, Spry will read it. All Spry cares about is the XML being well formed and the XPath being valid. This opens Spry up to the Web development community at large, not just the ColdFusion community.

Now that this new functionality is available to ColdFusion developers, where does it fit into the user experience? A common problem that Spry solves quite easily is the creation of dynamic dropdown boxes that don't cause the page to be refreshed entirely. This lack of a visible round-trip to the server is beneficial because it enhances the user experience and makes a Web application behave more like a traditional desktop application. Does this new ability to create rich user interfaces for Web applications mean that Spry should be implemented heavily throughout ColdFusion applications? The answer is a resounding "no." If using ColdFusion to generate XML for Spry, remember that each time a request for XML is made, a ColdFusion thread is being used. High-load applications can suffer performance problems in such an environment if the load gets to be too great.

Spry is a perfect fit for teams that are already in a page-based pure HTML environment. For teams that already have a MVC framework in place, Spry offers a nice enhancement to the view layer of the model without compromising the existing framework. If the goal of a development team is to move into more advanced user interfaces then Spry may not be the best fit. An environment such as that may want to look to Flex 2, especially if the development team needs to visualize and manipulate data in a non-HTML manner. Each project and team will have different requirements. Some situations will call for Flex, whereas some situations will call for Spry. It's simply a matter of choosing the proper solution for the problem at hand.

Spry is giving developers a way to change the way interfaces are built within the current limitations of the browser. This opens new doors for developers who are not looking towards Flex 2, but want some of the same type of functionality in their applications. Combine this powerful AJAX framework with the power of ColdFusion and the possibilities are endless. Spry was really meant for the designers on the Web, but ColdFusion users are the ones who will take it beyond its boundaries and into an even richer experience than was originally intended.SIDEBAR In addition to simply displaying the XML data, Spry offers functionality for sorting, filtering, and dynamically reloading datasets. This allows a richer user experience than simply displaying data. Datasets now become interactive and bring a richness to the browser that was previously non-existent. Adobe is continually adding more functionality to Spry with every pre-release. As Spry moves closer to full release, expect to see nested datasets and support for JSON (JavaScript Object Notation) datasets as well.SIDEBAR.

About the Author

Andy Powell, a member of the editorial board of Web Developer's & Designer's Journal, is a consultant at Universal Mind (www.universalmind.com)

andrewp@univeralmind.com

Listing 1: roster.cfc

```
<cfcomponent>
<cffunction name="getPlayers" access="public" returntype="string"
output="false"> <cfset var plrQry = queryNew("")/> <cfset var retXML
= ""/> <cfquery dataSource="myDataSource" name="plrQry"> SELECT name,
position FROM players </cfquery> <cfsavecontent variable="retXML">
<cfoutput>
<firstXI>
   <cfloop query="plrQry">    <player>     <name>#name#</name>
<position>#position#</position>    </player>   </cfloop>  </firstXI>
  </cfoutput> </cfsavecontent> <cfreturn retXML/></cffunction>
</cfcomponent>
```

Listing 2: playersXML.

```
cfm<cfset roster = createObject('component', 'cfc.roster')/><cfcontent
type="text/xml" reset="true"> <cfoutput>#roster.getPlayers()#</cfout-
put></cfcontent>
```

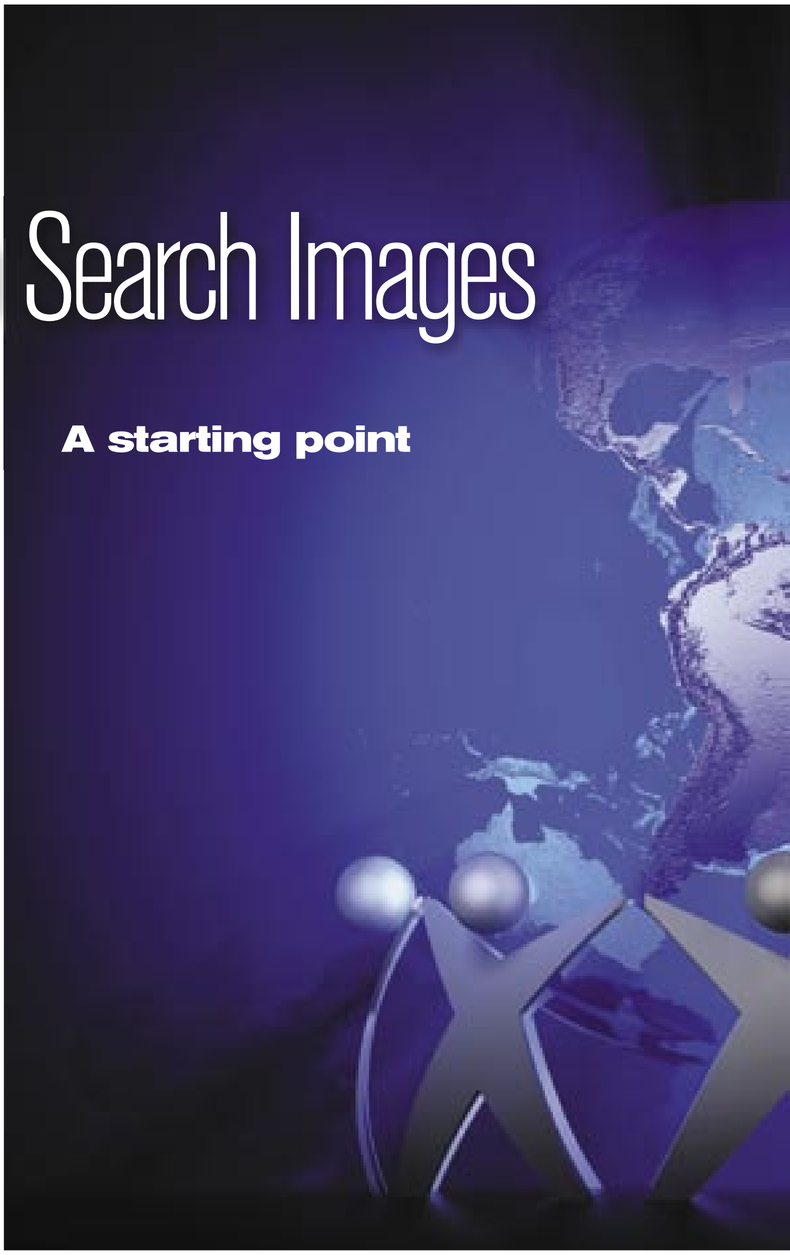
Download the Code...
Go to http://coldfusion.sys-con.com

"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.



Using ColdFusion to Search Images Based on Color

A starting point

By Seth Duffey

This article describes a basic method for indexing and searching images and digital photographs based on color using ColdFusion and CFImage-Histogram (http://www.leavethatthingalone.com/projects/cf-histogram/). This method indexes and searches color in images quickly using ColdFusion.

I don't pretend to claim that this is the best way to search images based on color as there are other more in-depth and precise methods. For more information I suggest you read more about content-based image retrieval systems (http://en.wikipedia.org/wiki/Content-based_image_retrieval).

Note: The code examples below range from pseudo-code to 95% of what you will need to index and search images. There are so many possibilities of how images and image color information can be stored that it's hard to demonstrate a one-size-fits-all solution.

Where to Start

To be able to search for colors in an image we have to know something about the colors that make up that image. A histogram is a good way to do this. More important, a color histogram is useful because it tracks the frequency of colors that occur in an image. Each occurrence of a color, in this case red-green-blue, is counted based on its value 0-255.

To get the color histogram of an image, you need to inspect each pixel and calculate the red/green/blue components of that pixel. This can be a very slow process, so it may be best to resize a large image before trying to calculate the color histogram.

Some Statistics

The CFImageHistogram (http://www.leavethatthingalone.com/projects/cfhistogram/) color histogram creates an array for each of the three colors (R-G-B). In each array there are 255 cells that store the count of the occurrences of that particular color (0-255). While the arrays are nice to have, it's more important to be able to search knowing the mean and standard deviation of that color's array. The mean will tell us what the average color is and the standard deviation will give us an idea of how spread out the color range might be.

Color Information

This statistical color information works well. We can take an image and find the average red, green, and blue for it, then store

this information in a database and query for images that have a high occurrence of blue, for example. But the problem is that most images have a mix of colors, so when you retrieve the color histogram of the entire image, you get an average result that tends to be the average color for the entire image and that's most likely some grayish color (see Figure 1).

A solution to this is to split the image into sub-images. By looking at the color histogram information and the mean/standard deviation of smaller regions of the image, we can better search within that image. By looking in smaller areas of the image we're more likely to get average colors for the histogram that better reflect the colors in the image (see Figure 2).

The more sub-images (or bins) that you use, the more accurate your searches can be; however, there is a trade off in the processing time it takes to index the images and then later the search query time.

Indexing the Images

The code examples below might be a little unclear, so I've included a basic flow chart that should help in understanding the process of indexing the color information on the images (see Figure 3). This image is also available at www.leavethatthing-alone.com/examples/cfcolorsearch/img/cfflowchart.png.

The first step to getting this color information out of an image is to buffer an image so we can manipulate it:

```
<cfscript>
 //image to get color information from
 photo = expandPath("photo.jpg");
 //Jpeg Codec
 jpegCodec = createObject("java", "com.sun.image.codec.jpeg.JPEGCodec");
 //file input
 fileInputStream = createObject("java", "java.io.FileInputStream").
init(photo);
 //decodes the JPEG
 decoder = jpegCodec.createJPEGDecoder(fileInputStream);
 //get buffered image
 bufferedImage = decoder.decodeAsBufferedImage();
 imageHeight = bufferedImage.getHeight();//buffered image height
 imageWidth = bufferedImage.getWidth();//buffered image width
</cfscript>
```

Now that we have a buffered image, we need to be able to isolate the subimages within the image. The "subImage" method of the bufferedImage can be used. The subImage creates a new buffered image based on a rectangle defined in the arguments:

```
subImage = bufferedImage.getSubimage(x,y,width,height);
```

Once we have an isolated region of an image, we need to get the color histogram and color statistics of it. This can be done with the CFImageHistogram:

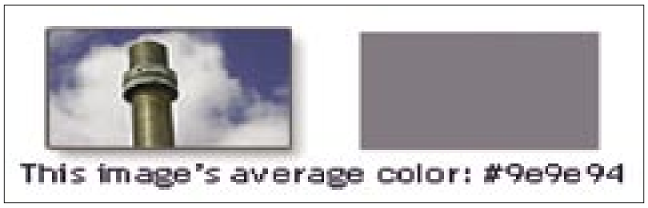


Figure 1: Example image and its mean color to illustrate that the average color of an entire image is not accurate to search with.

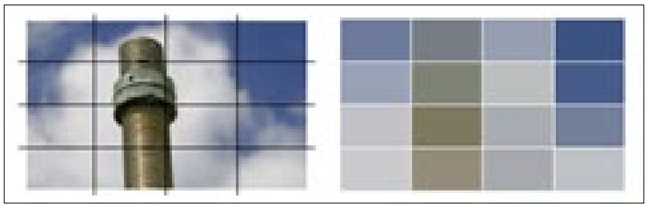


Figure 2: Example image shows that the mean color of the 16 sub-images better reflects the colors in the original image.

```
<cfscript>
 //create image histogram object
 imageHistogram = createObject("component","imageHistogram").init();
 //set the buffered subimage into the image histogram object
 imageHistogram.setBufferedImage(subImage);
 //get the color histogram and color statictics struct
 hist = imageHistogram.getColorHistogram();
</cfscript>
```

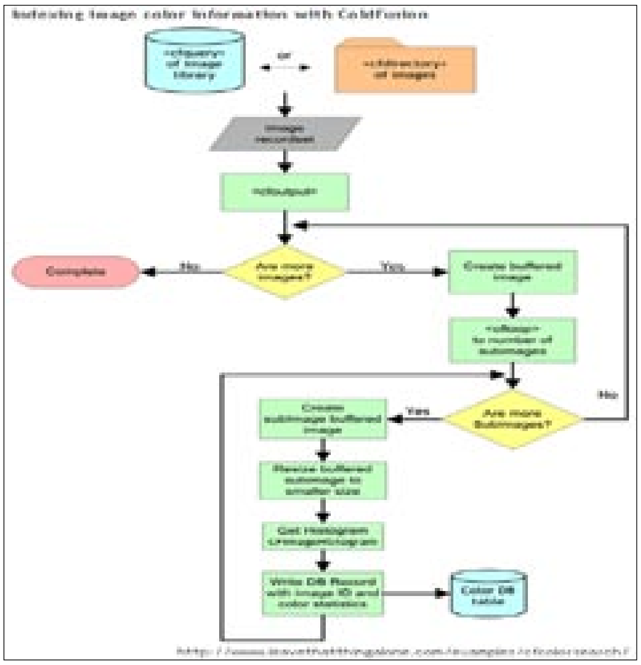

Indexing image color information with ColdFusion
<cfquery> of image library
or
<cfdirectory> of images
Image recordset
<cfoutput>
Are more images?
No
Complete
Yes
Create buffered image
<cfloop> to number of subimages
Are more Subimages?
Yes
No
Create subimage buffered image
Resize buffered subimage to smaller size
Get Histogram CFImageHistogram
Write DB Record with image ID and color statistics
Color DB table
http://www.leavethatthingalone.com/examples/cfcolorsearch/

Figure 3: Flow chart of image index process

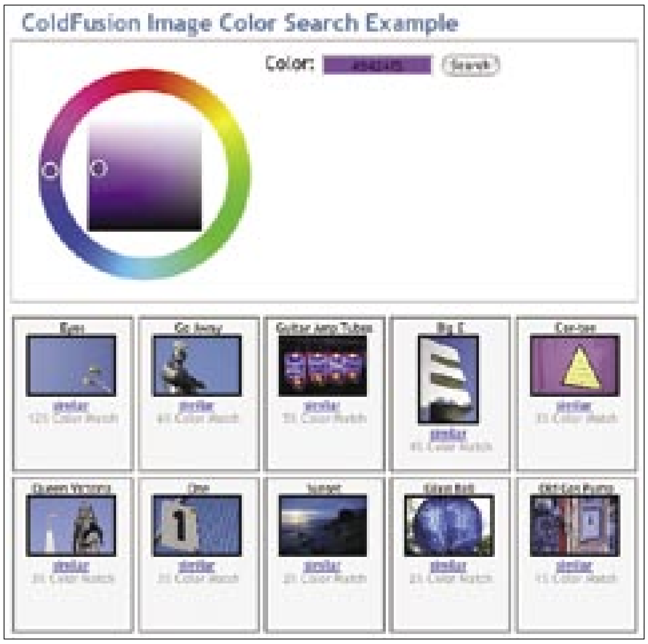


Figure 4: Demo color search

We need to store this image color histogram information in a database table so that we can query it. We'll also need to store a unique ID of the image (either a file name or database ID) and the means and standard deviations. CFImageHistogram will return a struct containing the histogram and color statistics for the R-G-B of the image. Here is an example of what that query might look Listing 1.

While looping through each of the 16 sub-images, we need to store this information based on a unique ID or the filename of this image so we can later query this data.

Putting the above code snippets together we can index images. Listing 2 provides a more in-depth example of how to index a database or cfdirectory of images, storing color data for each sub-image within each image.

This process of storing the data of each sub-image needs to be repeated for all the images you would like to search. You can do this within a cfoutput of a cfdirectory or a query result of image locations. This process can be very slow; each image may take as long as 1–5 seconds to process depending on the size of the original image, the size and number of sub-images, and your server's speed. This can also create a large number of records, for example, 200 images at 16 bins per image = 3200 records.

Searching the Indexed Images

Once we have a database table populated with image region color information, we need to figure out how to search for a color. We'll do this based on a single color and look for images that have the most occurrences (or nearest) in their sub areas. We'll query the database looking for rows that are closest to the R-G-B color selected. We'll use a SQL "between".

That query will return all the matches of the colors that fall in those ranges, which is fine but we'd like to know which images have the most occurrences. We will add some grouping and sorting to the query so that we return, in order, the images that have the highest occurrence of the color selected (see Listing 3). Chances are fairly high that most selected colors won't return any exact matches to the selected color, so you may need to add more range that the record's color mean can fall into. You may even want to loop the query several times, each time expanding the range until you have, say, 10 results.

To view a demo color search (see Figure 4) go to: http://www.leavethatthingalone.com/examples/cfcolorsearch/index.cfm.

Final Notes

This page article is meant as a starting point. There is much more that could be done and/or improved with this method. The search query could definitely be improved to better take into account the standard deviation of sub-images. Also since we are storing colors in areas of an image, it would be possible to search images based on patterns or drawings similar to the way retrievr (http://labs.systemone.at/retrievr/) works.

About the Author

Seth Duffey is the Webmaster for the City of Davis, California. He is also the manager of the Sacramento ColdFusion Users Group. His blog is http://www.leavethatthingalone.com.

sethduffey@gmail.com

Adobe | Solution PARTNER

INTERGRAL information solutions

</cf_bugs>

2.0

FusionDebug™ - The Interactive CF Debugger

Faster. Easier. Smarter.

fusion debug™ 2.0

www.fusiondebug.com

Announcing FusionDebug™ 2.0

- installers for Windows, MacOS, Linux...
- improved user interface layout
- easy to use configuration tool
- improved support for CF frameworks
- step though CF pages, tags and CFC's
- set breakpoints
- view and modify variables
- watch expressions

Listing 1

```
<!--- histogram struct name = "hist" --->
<cfquery datasource="#DSN#">
INSERT INTO colors (
photoid,
redmean,
redsd,
greenmean,
greensd,
bluemean,
bluesd,
bin
)
VALUES (
<cfqueryparam value="#photoid#" cfsqltype="cf_sql_char" />,
<cfqueryparam value="#hist.r.mean#" cfsqltype="cf_sql_double" />,
<cfqueryparam value="#hist.r.standarddeviation#" cfsqltype="cf_sql_dou-
ble" />,
<cfqueryparam value="#hist.g.mean#" cfsqltype="cf_sql_double" />,
<cfqueryparam value="#hist.g.standarddeviation#" cfsqltype="cf_sql_dou-
ble" />,
<cfqueryparam value="#hist.b.mean#" cfsqltype="cf_sql_double" />,
<cfqueryparam value="#hist.b.standarddeviation#" cfsqltype="cf_sql_dou-
ble" />,
<cfqueryparam value="#bin#" cfsqltype="cf_sql_numeric" />
)
</cfquery>
```

Listing 2

```
<!--- cfquery of cfdirectory here --->
<cfoutput query="photos">
 <cfscript>
  //size of sub images (WARNING this number will be squared so 4 = 16
subimages)
  size = 4;
  //max size in pixels of subimage (saves processing time)
  maxSize = 50;
  //Jpeg Codec
  jpegCodec = createObject("java", "com.sun.image.codec.jpeg.JPEGCo-
dec");
  //file open
  fileInputStream = createObject("java", "java.io.FileInputStream").
init("#directory#\#photos.photofilename#");
  //decodes the JPEG
  decoder = jpegCodec.createJPEGDecoder(fileInputStream);
  //give us an image to scale
  image = decoder.decodeAsBufferedImage();
  imageHeight = image.getHeight();
  imageWidth = image.getWidth();
 </cfscript>
 <cfset bin = 0 />
 <cfloop from="1" to="#size#" index="i">
  <cfloop from="1" to="#size#" index="ii">
   <cfset bin = bin + 1 />
   <cfscript>
    //get subimage
    x = imageWidth/size*(ii-1);
    y = imageHeight/size*(i-1);
    subImage = image.getSubimage(JavaCast("int",x),JavaCast("int",y),Ja
vaCast("int",imageWidth/size),JavaCast("int",imageHeight/size));

    subImageHeight = subImage.getHeight();
    subImageWidth = subImage.getWidth();
    sizeRatio = subImageWidth/subImageHeight;

    //figure out if the photo is portrait or landscape make the larger
equal to the max thumb size given in agruments
    if (subImageWidth gte subImageHeight) {
     subImageWidth = maxSize;
     subImageHeight = round(maxSize/sizeRatio);
    } else {
     subImageWidth = round(maxSize * sizeRatio);
     subImageHeight = maxSize;
    };

    //create a scaled image
    scaledImage = subImage.getScaledInstance(JavaCast("int", subImage-
Width), JavaCast("int", subImageHeight), subImage.SCALE_SMOOTH);
    BufferedImage = createObject("java", "java.awt.image.BufferedIm-
age").init(JavaCast("int", subImageWidth), JavaCast("int", subImage-
Height), subImage.TYPE_INT_RGB);
    //draw the image into the buffered image
    graphics = BufferedImage.createGraphics();
    graphics.drawImage(scaledImage, 0, 0, Javacast("null", ""));

    //get histogram
    imageHistogram.setBufferedImage(BufferedImage);
    hist = imageHistogram.getColorHistogram();
   </cfscript>

   <cfquery datasource="#DSN#">
   INSERT INTO colors (
   photoid,
   redmean,
   redsd,
   greenmean,
   greensd,
   bluemean,
   bluesd,
   bin
   )
   VALUES (
   <cfqueryparam value="#photos.photoid#" cfsqltype="cf_sql_char" />,
   <cfqueryparam value="#hist.r.mean#" cfsqltype="cf_sql_double" />,
   <cfqueryparam value="#hist.r.standarddeviation#" cfsqltype="cf_sql_
double" />,
   <cfqueryparam value="#hist.g.mean#" cfsqltype="cf_sql_double" />,
   <cfqueryparam value="#hist.g.standarddeviation#" cfsqltype="cf_sql_
double" />,
   <cfqueryparam value="#hist.b.mean#" cfsqltype="cf_sql_double" />,
   <cfqueryparam value="#hist.b.standarddeviation#" cfsqltype="cf_sql_
double" />,
   <cfqueryparam value="#bin#" cfsqltype="cf_sql_numeric" />
   )
   </cfquery>
  </cfloop>
 </cfloop>
</cfoutput>
```

Listing 3

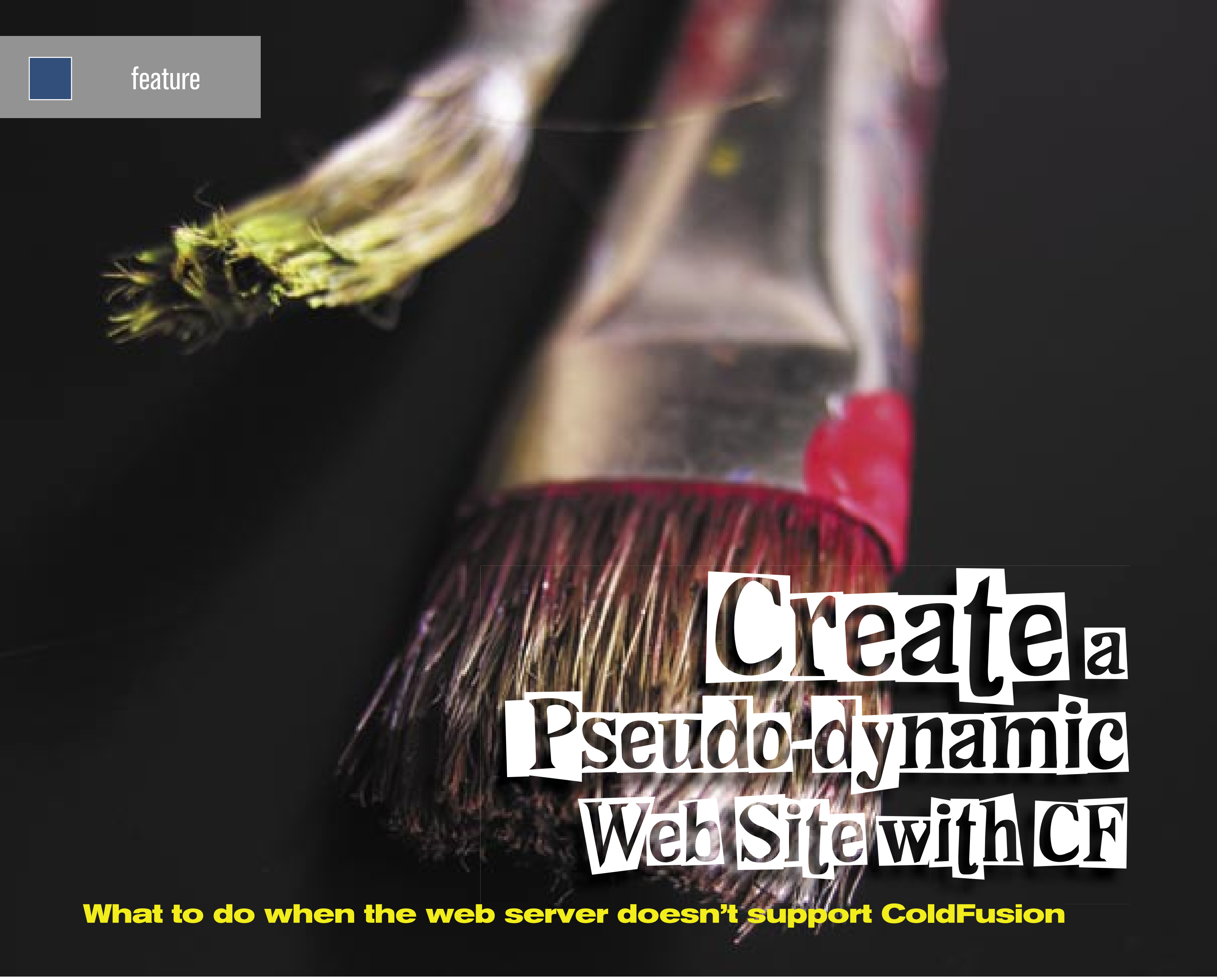
```
<!--- set range of colors to fall in based on mean --->
<cfset sd = 10 />
<!--- query for colors expecting red,green,blue 0-255 from form--->
<cfquery datasource="#datasource#" name="search">
SELECT Count(colors.photoid) AS CountOfphotoid,cfcphotoblogphotos.pho-
toid , cfcphotoblogphotos.photoTitle,cfcphotoblogphotos.photoThumbFile-
Name, avg(colors.redmean) as redmean, avg(colors.bluemean) as bluemean,
avg(colors.greenmean) as greenmean
FROM cfcphotoblogphotos RIGHT JOIN colors ON cfcphotoblogphotos.photoID
= colors.photoid
WHERE 1=1
AND colors.redmean between <cfqueryparam value="#form.red-sd#"
cfsqltype="cf_sql_numeric" />  and <cfqueryparam value="#form.red+sd#"
cfsqltype="cf_sql_numeric" />
AND colors.greenmean between <cfqueryparam value="#form.green-
sd#" cfsqltype="cf_sql_numeric" />  and <cfqueryparam value="#form.
green+sd#" cfsqltype="cf_sql_numeric" />
AND colors.bluemean between <cfqueryparam value="#form.blue-sd#"
cfsqltype="cf_sql_numeric" />  and <cfqueryparam value="#form.blue+sd#"
cfsqltype="cf_sql_numeric" />
GROUP BY colors.photoid, cfcphotoblogphotos.photoTitle
ORDER BY Count(colors.photoid) DESC;
</cfquery>
```

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

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.

HostMySite.com

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



Create a Pseudo-dynamic Web Site with CF

What to do when the web server doesn't support ColdFusion

By Michael H. Markowski

You have to love a software product that makes developers, even relatively inexperienced ones, productive quickly and empowers them to do seemingly impossible things. The more I work with and learn about ColdFusion the more it seems as if there are no limits to what can be done with it.

I've experienced the "ColdFusion can do anything" effect on a number of occasions after having used it for some programming task that I previously thought was impossible. Just when it seemed I'd hit a dead end and was about to give up, a quick look through the ColdFusion documentation or some other resource produced a tag or function that was just what I needed. And after a few minutes of learning how to leverage my newly discovered tag (or function), the problem was solved and the supposedly impossible task was thrown onto the pile with all the other "impossible" hurdles that I've conquered with ColdFusion. This article will describe how I used ColdFusion to do one such task that, at the time, seemed impossible to me.

Mission: Possible?

As a webmaster for the Air Protection Division at EPA Region 3 I wear many hats. Besides being a ColdFusion developer responsible for keeping our dynamic data-driven intranet applications up and running, I'm also required to develop/maintain a number of static Web sites on which the content doesn't change all that much. By "static" I mean a Web site that resides on a Web server without any ColdFusion (or database) support. I was recently asked by someone in my organization if there was a way to publish data from a large internal database on a static Web site in the form of HTML Web pages. The data for each record in the database would have to be displayed on a single static Web page, so there would be one page for each database record. The data would need to be updated only occasionally, but the site would have to be created in HTML format to ensure maximum accessibility, usability, and consistency with the rest of the site. Oh, and did I mention that the database contained over 700 records? That's over 700 static Web pages that had to be created and then populated with unique content. My initial reaction was "That's impossible" but, as I've come to expect, ColdFusion would prove me wrong and enable me to do the impossible.

Of course, writing and deploying a small ColdFusion application would be the easiest and fastest way for me to address this problem since I am, after all, a ColdFusion developer and this would be a fairly straightforward data drill-down application to write. However, there was a serious problem with the "let's use ColdFusion" approach: there was no ColdFusion (or database) support of any kind on the destination Web server, nor was there likely to be in the future. Therefore, what I needed was a way to create hundreds (or thousands) of static Web pages from the database information automatically and rapidly so that the resulting pages could be served statically. I could use ColdFusion (or any technology) to create the HTML pages that comprise the static Web site, but I couldn't build an actual ColdFusion application because the server lacked ColdFusion support. Now that I think about it, situations like this probably occur all the time because many organizations that, for whatever the reason, don't have ColdFusion support on their Web site still have data that has to be published on those sites. And they can't just upload a large database or spreadsheet file to the site and be done with it because such a "solution" would be inaccessible and pretty much unusable. In my case the final product had to be user-friendly, accessible, easily navigable, and logically organized. In other words, it had to be a standard HTML Web site consisting of several hundred static Web pages.

If You Build It, They Will Come

After a lot of experimentation and a fair amount of trial and error, I discovered that it's actually possible to automatically create an entire static Web site consisting of several hundred unique Web pages using ColdFusion, a database, and a few key CFML tags. Using the technique I developed, static Web pages are constructed "on the fly" from database records, one page per database record, and then written to the ColdFusion server as HTML files. I also realized that I could mimic the functionality of a dynamic data drill-down application in the static Web site by creating a "master" static Web page containing links to the many "details" pages on the site. I call the resulting Web site a "pseudo-dynamic" site because, although it's comprised entirely of static Web pages, the pages are all created (and maintained) automatically with ColdFusion MX 7.

There are two primary ColdFusion tags that enable the creation of documents that can be served statically via the Web: <CFFILE> and <CFDOCUMENT>. The former has been around forever and is extremely versatile. It can be used to create almost any type of text file, including an HTML file. The <CFDOCUMENT> tag is relatively new and is supported only in ColdFusion MX 7; it's used to create PDF and Flashpaper documents. As this article will demonstrate, <CFFILE> and <CFDOCUMENT> can be used to generate any number of unique HTML, PDF, or Flashpaper documents from database information.

ColdFusion File Writing 101

I could write a book about the many and varied uses of the <CFFILE> tag. The <CFFILE> tag has a multitude of capabilities by virtue of its "action" attribute. The action attribute determines how the tag functions and what it can do. When the action attribute is set to "write," the <CFFILE> tag can create a file and write it to a directory on the ColdFusion server. Other valid values of the action attribute include read, copy, move, upload, and append. However, we're only concerned with the tag's "write" attribute here. When writing files to the ColdFusion server with <CFFILE>, the other required attributes of the tag are "file" and "output." As you might expect, the "file" attribute defines the full path and filename of the file you're writing to the server. The content of a file created with <CFFILE> is defined using the "output" attribute of the tag. The value for this attribute can be as simple as a one-word text string or as long and complex as the HTML code for an entire Web page. In other words, by specifying HTML code in its "output" attribute, <CFFILE> can create a static Web page.

The <CFFILE> tag's "output" attribute defines the actual content of the file you're creating. The example code below creates a simple text file named "myCreatedFile.txt," populates it with the customary "Hello World" message, and then writes the file to the ColdFusion server with the filename "myCreatedFile.txt":

```
<CFFILE ACTION="WRITE" FILE="C:\Inetpub\wwwroot\mySite\myCreatedFile.txt"
OUTPUT="Hello World!">
```

As shown here, the value of the <CFFILE> tag's "file" attribute consists of the full path to the file you're creating on the server, plus the filename.

Now, let's get back to the task at hand which is, as you'll recall, creating an entire static Web site automatically with ColdFusion. What if we want to use <CFFILE> to create an HTML file instead of a text file? This is not much harder to do. The main difference is that we first use the <CFSAVECONTENT> tag to assign the necessary HTML code to a variable, and then make that variable the value of the <CFFILE> tag's "output" attribute. This sounds a lot more complicated than it really is, so an example should help. Look at the code Listing 1.

Here the <CFSAVECONTENT> tag "stores" the HTML code needed to create our output file in the "myHTMLCode" variable. Then, the <CFFILE> tag writes a file named "myCreatedFile.html" to the server using the code specified in the "myHtmlCode" variable as the file's content. <CFSAVECONTENT> is invaluable in this situation because it lets us save a long complex text string – HTML code in this case – in a variable that can then be used as the value of the "output" attribute in <CFFILE>. This keeps the code neatly organized and readable. Also, unlike the <CFSET> tag, we don't need to escape quotes or worry about other special characters that may be present in the HTML code because the <CFSAVECONTENT> tag automatically takes care of that for us.

We're not limited to storing HTML code in a variable defined by the <CFSAVECONTENT> tag. We can also specify JavaScript and/or Cascading Style Sheet (CSS) syntax in variables defined by <CFSAVECONTENT>. Better yet, we can use CFML to create dynamic content in HTML files that are written to the server with <CFFILE>. Taking our simple example a step further, let's add some CFML to the <CFSAVECONTENT> tag in Listing 1 to display the current date in our output file. This is shown in Listing 2.

Notice how we're able to use both HTML and CFML between the opening and closing <CFSAVECONTENT> tags in Listing 2. ColdFusion will evaluate the CFML between the opening and closing <CFOUTPUT> tags and then treat the resulting code as one long text string. It then assigns this string to the "myHtmlCode" variable. It's like we're "injecting" the HTML and CFML that's needed to create our file into the <CFFILE> tag's "output" attribute. When the <CFFILE> tag executes it will, once again, use

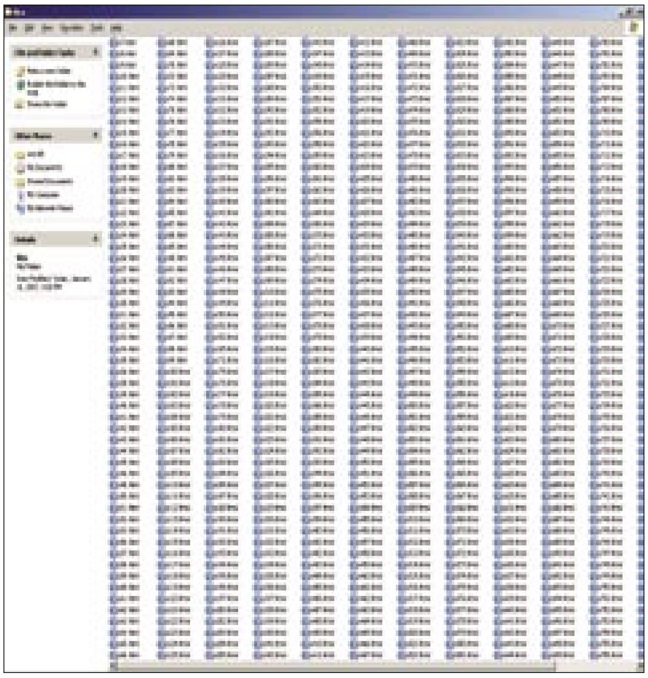


Figure 1: ColdFusion-generated HTML files

the source code that's stored in the "myHtmlCode" variable to create the HTML file and write it to disk. When loaded in a browser this page will display the message "Hello World, the date is [today's date]!" The <CFSAVECONTENT> tag lets us use CFML and achieve precise control over the code that's written to our output file.

There is another important aspect of <CFFILE> that you should know about. When <CFFILE> is instructed to write a file that already exists to the server, the original file is overwritten and replaced with the new one. So be careful when using <CFFILE> because you can overwrite important files on your server. I always make a special directory on the server to hold my ColdFusion-generated files and confine my file writing activities to that directory. That way I can be reasonably sure that nothing really important is being overwritten on the ColdFusion server.

Of course, it's also possible to retrieve data from a database and put it in the <CFSAVECONTENT> tag to include in a file that's subsequently written to the server with <CFFILE>. Conceptually, this isn't much different from the example in Listing 2 since we'd just be substituting query results for the current date and writing a single output file to the server. However, getting back to the task at hand, what if we wanted to generate a hundred or even a thousand static Web pages from information in a database and mimic the functionality of a dynamic data drill-down application using only static Web pages? By now it should come as no surprise that even these tasks aren't too difficult. After all, this is ColdFusion we're using here! The code in Listing 3 does this seemingly impossible task by putting the <CFSAVECONTENT> and <CFFILE> tags in a <CFLOOP> tag.

There's a lot of code in Listing 3, so let's get right into it. The code loops over the "listAllRecords" query executing the <CFSAVECONTENT> and <CFFILE> tags once for each row retrieved by the "listAllRecords" query. With each pass through the loop, the CFML inside the opening and closing <CFOUTPUT> tags is evaluated and the resulting HTML source code that comprises that HTML file is assigned to the "myHtmlCode" variable as a text string. In this case we're outputting, within each HTML file, the Project ID, Name, Description, and Last Modified date values for each record retrieved by the "listAllRecords" query. The myHtmlCode variable, which holds the HTML source code for each file, becomes the value of the "output" attribute in the <CFFILE> tag and the resulting HTML file is written to the "files" directory on the server. Thanks to the <CFLOOP> tag, this process is repeated for every record retrieved by the "listAllRecords" query. Each static Web page written to the server is unique because it's populated with the data from a single row in the database. Notice the code in the "file" attribute in the <CFFILE> tag. This code defines the full path (on the server) and filename of the HTML files that are written to disk. It's dynamic and will change with each pass through the loop. This means that the files (i.e., Web pages) that are written to the "files" directory will be assigned sequential filenames comprised of a "p" concatenated with the "projectid" value and will have an ".html" extension. This will produce filenames such as "p1.html," "p2.html," "p3.html," etc. for the files written to the server. In other words, the filenames are sequential and correspond to the "projectid" values in the database.

Now I'm a Believer

I must admit the first time I ran the code in Listing 3 in a Web browser I was doubtful it would work. I fully expected it to crash and give me a dozen error messages pointing out the flaws in my code and describing why it was impossible to do what I'd asked it to do. Imagine my surprise when the template, which I'd aptly named "pageGenerator.cfm," executed on the first attempt without any errors and sat there waiting for me. At first I thought that it hadn't done anything much less written hundreds of HTML files to the server, because it executed so fast. But when I checked the "files" directory on the server –which served as the repository for all of my ColdFusion-generated files – I found more than 700 HTML files there! (See Figure 1.) Moreover, each file had a unique sequential filename (based on the "projectid" parameter), syntactically correct HTML source code, and unique content derived from the "myProjects" database. So, the bottom line is that ColdFusion enabled me to create, in a matter of seconds, hundreds of static Web pages populated with data from a database. If that's not working efficiently and making the best use of my time, then I don't know what is.

Remember the basic data drill-down application I mentioned earlier? It would now be possible to mimic this functionality – using static pages – by creating a "master" static Web page containing links to each of the "details" pages I'd just created. The master static Web page, which is created with ColdFusion (of course), lists three database column values: Project ID, Project Name, and Last Modified Date. The “Project ID” values serve as links to the corresponding “details” pages. These links are created by concatenating a “p” with the “projectid” value as shown in Listing 4.

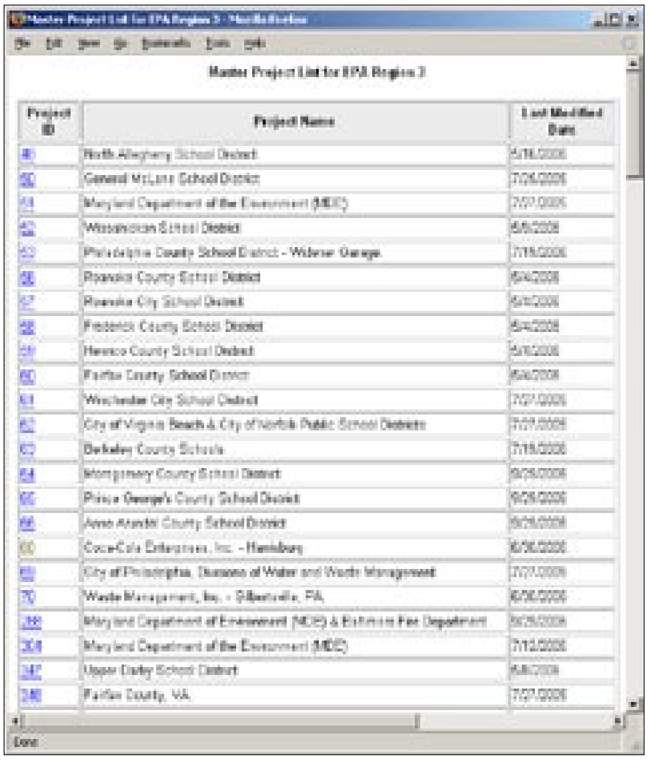
When this code is run, a static Web page named “masterpage.html” is created and written to the “files” directory on the server. This page displays the Project ID, Project Name, and Last Modified Date for every record retrieved by the “listAllRecords” query. The Project ID column values are linked to the corresponding “details” pages. Remember how the filenames of the 700-odd “details” pages were derived from the Project ID values? It’s the same thing here only this time we’re creating a single Web page with links to each of the 700+ “details” pages. There’s a direct correlation between the links in the “master” page and the filenames of the details pages, i.e., they are both created from the “projectid” values. So the details of any record can be viewed by clicking on the appropriate “Project ID” link in the master page, as shown in Figure 2. Using just two ColdFusion MX 7 templates (Listings 3 and 4) and a database, we can automatically create a large static Web site – with basic data drill-down functionality – in a matter of seconds. As a final touch and to aid navigation within the static site, a simple HTML link back to the master page can be put on each details page.

There are some significant benefits to using <CFFILE> to create static Web pages using ColdFusion. First, it doesn’t matter how many database records there are. Granted, it will take ColdFusion a little longer to process 700 database records (and write the corresponding HTML files) than it will to process just a few, but ColdFusion and the database are still doing all of the work, not me. Plus it’s getting done much faster than I could ever do it. Second, by modifying the query parameters it’s easy to control what static files are actually generated and written to disk. For example, I could have easily used a WHERE clause in the “listAllRecords” query in Listing 3 to limit the query results to only those records that meet certain criteria. Then only Web pages representing the retrieved records would have actually been generated and written to the server. In fact, this is an excellent way to update and maintain a pseudo-dynamic Web site. For example, assuming your database has a “Last Modified Date” column (indicating the date/time each record was last updated), you can easily limit the query with a WHERE clause to only those records that were modified during the past day, week, month, etc. Then, the only static pages that would be generated by <CFFILE> (and written to the server) would be those that contain data that was modified in the past day, week, or month. You could then overwrite the old outdated static Web pages with the new ones on your server, and your pseudo-dynamic site would be updated with only a minimal effort on your part. Of course, you’ll have to upload (FTP) any changed files to the server manually but you’d need to do that anyway with a static Web site.

Another benefit of creating and deploying a pseudo-dynamic Web site is better performance. While the performance of a properly written dynamic ColdFusion application is very good, it’s not going to surpass the performance of a static Web site. With this technique ColdFusion is used to create batches of files that are manually uploaded to a Web server and served statically. So the performance penalties normally associated with dynamic Web sites are avoided. In other words, you will always get better performance from a pseudo-dynamic (i.e., static) Web site than you will from a dynamic Web application even if ColdFusion powers that application.

The fourth and arguably greatest benefit of this technique stems from the use of Dreamweaver 8 templates. Being responsible for several static Web sites, I always use Dreamweaver 8’s “templates” feature so that whenever I need to make a routine change to a static site, all I need do is change the Dreamweaver template and all of the pages in the site (that use that template) are changed automatically. Therefore, when using <CFFILE> to write static Web pages to the ColdFusion server, I make certain to retain all Dreamweaver template references in the HTML source code. Dreamweaver embeds template references in the HTML source code as HTML comments. Keeping these template references in the HTML source code ensures that my ColdFusion-generated static files will be “linked” to the Dreamweaver template. This means that when (not if) there is a need to revise the Dreamweaver template to accommodate some routine change in the HTML source code, all of the static pages that were generated with <CFFILE> will be changed automatically by Dreamweaver. This is a terrific ancillary benefit because it saves me (or someone else) from having to recreate all of the ColdFusion-generated static pages from scratch just to accommodate routine changes to the HTML source code.

As I mentioned earlier, the <CFDOCUMENT> tag can be used to write PDF and Flashpaper documents to a server. By modifying the <CFFILE> technique, it’s possible to create any number of unique PDF or Flashpaper documents from database information in the same way that HTML files are created from database information using <CFFILE>. This time, however, we loop over the <CFDOCUMENT> tag (not <CFFILE>) and specify



Master Project List for EPA Region 3

Project ID	Project Name	Last Modified Date
49	North Allegheny School District	5/18/2006
50	General McLane School District	7/26/2006
51	Maryland Department of the Environment (MDE)	7/27/2006
52	Wissahickon School District	6/6/2006
53	Philadelphia County School District - Widener Garage	7/18/2006
56	Roanoke County School District	6/4/2006
57	Roanoke City School District	6/1/2006
58	Frederick County School District	6/4/2006
59	Henrico County School District	5/8/2006
60	Fairfax County School District	6/6/2006
61	Winchester City School District	7/27/2006
62	City of Virginia Beach & City of Norfolk Public School Districts	7/27/2006
63	Berkeley County Schools	7/18/2006
64	Montgomery County School District	9/26/2006
65	Prince George's County School District	9/26/2006
66	Anne Arundel County School District	9/26/2006
67	Coca-Cola Enterprises, Inc. – Harrisburg	6/30/2006
69	City of Philadelphia, Divisions of Water and Waste Management	7/27/2006
70	Waste Management, Inc. - Gilbertsville, PA	6/30/2006
288	Maryland Department of Environment (MDE) & Baltimore Fire Department	9/26/2006
304	Maryland Department of the Environment (MDE)	7/12/2006
347	Upper Darby School District	6/8/2006
348	Fairfax County, VA	7/27/2006

Figure 2: Master Project List for EPA Regon 3

the filenames via the <CFDOCUMENT> tag's "filename" attribute as shown in Listing 5.

Note that this code omits the <CFSAVECONTENT> and <CFFILE> tags since they are no longer needed. Conceptually this code is the same as Listing 3 because we're still looping over a query and writing a single output file to the ColdFusion server on each pass through the loop. The "filename" attribute of the <CFDOCUMENT> tag is used to specify the full path and filename of the PDF files to be written to the server. As we did before with <CFFILE>, we dynamically construct sequential filenames for the PDF output files and specify "PDF" as the output format in the <CFDOCUMENT> tag. This results in filenames such as "p1.pdf," "p2.pdf," etc. The "overwrite" attribute of the <CFDOCUMENT> tag enables us to overwrite and replace any existing PDF files that may reside in the target directory on the server. The code in Listing 5 will create a unique PDF document for each record retrieved by the "listAllRecords" query; each PDF file will be populated with the data from a single database record and then written to disk.

You may receive the error message "The request has exceeded the allowable time limit" when executing an exceptionally long running template such as the code in Listing 5. To remedy this, you can use the "requesttimeout" attribute of the <CFSETTING> tag to increase the timeout limit for the template and override the ColdFusion Administrator's default timeout setting. Just insert the following code at the top of the template:

```
 <CFSETTING REQUESTTIMEOUT = "360"
ENABLECFOUTPUTONLY = "no">
```

This will reset the timeout limit for the template to six minutes (i.e., 360 seconds) and preserve the ColdFusion Administrator's default timeout setting for all other templates on the server.

Conclusion

ColdFusion's <CFDOCUMENT> and <CFFILE> tags enable the rapid creation of static files, and each file can be populated with unique data from a database. This technique can be used to create any number of HTML, PDF, or Flashpaper documents for immediate deployment on a static (non-ColdFusion) Web site. If you have database information you need to display on a Web site but no ColdFusion support on the site, you could use this technique to create as many static files as there are records in your database. You could then upload those files and serve them statically from your site. Advanced ColdFusion developers could even implement an Event Gateway or employ the <CFSCHEDULE> tag to periodically generate a fresh set of static files whenever the database information changes or according to some preset schedule. Then it's just a matter of replacing the old files on the Web server with the new ones. Web servers that have ColdFusion support may also benefit because files (especially PDF and Flashpaper documents) could be periodically batch-generated and uploaded to the server rather than being served dynamically in real-time. This would lighten the ColdFusion server's processing load and most likely improve performance.

Needless to say, both the <CFFILE> and <CFDOCUMENT> tags are extremely powerful and important development tools. They come in very handy when you're a webmaster like me and responsible for one or more static Web sites that have no ColdFusion support. In such situations you need to get creative with ColdFusion or you could find yourself hand-coding hundreds (or thousands) of static Web pages. Obviously, this would be a mind-numbing, time-consuming exercise even if you use Dreamweaver's templating feature. Whenever you need to publish static text, HTML, PDF, or Flashpaper documents, the <CFFILE> and <CFDOCUMENT> tags should come to mind. Using them is simplicity itself, and the benefits can be enormous. As for me, I was able to accomplish another "impossible" task with ColdFusion MX 7 and save precious time for other development projects. So all of the harried, overworked ColdFusion developers out there should relax, take a deep breath, and repeat after me: "ColdFusion can do anything."

About the Author

Michael Markowski works for the Air Protection Division at the Environmental Protection Agency and is a Macromedia/Adobe Certified Professional.

markowski.mike@epamail.epa.gov

Listing 1

```
<!--- Write an HTML file to disk using CFFILE --->
<CFSAVECONTENT VARIABLE="myHtmlCode">
<html>
<head><title>A CFFILE example</title></head>
<body>
<h2>Hello World!</h2>
</body>
</html>
</CFSAVECONTENT>

<CFFILE ACTION="write" FILE="C:\Inetpub\wwwroot\mySite\myCreatedFile.
html"
OUTPUT="#myHTMLCode#">
```

Listing 2

```
<!--- Include dynamic content in a file written to disk with CFFILE --->
<CFSAVECONTENT VARIABLE="myHtmlCode">
<html>
<head><title>A CFFILE example</title></head>
<body>
<CFOUTPUT>
<h2>Hello World, the date is #DateFormat(Now(), "m/d/yy")#!</h2>
</CFOUTPUT>
</body>
</html>
</CFSAVECONTENT>

<CFFILE ACTION="write" FILE="C:\Inetpub\wwwroot\mySite\myCreatedFile.
html"
OUTPUT="#myHtmlCode#">
```

Listing 3

```
<!--- Create 700+ unique "details" pages from the database data --->
<CFQUERY NAME="listAllRecords" DATASOURCE="myProjects">
SELECT projectid, projectname, projectdesc, lastmodified
FROM projects
</CFQUERY>

<CFLOOP QUERY="listAllRecords">
```

STREAM57

Customized Flash-based media solutions

Software and services for collaboration, video conferencing, and e-learning

visit us
stream57.com

e-mail us
streamline@stream57.com

call us
212.909.2550 x1012

```
<CFSAVECONTENT VARIABLE="myHtmlCode">
<html>
<head><title>Project Details</title></head>
<body>
<CFOUTPUT>
<h4>Project Details</h4>
<!--- "projectid" is an automatically incremented integer (& primary
key) generated by the database --->
<p>Project ID: #listAllRecords.projectid#</p>
<p>Project Name: #listAllRecords.projectname#</p>
<p>Project Description: #Trim(listAllRecords.projectdesc)#</p>
<p>Last Modified: #DateFormat(listAllRecords.lastmodified, "mm/dd/yy")#</
p>
</CFOUTPUT>
</body>
</html>
</CFSAVECONTENT>

<CFFILE ACTION="WRITE" FILE="C:\Inetpub\wwwroot\myProject\files\
p#listAllRecords.projectid#.html"
OUTPUT="#myHtmlCode#">

</CFLOOP>
```

Listing 4

```
<!--- Create the "master" static Web page with a link to each "details"
page --->
<CFQUERY NAME="listAllRecords" DATASOURCE="myProjects">
SELECT projectid, projectname, lastmodified
FROM projects
<!--- for demo, limit the query to Region 3 records only --->
WHERE region = 3
ORDER BY projectid
</CFQUERY>

<CFSAVECONTENT variable="pageContent">
<html>
<head><title>Master Project List for EPA Region 3</title></head>
<body>
<h4>Master Project List for EPA Region 3</h4>
<table cellpadding="2" border="1">
<tr>
  <th>Project ID</th>
  <th>Project Name</th>
  <th>Last Modified Date</th>
</tr>
<CFOUTPUT QUERY="listAllRecords">
<tr>
  <!--- Create a link to each "details" page using projectid parameter-
-->
  <td><a href="p#projectid#.html">#projectid#</a></td>
  <td>#projectname#</td>
  <td>#DateFormat(lastmodified, "mm/dd/yy")#</td>
</tr>
</CFOUTPUT>
</table>
</body>
</html>
</CFSAVECONTENT>

<CFFILE ACTION="write" FILE="C:\Inetpub\wwwroot\myProject\files\masterp-
age.html"
      OUTPUT="#pageContent#">
```

Listing 5

```
<CFQUERY NAME="listAllRecords" DATASOURCE="myProjects">
SELECT *
FROM projects
</CFQUERY>

<CFLOOP QUERY="listAllRecords">
<CFDOCUMENT FORMAT="PDF" FILENAME="C:\Inetpub\wwwroot\myProject\files\
p#listAllRecords.projectid#.pdf"
OVERWRITE="yes">
<html>
<head><title>Project Details</title></head>
<body>
<h4>Project Details</h4>
<CFOUTPUT>
<p>Project ID: #listAllRecords.projectid#</p>
<p>Project Name: #listAllRecords.projectname#</p>
<p>Project Description: #Trim(listAllRecords.projectdesc)#</p>
<p>Last Modified: #DateFormat(listAllRecords.lastmodified, "mm/dd/yy")#</
p>
</CFOUTPUT>
</body>
</html>
</CFDOCUMENT>
</CFLOOP>
```

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

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
Adobe	www.adobe.com/creativemind		2,3
AJAXWorld East	www.ajaxseminar.com	201-802-3020	29
CFDynamics	www.cfdynamics.com	866-233-9626	35
CFDJ	http://coldfusion.sys-con.com/	800-303-5282	33
Cynergy Systems	www.cynergysystems.com		12,13
EdgeWebHosting	edgewebhosting.net	1-866-334-3932	4
Hal Helms	halhelms.com		15
HostMySite	www.hostmysite.com	877-215-4678	21,36
Hotbanana	hotbana.com/cfdj-demo	866-296-1803	6
Intergral	www.fusiondebug.com		19
Stream57	www.stream57.com	212-909-2550	27
TeraTech	www.teratech.com		11
VitalStream	www.vitalstream.com/tools/mxdj.asp	800-254-7554	14-15

REGISTER TODAY AND $AVE!

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

www.AjaxWorldExpo.com

AjaxWorld™ East Conference & Expo

NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

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

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

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

BEING HELD MARCH 19 - 21, 2007!

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

ColdFusion

For more information go to...

U.S.

Alabama
Huntsville, AL CFUG
www.nacfug.com

Arizona
Phoenix, AZ CFUG
www.azcfug.com

California
Bay Area CFUG
www.bacfug.net

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

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

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

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

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

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

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

Florida
South Florida CFUG
www.cfug-sfl.org

Georgia
Atlanta, GA CFUG
www.acfug.org

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

Indiana
Indianapolis, IN CFUG
www.hoosierfusion.com

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

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

Maryland
Maryland CFUG
www.cfug-md.org

Massachusetts
Boston CFUG
http://bostoncfug.org/

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

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

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

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

Minnesota
Twin Cities CFUG
www.colderfusion.com

Missouri
Kansas City, MO CFUG
www.kcfusion.org

Nebraska
Omaha, NE CFUG
www.necfug.com

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

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

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

New York
Albany, NY CFUG
www.anycfug.org

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

New York
Syracuse, NY CFUG
www.cfugcny.org

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

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

Oregon
Portland, OR CFUG
www.pdxcfug.org

Pennsylvania
Central Penn CFUG
www.centralpenncfug.org

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

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

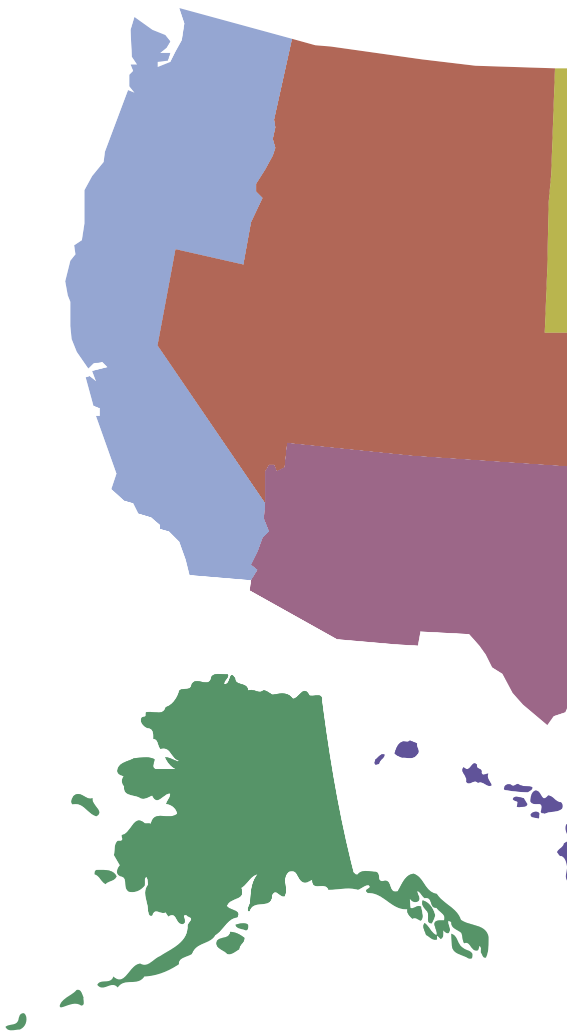
Tennessee
Nashville, TN CFUG
http://www.ncfug.com

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

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

Texas
Dallas, TX CFUG
www.dfwcfug.org/

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



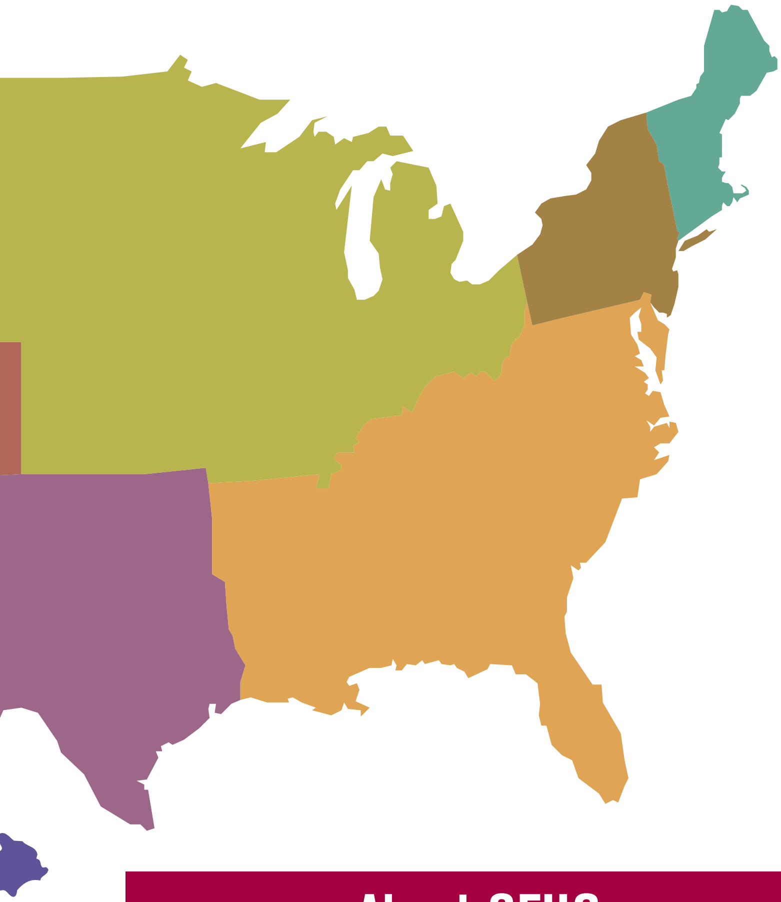
Utah
Salt Lake City, UT CFUG
www.slcfug.org

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

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

Jser Groups

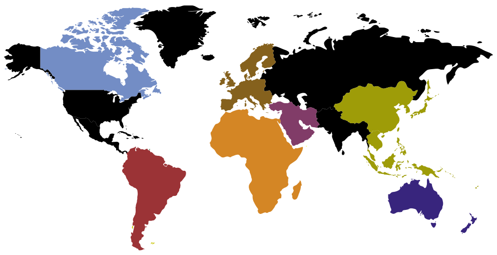
http://www.macromedia.com/cfusion/usergroups



About CFUGs

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

INTERNATIONAL



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

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

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

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

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Germany
Central Europe CFUG
www.cfug.de

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

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

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

Scotland
Scottish CFUG
www.scottishcfug.com

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

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

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

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

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

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

Poor Man's HTTP Compression with ColdFusion

Speed up your Website and lower bandwidth utilization

By Sami Hoda

There are a couple of things to note before we begin.

First, HTTP Compression is a great way to speed up your Website and lower bandwidth utilization all at the same time. In this example, your ColdFusion server (6.1 and above) will encode CFML output using GZIP and have the browser decompress this data on the fly. In a corporate environment, Webmasters may choose to go with third-party software such as Port80 Software's httpZip, which can compress other files as well, including .js, .css, and your HTML files. In this example, we are only compressing the generated HTML output from a .cfm template execution.

Doing a Google search will reveal many different Java servlet filters that can do the job for you for free. Many of them have issues with memory leaks, don't scale well, or have other issues related to them, so make sure you are picking one that has been well tested, and make sure to always test any server changes in a development environment before deploying to production. Another thing to note is that not all browsers support GZIP encoding, so make sure your user group is using Microsoft Internet Explorer 6+ and Firefox 1.0+.

As a side note, if you are using ColdFusion to consume data from another ColdFusion server that is using HTTP Compression, be sure to read Steven Erat's blog post on potential issues and workarounds at http://www.talking-tree.com/blog/index.cfm/2004/7/28/20040729. A couple of CFHTTPParam tags allow you to tell the server to send an uncompressed response.

In this example, we will be using Coldbeans Software's Compress Filter, which is available at http://www.servletsuite.com/servlets/gzipflt.htm.

- **Step 1**
  Shut down ColdFusion.

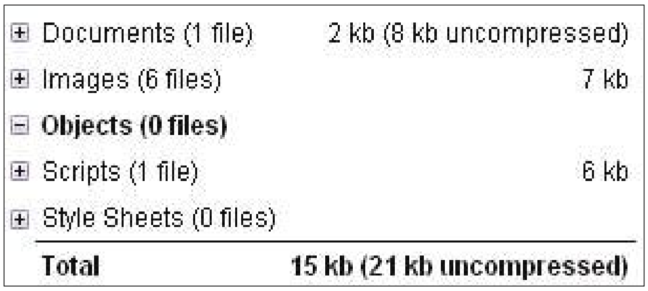
- **Step 2**
  In cfroot\wwwroot\WEB-INF\lib, download the gzipflt.jar file.
  In cfroot\wwwroot\WEB-INF, edit the web.xml file and place the following XML markup near the end of the file just before the closing </web-app> element:

```
<!-- start GZipFilter settings -->
<filter>
  <filter-name>GzipFilter</filter-name>
  <filter-class>com.cj.gzipflt.GzipFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>GzipFilter</filter-name>
  <url-pattern>*.cfm</url-pattern>
</filter-mapping>
```

- **Step 3**
  Restart ColdFusion.

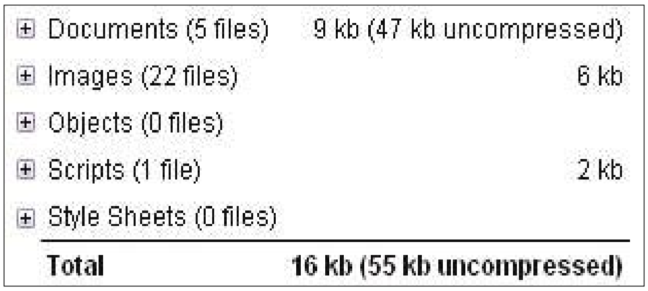
That's it!

The effects of the changes can be measured quite easily in Firefox using the Web Developers Extension version 1.1.3. This extension is available at http://chrispederick.com/work/webdeveloper/.



⊞ Documents (1 file)	2 kb (8 kb uncompressed)
⊞ Images (6 files)	7 kb
⊟ **Objects (0 files)**	
⊞ Scripts (1 file)	6 kb
⊞ Style Sheets (0 files)	
Total	**15 kb (21 kb uncompressed)**

Figure 1:



⊞ Documents (5 files)	9 kb (47 kb uncompressed)
⊞ Images (22 files)	6 kb
⊞ Objects (0 files)	
⊞ Scripts (1 file)	2 kb
⊞ Style Sheets (0 files)	
Total	**16 kb (55 kb uncompressed)**

Figure 2:

Load the ColdFusion Administrator Login page. On the Web Developer Extension, if you click Information -> View Response Headers, you should see something similar to this:

```
Server: Microsoft-IIS/6.0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Content-Length: 2261

200 OK
```

Note, Content-Encoding and Content-Length, both of which indicate that the GZIP filter is running successfully. Go back to the ColdFusion Administrator Login page and, on the Web Developer Extension, if you click Information -> View Document Size, you should see something similar to Figure 1.

In the "Documents" line, note that it compressed the ColdFusion Administrator Login page to one-fourth its original size, giving you a total savings of 6 kb. This may not seem like a lot, but try the same thing after logging in. You should see something similar to Figure 2.

This page shows a savings of almost 40 kb! Now that's huge. Bring up other Websites on this server and watch the savings add up!

I'd like to thank Steven Erat for reviewing this article prior to publication.

About the Author

Sami Hoda is an 11-year IT and Web development veteran with degrees in computer information systems and management and human resources. You'll find him reading profusely and passionate about best practices.

samihoda@gmail.com

ColdFusion: So Easy, Even a Caveman Can Do It

– CONTINUED FROM PAGE 5

While many people won't argue with this logic, there's a common rebuttal – "at what cost?" Yes, there is a cost to getting these features out of the box. Part of that cost is obviously the actual price of ColdFusion. Of course, you save so much time and money doing your programming that the server pays for itself in this respect. Another part of the cost is performance. Given all that it does, the ColdFusion Application Server does what it does remarkably fast, but it is an extra layer of abstraction between the code and the J2EE Application Server that the code is running on. To meet the needs of an overwhelming number of users, the performance impact that this layer of abstraction comes with is not only acceptable but well worth it. The last cost of ColdFusion being easy to use is the effect it has on the quality of developers and code associated with it. Unfortunately, being easy does mean that ColdFusion, more than other programming languages, attracts more developers who have little or no programming experience or education. This is actually not such an issue – in fact the majority of the best developers I've worked with come from very non-technical backgrounds. On the other hand, couple this with the fact that the programming language and the CF server are very forgiving and you do have a recipe for creating a lot of bad code. Bad code usually means poor performance and/or difficult maintenance, and any company/manager who experiences this is likely to make a general assumption about ColdFusion in general. This is the fault of the developer, not of ColdFusion, but unfortunately the saying that one bad apple spoils the bunch is especially true in business, where discovering one bad apple can cost a lot of money.

To take the questions originally posed in this editorial and rephrase them as statements:

- There is definitely no such thing as a programming language that is too easy to learn or use, but I strongly recommend avoiding any language that makes tasks more difficult than need be.
- A programming language that is easy to learn and easy to quickly achieve results with is definitely the most viable business solution. Anyone who says otherwise won't be in business for very long.
- Plenty of ColdFusion developers are not what some people might call "real programmers." This is not necessarily a reflection on the platform; I've met plenty of Java and .NET developers who couldn't code their way out of a wet paper bag either. The very idea that someone is "not a real developer" is ridiculous. You're either a developer or you're not. If you get the job done and you do it fast and well, I don't care what you wrote it in – even if you are a caveman.

ColdFusion Hosting is our Complete Focus

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

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

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

CFDynamics

866.233.9626 > CFDYNAMICS.COM

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

macromedia | ALLIANCE PARTNER

BlueDragon Certified V.I.P. Hosting Partner

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.

HostMySite.com

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