

COLD FUSION Developer's Journal

ColdFusionJournal.com

April 2005 Volume:7 Issue:4

Database Design 8

MVC for You and Me 18

**Application Framework
and Development
Methodology** 22

LAMDA Boxes 24

**Creating
Configuration Files** 28

**ColdFusion in
Education** 34

**ColdFusion +
Model-View-Controller
= CFMVC** 42

**Challenging Yourself
with Design Patterns** 48

ColdFusion Gateways

What are these things and what can I do with them – including SMS phone applications, IM bots, and Oracle triggers PG 14

PLUS...

Challenge # 2: Taking a Poll	7
Two of My Favorite Things: Software Architecture + ColdFusion	32
ColdFusion User Groups	47



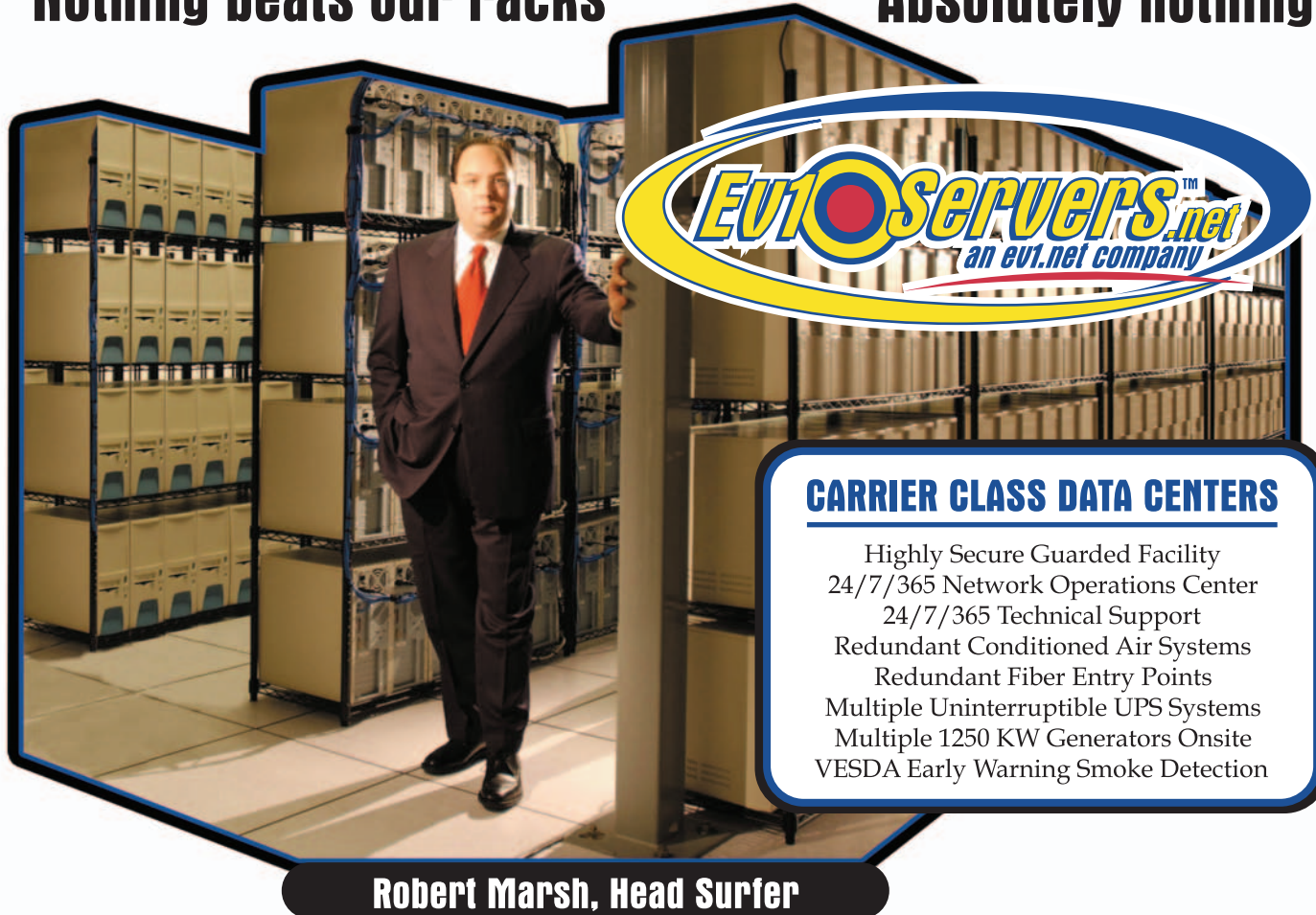
We figured it was about time that web video stopped looking like web video.

This is a scene from one of the world's best websites, with video playing next to traditional web animation. The look of full-screen video, without ugly web players or pop-ups.

To tour the site—and see the future of video—visit:
macromedia.com/go/video5

Nothing beats our racks

Absolutely nothing



ev1servers.net
an ev1.net company

CARRIER CLASS DATA CENTERS

- Highly Secure Guarded Facility
- 24/7/365 Network Operations Center
- 24/7/365 Technical Support
- Redundant Conditioned Air Systems
- Redundant Fiber Entry Points
- Multiple Uninterruptible UPS Systems
- Multiple 1250 KW Generators Onsite
- VESDA Early Warning Smoke Detection

Robert Marsh, Head Surfer

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from **\$299*** **Dedicated Server**

Dual Xeon 2.4 GHz
2 GB RAM • 2 x 73 GB SCSI HD
Remote Console • Remote Reboot
2000 GB Monthly Transfer Included

Instant Activation!

23,000+ Servers and Growing!

1-800-504-SURF | ev1servers.net

PLESK7
RELOADED
Preferred Control Panel

IP Compliant. Price subject to change. Quantities Limited.
*Per month. Set-Up fees apply. See web site for complete details.

ARE YOU READY FOR

MX7

CFDdynamics
1.866.233.9626

- 24/7 network monitoring
- Reseller program
- Comprehensive on line support
- State of the art data center

editorial advisory board

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

editorial

editor-in-chief

Simon Horwith simon@sys-con.com

technical editor

Raymond Camden raymond@sys-con.com

editor

Nancy Valentine nancy@sys-con.com

associate editor

Seta Papazon seta@sys-con.com

research editor

Bahadir Karuv, PhD bahadir@sys-con.com

production

production consultant

Jim Morgan jim@sys-con.com

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
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com
Andrea Boden andrea@sys-con.com

contributors to this issue

Simon Horwith, Jeff Houser, Mike Nimer, Joe Rinehart,
Neil Ross, Jeff Peters, Hal Helms, Andrew Simon,
Steven Forehand, Phil Hulsey, Sam Farmer

editorial offices

SYS-CON MEDIA

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

postmaster: send address changes to:

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

©copyright

Copyright © 2005 by SYS-CON Publications, Inc.
All rights reserved. No part of this publication may
be reproduced or transmitted in any form or by any means,
electronic or mechanical, including photocopy
or any information, storage and retrieval system,
without written permission.

Worldwide Newsstand Distribution

Curtis Circulation Company, New Milford, NJ

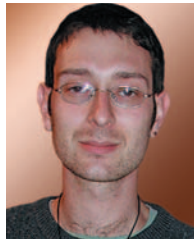
FOR LIST RENTAL INFORMATION:

Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

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

All brand and product names used on these pages
are trade names, service marks, or trademarks
of their respective companies.

Two of My Favorite Things: Software Architecture + ColdFusion



By Simon Horwith

The topic of focus for this month's issue is "architecture." Software architecture is the study and practice of the art of planning and developing applications, and it also happens to be my favorite topic and area of expertise. It is the cornerstone of every-

thing we do, and developers of every level of expertise and experience can gain from its understanding.

By the time this issue reaches our readers, the CFUnited conference will be fast approaching – and I encourage ColdFusion developers everywhere to try to attend. If software architecture and/or object-oriented programming are of interest to you, there will be many sessions covering various aspects of these areas of knowledge, in addition to many other great topics. Of course, CFUnited will also be the first opportunity that ColdFusion developers have to attend so many presentations devoted to the new features in ColdFusion MX 7. I look forward to seeing and meeting many of you there. You can read all about the CFUnited conference in this month's community column. In addition to our community focus article on the CFUnited conference, we've got plenty of excellent articles about architecture for you in this month's issue.

Joe Rinehart has written an article explaining the Model View Controller

(MVC) design pattern and offers insight about his approach to implementing it in ColdFusion applications. Of course, MVC is hands-down the most popular and common design pattern used by developers on the web, so in order to offer an alternative take on this subject we also have an article by Sam Farmer about an MVC-based framework he developed for use in a membership driven application that he built. Speaking of frameworks, Neil Ross developed a framework based on the front-controller design pattern called "TheHUB" and has written an excellent article explaining how he architected that framework. I have written an article explaining the resource pool design pattern – that article includes a URL to download a flexible generic implementation of resource pool that can be implemented in your applications.

In our "Macromedia Speaks Out" column this month is an interesting article by Mike Nimer about possible uses for the Event Gateway that was introduced in ColdFusion MX 7. If you have been wondering why there is so much hype surrounding this new feature, look no further than his article!

– continued on page 12



About the Author

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal. Simon is a Macromedia Certified Master Instructor and a member of Team Macromedia. He has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com simon@horwith.com

Brad Lawryk: "Before MX Kollection, I 'thought' I was doing a **pretty good job**, and now ...

Everyone tells me I'm doing a **brilliant** job!"

If you own Dreamweaver,

MX Kollection is a must!

List Database Information

- Create dynamic lists
- Order records within a list
- Insert, Edit, Delete field in list
- Automatic navigation bars
- Automatic list filter
- Sort column by clicking on the header
- Automatic row counter
- Delete multiple records directly from list
- Create master/detail lists

Manage Recordsets Visually

- Create, Edit recordsets visually
- Visually add tables to query
- Contextual menu to refresh fields
- Large query management Zoom In/Out
- JOINS between tables
- Define and apply complex conditions
- Automatically generate SQL conditions
- Add GROUP BY for aggregated fields
- Extract information from multiple tables
- Optimized SQL generation
- Smart SQL queries for list filters
- Automated database introspection

Rapid HTML Form Creation

- Generate Insert Record Form
- Generate Update Record Form
- Insert, Edit or Delete form field
- Redirect to page after form submit
- Table and CSS form generation

Form Validation and Error Handling

- Validate form fields
- Rich formats library
- Allow custom validation formats
- Error handling
- Preserve submitted values on error

Upload Files and Images

- File Upload
- Image Upload
- Resize and sharpen image on upload
- Show Dynamic Image
- Download Uploaded File
- Delete file from specified folder
- Show If File Exists on Server

Send E-mails

- Send e-mail after form submit

Online HTML Editor

- Edit HTML content visually
- Use your own CSS styles
- Upload and manage server images
- Format tables and align images

Enhanced HTML Form Controls

- Date Picker
- Dependent Drop-downs
- Editable Drop-down (Combo-box)
- Masked Textfield
- Numeric Textfield
- Dynamic Search
- n...1 Dependent field
- Smart Date

Security & User Authentication

- Login form generator
- Login with User Levels
- Remember me feature
- Encrypted password
- Restrict access to page
- Generate password with fixed length
- Update user password
- Send password by e-mail

DREAMWEAVER PRODUCTIVITY TOOL FOR DYNAMIC WEBSITE DEVELOPMENT

Download a trial version here - <http://interaktonline.com/go/MXKollection/>

Find out how easy is to use our tools - **save 30%** by using the coupon in the right.

Caution: InterAKT extensions are known to cause a high increase in productivity. For those of you who can't deal with the free time please continue to hand-code or refer to our competitors for help.



\$100 coupon - 199920002001

Interakt

macromedia
approved



macromedia
ALLIANCE PARTNER

<http://www.interaktonline.com/>

president & ceo

Fuat Kircaali fuat@sys-con.com

vp, business development

Grisha Davida grisha@sys-con.com

group publisher

Jeremy Geelan jeremy@sys-con.com

advertising

senior vp, sales & marketing

Carmen Gonzalez carmen@sys-con.com

vp, sales & marketing

Miles Silverman miles@sys-con.com

advertising director

Robyn Forma robyn@sys-con.com

advertising manager

Megan Mussa megan@sys-con.com

sales & marketing director

Dennis Leavey dennis@sys-con.com

associate sales managers

Kristin Kuhnle kristin@sys-con.com

Dorothy Gil dorothy@sys-con.com

Kim Hughes kim@sys-con.com

sys-con events

president, events

Grisha Davida grisha@sys-con.com

national sales manager

Jim Hanchrow jimh@sys-con.com

customer relations

circulation service coordinators

Edna Earle Russell edna@sys-con.com

Linda Lipton linda@sys-con.com

Monique Floyd monique@sys-con.com

manager, jdj store

Brunilda Staropoli bruni@sys-con.com

sys-con.com

vp, information systems

Robert Diamond robert@sys-con.com

web designers

Stephen Kil Murray stephen@sys-con.com

Matthew Pollotta matthew@sys-con.com

online editor

Martin Wezdecki martin@sys-con.com

accounting

financial analyst

Joan LaRose joan@sys-con.com

accounts payable

Betty White betty@sys-con.com

credits & accounts receivable

Stephen Michelin smichelin@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

Challenge #2: Taking a Poll

This month's coding contest is all about asking questions



By Simon Horwith

At the time of this writing, the

submission deadline

for last month's contest

("Developer's Challenge

#1: Create a Blog, Win ColdFusion MX 7")

is still pending, so the winning entry will be

announced next month. In the meantime,

another month means another contest. . .

This month, our contest is to develop a generic poll application. By "generic," I mean that it should be able to be integrated with other applications in order to add the capacity to poll site visitors either with a single question or with several questions (a survey). The application must include an administrative interface. In this administrative interface, I would expect that an administrator should be able to create new polls and surveys, define the questions that make up a survey or poll as well as the order in which they appear, flag questions required or optional, view reports about survey usage, and should permit the editing or deleting of surveys or survey responses. I'd think that a developer should be able to make the statistics regarding answers submitted to date publicly viewable as well as viewable in the admin area, if they choose to do so. I say that "I would expect" this functionality advisedly because these are only suggestions, not requirements.

Obviously data should ultimately be stored somewhere other than memory – you can choose to use the file system, Access, or SQL Server. If there's another database that you'd rather use, so long as it's a freely available database and/or you provide scripts to set up one or more other database platforms,

then that is fine as well. Like last month, the code will actually be tested on a developer edition of ColdFusion MX 7 on J2EE...but for this month's contest, using new features in ColdFusion MX 7 is not a requirement – as long as the code will run on CFMX 7. Obviously, if you plan to add reporting and/or graphing capabilities to your solution, ColdFusion MX 7 will most likely make your life easier and offer you more functionality for achieving this.

I should note that the contests run in this column are coding contests. Appearance is not going to be judged, so don't worry if things aren't pretty. That said, whenever applicable, entries should take into account best practices with regards to separation of presentation layer from business logic, use of CSS, and other usability best practices.

The prize for the winning entry this month is a free entry to the CFUnited conference being held just outside Washington DC this June, for you AND a colleague. The winner will have to tell me the name of the other person that's going to attend in advance. Thanks to TeraTech for donating this great prize! You can learn more about the CFUnited conference in this month's community column.



About the Author

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

simon@horwith.com

Database Design

The relational database is the center of most advanced applications



By Jeff Houser

You won't get very far while building ColdFusion applications without the need for a relational database. Since ColdFusion is so easy to use, many developers come from non-programming based back-

grounds, and proper database design remains a mystery.

Perhaps you've worked on other applications, such as a spreadsheet application in Microsoft Excel or used an Access database with a single table to store data. Both of these options serve their purpose, however they are not the best choice for web applications. In this architecture focus issue, I wanted to take some time to introduce you to relational database concepts.

Understanding the Terms

Before I start blabbing about database this and database that, I want to make sure you understand what I'm saying. So let's start by defining some common terms that you should be familiar with.

- **Database:** A database is any collection of related data. It could be as simple as a grocery list or as complex as a full Enterprise Resource Planning application. The database is made up of many elements, starting with...
- **Tables:** A table is similar to a single spreadsheet. A relational database is made of multiple tables. Tables are made up of columns and rows. A column represents a single piece of data, such as a name, zip code, or street address. A row represents a single set of all columns. No two rows within a table can be identical. You can relate tables together using ...
- **Keys:** There are two types of keys in a relational database. A primary key is a column, or group of columns, that can uniquely identify a single row in the table. A foreign key is

a column in one table that can be used to uniquely identify the row from another table. It is through the use of primary and foreign keys that you define relations between tables.

The act of designing a database is to take your data, and define the tables and relationships that you want to store it. This process is often called database normalization.

Database Normalization

Normal forms are a way to check that your database structure is correct. There are seven different levels of normal forms and each normal form exists to avoid a certain anomalies, that will not allow you to insert, or delete, data without also inserting or deleting a piece of unrelated data. You probably don't need to know specifics about the each level of normal forms. When developing a database, you just want to think about storing each piece of data only one time. You don't want duplicates. In your table structure, have you duplicated data somewhere? If so, you may want to re-work your structure so that you don't.

This will probably make more sense with an example. Suppose you were writing a program to catalog your CDs. You might start with a list of sample data, maybe something like Table 1. This is a table with four columns: Artist, Album Name, Genre, and AlbumID. The AlbumID is intended to be the primary key of this table. In a real application, you'll probably have a lot more data, such as a song list for each album, release dates, or the names of band members. We'll keep it simple for this example, though. If you examine the data, you'll notice some places where the data is duplicated. The artist name, Guster appears twice. The genres Grunge, and Alternative appear twice. If you were building an on-line record store, you would not want to store the genre or artist name a lot of times.

This table also exhibits insertion and deletion anomalies, which were discussed earlier. What happens if you want to delete the Bishop Allen album? You will also inadvertently delete the "Rock" genre. That is a deletion anomaly. What if you wanted to create a genre for blues? You wouldn't be able

Complete source code and asset management in Dreamweaver MX—now possible with Surround SCM.

Dreamweaver users know a beautiful Web-based product is only skin deep. Underneath, it's a tangle of hundreds or thousands of ever changing source files. Without a good development process and strong tools, bad things happen. Surround SCM can help.

Surround SCM lets you...

Track multiple versions of your source files and easily compare and merge source code changes.

Check out files for exclusive use or work in private workspaces when collaborating on a team.

Automatically notify team members of changes to source files—push changes through your organization.

View complete audit trails of which files changed, why, and by whom.

Associate source code changes with feature requests, defects or change requests (requires additional purchase of TestTrack Pro).

Remotely access your source code repository from Dreamweaver MX.

Surround SCM adds flexible source code and digital asset control, powerful version control, and secure remote file access to Dreamweaver MX. Whether you are a team of one or one hundred, Surround SCM makes it easier to manage your source files, letting you focus on creating beautiful Web-based products.

Features:

Complete source code and digital asset control with private workspaces, automatic merging, role-based security and more.

IDE integration with Dreamweaver MX, JBuilder, Visual Studio, and other leading Web development tools.

Fast and secure remote access to your source files—work from anywhere.

Advanced branching and email notifications put you in complete control of your process.

External application triggers let you integrate Surround SCM into your Web site and product development processes.

Support for comprehensive issue management with TestTrack Pro—link changes to change requests, bug reports, feature requests and more.

Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Want to learn more?



Download Seapine's just-released white paper, **Change Management and Dreamweaver**, to discover tips, tricks and best practices for achieving full software configuration functionality. Visit www.seapine.com/whitepaper.php and enter code WM0604 to receive your copy today.

www.seapine.com

1-888-683-6456



AlbumID	Artist	Album Name	Genre
1	Nirvana	Nevermind	Grunge
2	Mudhoney	Every Good Boy Deserves Fudge	Grunge
3	Guster	Lost and Gone Forever	Alternative
4	Guster	Goldfly	Alternative
5	Bishop Allen	Charm School	Rock

Table 1: Album Data

to do so without also entering an album. That is an insertion anomaly. You want to build your tables to avoid these sorts of issues. So, how do you do it?

Well, in this situation, you may first want to separate the Genre into its own table. You can see the genre table in Table 2. I added a GenreID column to the table. This is an integer column intended to be the primary key of the table. We can also split out the artist information into a separate table, as shown in Table 3. The table has two columns, a primary key named ArtistID and an artist column. With the artist and genre information moved into their own tables, what does your original album table look like? There isn't much left, just a primary key and an album name. However, you'll still want to preserve the relationships between the album, genre, and artist tables. How do you do this? You take the primary key of the genre and artist tables and put them in the album table. Our updated album table is shown in Table 4.

Why is this different? Well, it takes less disk space to store an integer than it does text. While you probably won't notice any problems in tables with just a few rows, the difference becomes much greater when you are dealing with larger amounts of data, such as a

GenreID	Genre
1	Grunge
2	Alternative
3	Rock

Table 2: Genre Table

ArtistID	Artist
1	Nirvana
2	Mudhoney
3	Guster
4	Bishop Allen

Table 3: Artist Table

song database with a thousand records, such as iTunes. Each little bit starts to add up.

Database Relationships

There are really three different types of relationships that come into play when putting your data into tables. The first type is called a one-to-one relationship. This means that for any single piece of data A, there will only be a single piece of data B. If A, were a username, then B might be a password. For every username, there must be only one password and for every password there must be only one username. In most one to one relationships the data is stored in the same table. If for some reason you are splitting the data between two tables, you can represent the relationship by moving the primary key from either table into the other table as a foreign key.

The next type of relationship is a one-to-many relationship. This means that for every piece of data A, there will be multiple pieces of data B. For every piece of data B, there will only be one piece of data A. A good example of this is the artist/album, which we described above. For every artist, there may be multiple albums. But, each album only has a single artist. (For the sake of this example we are ignoring multi band compilations). You can represent one-to-many relationships by taking the primary key of the "one" side and put it into the table of the "many" side as a foreign key. This is what we did to split up the album and artist tables. Our example, above, created the genres and albums relationship as one-to-many.

The third type of relationship is a many-to-many relationship. This means that for every piece of A data there will be multiple pieces of B data, and for every piece of B data there will be multiple pieces of A data. Perhaps,

you've got the Aerosmith album. It's rock. It's blues. Where do you categorize it? The Genre relationship to an album could be a many-to-many relationship. To implement this type of relationship in the database, we create a special type of table, often called an intersection or linking table. This table does not usually contain any data, only the primary keys of the two tables that it is linking. To implement a many-to-many relationship between genres and albums, we would use the GenreID from the albums table and create a new table, as shown in Table 5.

I know it looks like a table of numbers, and that's what it is. The primary key of intersection tables is usually made up of all the columns in it.

Common Database Design Mistakes

Before wrapping up this column, I want to finish off by pointing out some mistakes I often see beginner developers making. All of these are obvious in hindsight, but you probably don't realize you did it "wrong" until you have a problem and the light bulb inside in your head brightens up, and "Uh-oh" escape from your lips.

- **Names:** You have a site registration, or an address book application, or something that requires you to collect and store the names of your users. Make sure that you store the first name and last name as separate columns in a database table, do not combine them. If the data is being collected, at some point you are going to be asked to do a mail merge. When you do that, it'll be a lot easier to do if you have the first and last names separate, so you can address people as "Mr Houser" instead of "Mr Jeff Houser."
- **Deleting Data:** It's easy to write a delete statement in SQL, but what happens when your client or boss calls up to ask where their data went? If the data is truly deleted, you have no way to restore it. If you're lucky you have backup tapes, from which you can restore yesterday's data, but that can get messy. A better way is to not ever allow users to directly delete data. Create a Boolean field in your database table called Deleted. If set to



I am not the intranet

There are workarounds for a mediocre intranet, but they still waste time and money.

Meet the Macromedia Web Publishing System. Unlike the average CMS, this works. By allowing business users to publish content, IT bottlenecks are eliminated. Users can simply point and click to update specific pages or sections—while other areas remain protected. So everyone in your organization can share critical information quickly and easily, and you have an intranet that actually lives up to its potential.

Learn how to make your intranet work for you, not the other way around:
macromedia.com/go/webupdate



AlbumID	ArtistID	Album Name	GenreID
1	1	Nevermind	1
2	2	Every Good Boy Deserves Fudge	1
3	3	Lost and Gone Forever	2
4	3	Goldfly	2
5	4	Charm School	3

Table 4: Album Table Update

AlbumID	GenreID
1	1
2	1
3	2
4	2
5	3

Table 5: Album Genre Intersection Table

1, the record is flagged for deletion. If set to zero, the record is fine. You can run batch scripts on a routine basis to delete the data, as needed. When the user calls up to find out where their data is located, you can just flip that flag to restore it for them.

- **Store the Date:** At some point, someone is going to want to look at the data you've been collecting through the web site. When are people registering on the site? When was the last time they modified their information? On most tables, I will add a "DateCre-


ated" field to store the time that the record in the database was created, and a "DateLastModified" field to store the time the data was last modified. On some projects, I've had a few "higher ups" quite shocked that this data was not being collected.

- **Define Relationships:** You should always make sure you use the built-in facilities of your database to specify keys, and define relationships between tables. In addition to helping you protect your data from inadvertent corruption, the database engine can often use these relationships and keys to automatically optimize queries. I've had quite a few sleepless nights trying to fix the invalid data that had resulted from deleting one piece of data, without deleting data that relates to it. For example, what if we deleted a genre, but did not delete the entries in our intersection table

that related to that genre? Something is going to break somewhere.

Those were just some of the more common mistakes I've made myself or seen others make.

What Next?

The relational database is the center of most advanced applications, web-based or otherwise. Some of the more common databases used in ColdFusion development are SQL Server, MySQL, and PostgreSQL. Oracle often shows up on sites with larger load, and Access will sometimes show up on smaller sites. I would recommend putting aside some time to learn about the database of your choice, because many of the skills you learn can easily be transferred to all database platforms. 

About the Author

Jeff 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


Two of My Favorite Things: Software Architecture + ColdFusion

—continued from page 5

Hal Helms has written a very interesting article for us this month, explaining the use of configuration files in application architecture, and Jeff Peters writes about his "LAMBDA Box" solution that uses Linux, Apache, MySQL, and BlueDragon to create very low-cost servers. We have our second developer contest this month and the winning prize is a free registration for the CFUnited conference for both the winner and a friend. Jeffry Houser has written an article about relational database design in his "CF-101" column this month, so if your database design skills aren't quite where you'd like them to be, Jeff's got your back.

Last but certainly not least, our case study column this month is an excellent article from

Steven Forehand and Phil Hulsey about the web portal at East Carolina University. The article traces the ColdFusion-based portal's seven-year evolution, including many of the archi-

tectural hurdles they've had to overcome along the way. I found it to be not only a great testimony to the power of ColdFusion, especially given the robustness of the application and the small size of the development team, but also a prime example to the kind of services and online experiences that a university can offer to their students and staff. I only wish I'd had something like that when I was a student in college. Well done, East Carolina! 

simon@horwith.com



[Home](#) [Help](#)[\[-\] App Server](#)[ColdFusion Admin](#)[JRun MC](#)[Restart](#)[System Logs](#)[Heartbeat](#)[Configure](#)[\[-\] Site Management](#)[Domains](#)[Emails](#)[Horde Web Mail](#)[File Manager](#)[Site Builder](#)[Web Counter](#)[AWStats](#)[Web Logs](#)[\[+\] Database](#)[\[+\] Apache](#)[\[+\] Applications](#)[\[+\] Advanced](#)[more...](#)**testmycfm DNS, Domain Host, and Email Options:**[Domain Host and DNS Configuration](#) [Email Configuration](#)**:: Domain Host and DNS Configuration ::**Please refer to this topic in the [Help](#) section before proceeding.**DOMAIN****Select Domain:** (1) **Dedicated Address:** [testmycfm.webappcabaret.net] **Server Address:** [pointer.webappcabaret.net]**Domain Name (mydomain.com) :** [Set Domain](#)**IP/Host Address:** **Apache Document Root :** **Apache Directory Index:** **AppServer Virtual Path(s) :** [Customize Apache Virtual Host](#)**DNS & SUB-DOMAINS****DNS Service :** [Set DNS](#)**MX Address (separate multiple MX with commas):**

Sub Domain	Apache Document Root	Directory Index	Virtual Path(s)	IP/Host Address	CNAME
	/usr/ngasi/contexts/testmy	index.html index.jsp index.		testmycfm.webapp	<input type="checkbox"/>
	/usr/ngasi/contexts/testmy	index.html index.jsp index.		testmycfm.webapp	<input type="checkbox"/>

[Email Configuration](#) ▲

Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with ColdFusion MX.

At WebAppCabaret our standards based process and tools make deploying ColdFusion MX applications as easy as a point-and-click.

We call it **Point-and-Deploy Hosting**.

Our advanced NGASI Web Hosting management Control was designed for the hosting and management of ColdFusion web sites and applications thus cutting down on maintenance time.

Backed by an experienced staff as well as a **Tier 1 Data center** and network. Our network is certified with frequent security audits by reputable security firms.

All ColdFusion hosting plans have separate and individual installation AND instances of ColdFusion MX 6.1 Enterprise with **full access to ColdFusion Administrator** and JRun Management Console; so there is virtually no restriction or customization required for your application.

Complete power and control at the tip of your fingers. We take care of the system and hosting infrastructure so you can concentrate on development and deployment of your application. That is **real ROI**.

Log on now at <http://www.webappcabaret.com/cdj.jsp> or call today at 1.866.256.7973



Brainstorming with Nimer:

ColdFusion Gateways

What are these things and what can I do with them — including SMS phone applications, IM bots, and Oracle triggers.



By Mike Nimer

With the announcement of ColdFusion MX 7 in February, we introduced event gateways that turn the ColdFusion web application server into a bona fide Internet application server. The gateways are a great new feature for developers. You can now build applications for a variety of protocols and devices, not just HTTP. Because it's ColdFusion, we made it easy to develop and deploy outside the browser, and you can let your imagination run wild.

But why, you may ask, didn't we have these to begin with? Ten years ago, when ColdFusion was first introduced, the Internet consisted of only a few network protocols including HTTP, ftp, and gopher. At the time, the idea of creating a server that focused specifically on one protocol, HTTP, made sense. Today there are dozens and dozens of protocols using the Internet: SMTP, XMPP, P2P, RMI, JMS, RTMP, just to name a few. ColdFusion MX 7 has now opened these protocols to the masses, making development for them as easy as development for http.

Let's do some brainstorming and think about some ways you can use these new gateways. To keep things simple, we'll just focus on the gateways we ship out of the box.

1. Asynchronous CFML Gateway

Logging

If you want to start keeping detailed logs about your site and your users' activities, the asynchronous CFML gateway is the way to go. You could do this in the past, but if you got too verbose and started logging too much, you could slow a page down with lots of file writes or database inserts.

Now you can create a logging CFC that will do the file writes and the database inserts. Instead of calling the CFC in your page and slowing your page down, you can send a quick message to the CFML gateway and continue processing your page with no performance hit. The CFML gateway will then take the time to wait behind other requests to call the logging CFC and to log the message

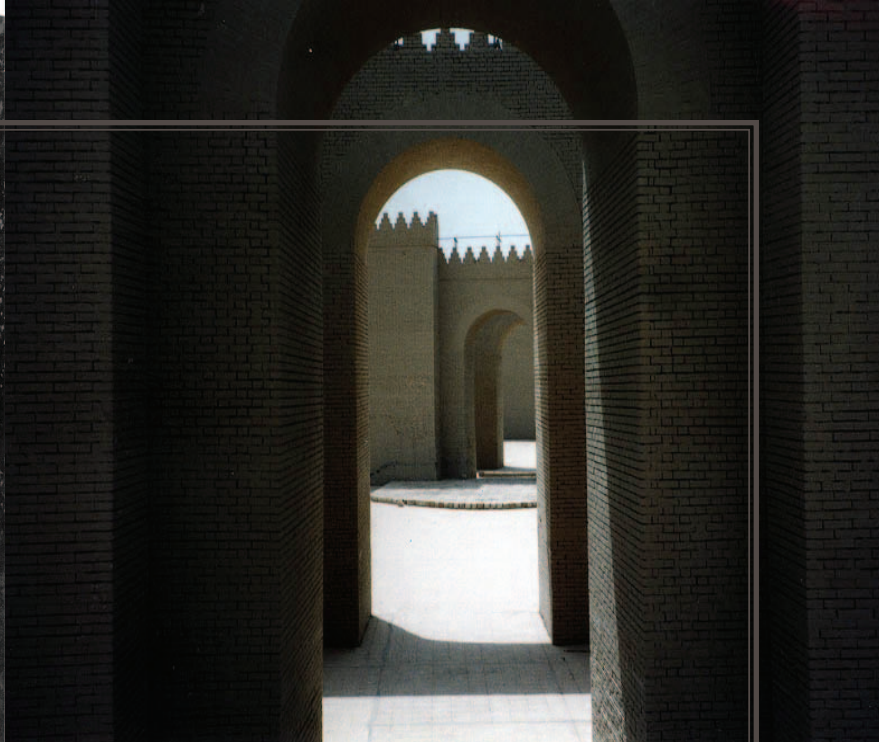
Caching

Let's say you have a page you run every night that loops over a list of URL's using cfhttp to call each URL and writes the results to an HTML file for you to serve up on your site later. Now, instead of this taking a few hours while CF calls the first URL, waits for a reply, writes the file, calls the second URL, and so on for each URL. You can quickly loop over each URL and send the URL to the CFML gateway, which will forward it on to your caching.cfc. The CFML gateway will then queue up all URL requests and run them in multiple threads simultaneously. What used to take hours might now only take a few minutes.

2. Directory Watcher

File Indexing

Imagine you have a site where users can upload documents that you will want available for searching on the site. For instance, this could be resumes for your HR department. Currently you would upload the file and then invoke the cfindex



tag to index the new file and update the collection. This works, but the user has to wait for the file to upload and for you to index the file for your site. They don't want to wait for that.

With the directory watcher you just upload the file and move on. The gateway will automatically pick up the new file and send it to your `index_file.cfc`, which will run the `cfindex` tag and update the collection.

Build Process Alerts

One application built with the directory watcher for the ColdFusion team is a file watcher for build alerts. Whenever a new check-in is made in the ColdFusion code base, we automatically rebuild the installers. When the file watcher notices that a new installer file exists, it will send out an e-mail to the engineering and QA teams alerting them to the fact that a new build is available and ready to test.

3. Socket Gateway

Server Administration

Have you ever wanted to just Telnet into a remote ColdFusion server and turn debugging on or off, add a custom tag path, run a scheduled task, add or build a verity collection, or reset log files? With the socket gateway you can build a Telnet interface into ColdFusion allowing a CFC to respond to the telnet session. Any `cfml` code you write can now be accessed through a Telnet or command window instead of a web page.

4. JMS Gateway

Oracle Triggers

In the past ColdFusion interaction with the database has always been a pull relationship. ColdFusion has to ask the database for data, ask for updates, and ask for changes. This is fine if the same application that updates the database is the same one that checks for changes. However, what if you have multiple apps that update the database? How can one application know if the data has changed without causing a lot of unnecessary database traffic?

Oracle ships with an embedded JVM, which means that you can invoke Java calls with a database trigger. All you need to do is connect Oracle to a JMS provider (one ships with JRun) and write some Oracle triggers that send JMS Messages whenever a database value changes.

Now you can also use the JMS gateway in ColdFusion to listen for new messages. When the data changes, ColdFusion can act accordingly. These actions can include: send an e-mail, send an SMS, update an IM status with new totals from the database, or update an application variable. The best part with JMS is you can have multiple applications all registered as listeners for the same Oracle trigger messages.

ERP/CRM Systems

A number of different ERP/CRM companies use JMS providers under the covers to send and receive messages between the different portions of the application. With the ColdFusion JMS gateway, you add ColdFusion to the suite of tools that are used with the ERP/CRM system.

5. XMPP/SAMETIME IM Gateways

Help Desk

This is where it gets fun as we can begin to build IM bots. You can create a "help desk" buddy on Jabber and, with the XMPP/SAMETIME IM gateway, register each of your IT and help desk technicians as approved buddies to the "help desk" buddy. Then you register the "help desk" buddy with every employee in your company.

When a user has a question for the help desk, they can just ask their "help desk" buddy and CF (the actual "help desk" buddy) will check its list of buddies (the help desk staff) and forward the request to the next online help desk technician. At the same time, it can automatically create a help desk ticket used to track the request or problem that you are asking them about.

Web Site Chat with the Help Desk

Along the same lines as the help desk idea, you can use the XIFF ActionScript library that's available online to write a chat window on your website. This Flash chat window would be hard-wired to use the ColdFusion XMPP buddy, which can track the conversation and manage the routing of the help desk request to the right help desk technician.

Time Tracker

Members of your teams may need to track their hours.

Instead of using an online time sheet or some scratch paper on their desk, teams can use the “time tracker buddy” and just send it a start or end message with a note.

Xyz project: start: phone call with client
 (10 minutes later)
 Xyz project: task: Remember to ask the supplier about new pricing.
 (20 minutes later)
 Xyz project: end: phone call, talked about lots of new ideas.

Abc project: start: designing logo
 (5 hours later)
 Abc project: end:

ColdFusion will automatically receive these messages, parse out the project name, action, and message, and enter the data into your database. This allows you to use the online time tracker application that you’ve already built to run a few reports and generate the weekly reports and invoices.

Expense Approval

Say you have an expense approval system, in which a user submits an expense report and it needs to be approved by their manager. You could write a ColdFusion application that can detect if the manager is online or not. If the manager is online when the user submits an expense report, an IM can be sent to the manager asking him to approve the expense report. The manager responds with a quick IM message of approval, the system registers the approval, and then sends on the expense report to the finance department. If the manager is not online, the system will just send an e-mail instead.

Bug Alerts

With the IM gateway, you can start alerting yourself to a number of smaller alerts that you need to know about but which don’t need to be kept around for permanent storage in your inbox. Using the IM gateway, you can set it up so that every time a new bug is sent to your bug queue a quick IM message can be sent off. You’ll then know that you have a new bug. This provides you with immediate feedback while at the same time keeping the extra clutter out of your inbox.

Status Indicator

IM clients have a cool feature: the status indicator. With many applications there is only certain data that you care about, which might include the most recent value of stock prices, highest bid in an auction, and sales figures for the month. With a simple ColdFusion application you can have ColdFusion update the stock quote buddy status with the latest stock quote or the action buddy with the current highest bid of the auction you are monitoring. All of this without sending a new e-mail every time the value changes. This now allows you to check the value just by looking at your IM client, not when the server wants to update you.

6. SMS Gateway

Phone Directory:

The phone directory on the home page of your intranet is a

great thing if you are sitting at your desk with your computer running and a browser open. Of course the time you need to really call someone is when you’re not at your desk. It’s when you are driving to work, stuck at the airport, or at home with no Internet connection. With a simple SMS phone directory application, you can send a text message with the name of the person you want to look up and seconds later ColdFusion will send back to you the name, office, and phone number of the person you need to talk to. Some phones even detect phone numbers in a message and allow you to select and call the person directly from the text message screen.

Flight Status

Lots of applications can send a change of status e-mail or even a text message, but with ColdFusion on the back-end of the SMS message, you can now send an alert and ask for a response. For instance, if a plane is canceled, a text message could be sent with a notice and a request for what to do: cancel flight, call customer server, or re-book on flight A or flight B.

Expense Approval

Similar to the IM example above, instead of only sending approval requests when the manager is at their desk, it can send a SMS to the manager regardless of where they are whether it be at home, the airport, or in their car.

Meeting Attendance

How many times have you scheduled a meeting with someone, only to have them get pulled away right before or miss their meeting reminder and pull a no-show? With the SMS gateway you could automatically send a confirmation message to the other attendees reminding and asking them if they will still make it to the meeting. “You have a meeting in 15 minutes with John, will you be able to make it or should we reschedule?” Depending on the answer, you may be able to skip a meeting!

Build Broken Alerts

We have a rule on the ColdFusion team: if you check-in new code and it breaks the build, you need to stay and fix it. As much as we try to prevent it, people will check-in, shut down their computers, and leave before waiting to see if they got a broken build notice sent to their email inbox.

Now you can set up your build machine to send a SMS text message to the person who broke the build instead of an e-mail. This way even if they are in the parking garage or driving home, they will get the notice and know they need to turn around and fix the build.

Server Monitor Alerts

There are lots of server monitoring tools out there. All of them can send you an e-mail if they detect a problem or even page you. But do you still carry a pager? What if the tool could send you a SMS text message asking you what to do: page someone, ignore, or reboot the server.

A problem has been detected with Server A. What would you like to do?

- 1 – Page Sys-Admin
- 2 – Ignore
- 3 – Reboot Server A

This silent text message allows you to fix the problem without interrupting dinner or the movie or game you are watching!

Website Authentication Via Phone


A common application already used by a few websites is User Authentication via the phone for website registration. A user registers for a new account with your website. Instead of sending an e-mail to confirm the registration was for the right person, you can have ColdFusion send the user an SMS text message asking them to reply to the message from their phone to confirm their identity.

Banking

SMS is an end-to-end secure protocol, authenticated against the SIM card in your phone. It is already used by banks around the world to notify users of bank balances changes or to allow them to perform online banking from their phone: transfer funds, check balances, pay bills, and so on. You can now expose secure financial data straight to a user's phone anywhere in the world.

On Call Alerts

Right now when companies need to find an on-call employee immediately – such as when a hospital needs to find a doctor – someone has to start making phone calls. With a simple on-call application, you can send a message to all of the on-call doctors with a short description of the problem and request for confirmation. As soon as the first doctor responds that they are on the way, the application can register the doctor with the emergency and send another alert to the other doctors canceling the request.

Some of the ideas I've mention above may seem complicated – SMS phone applications, IM bots, and Oracle triggers. But thanks to ColdFusion, the hard stuff is made easy, so many of these might only take a few minutes or some just an hour or two. I've only scratched the surface of what is possible. But hopefully I have your mind racing with new application ideas that are now made possible with the use of the ColdFusion MX 7 event gateway features! 

About the Author

Mike Nimer is a senior engineer on the ColdFusion engineering team, responsible for features such as the Rich Forms in ColdFusion 7 and the Administrator API. Before joining the engineering team, he spent three years working as a senior consultant with Allaire and then Macromedia consulting group, providing on-site assistance to customers with their architecture planning, code reviews, performance tuning, and general fire fighting.

mnimer@macromedia.com



> **Powerful**
Web Content Manager
Faster and
Easier

> **Call for DEMO**
1.866.870.6358

www.BeSavvy.com
Savvy Software Inc.



macromedia®
ALLIANCE PARTNER



MVC for You and Me

**Building a Model-View-Controller
application in ColdFusion isn't Hard**

I saw a one-man band once, and it was quite a sight. He had a bass drum on his back, operated by a cog attached to one ankle, thumping as he stomped his foot. Shoestrings tied to the neck of his guitar would move drumsticks attached to a snare drum perched atop the bass drum.

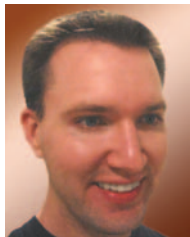
Another string attached to his right elbow operated a cymbal clamped to the side of the snare drum. I spotted duct tape in more than one place. It was a little complicated, and changing anything was bound to affect everything else.

Does this sound like an application you may have worked on? I'll admit it: it sure sounds familiar to me.

Now, I've also seen some very good orchestras. The conductor runs the show. She doesn't necessarily know how to play the viola or the sousaphone, but she knows how to use her musicians, and trusts them to do their jobs.

I don't know about you, but I'd much rather be a conductor. My musicians could change how their instruments worked, but as long as they still played in tune, it wouldn't matter to me. More importantly, I can add new instruments without having to learn to play them or figuring out how to tie, clamp, tape, and/or tack weld them into place.

That's a powerful paradigm, and this power is what the Model-View-Controller (MVC) design pattern gives you.



By Joe Rinehart

What Is Model-View-Controller?

According to James Dempsey's "Model-View-Controller Song," MVC is a way of organizing your code into "functional segments so your brain does not explode." MVC divides your application into three layers, each with a clearly defined task:

Model

The Model contains all of the business logic and data in your application. It is not a stretch to say that your Model literally is your application. Typically, a Model consists entirely of objects or services such as CFCs, Java classes, and / or Web Services. A Model never has any knowledge of Views or Controllers.

View

A View is what the user sees, and is how the user interacts with your application. The View's job is to allow user input and display the "state of the Model," an Object-Oriented (OO) way of saying "your application's data." Most often, Views are traditional ColdFusion pages (.cfm). Other mediums such as Flash can also act as views. Because your Model isn't aware of the Views, you can interchange views without affecting your overall application.

Controller

The Controller is simply "glue" code that passes data back and forth between the View and the Model. The Controller takes data from the user, such as form data, and gives it to the Model for processing. The results are then given back to the View, which is shown to the user.

While some users of MVC may object to passing the data back through the Controller, it's a common way to implement the MVC pattern in a Web environment. In a desktop application, however, the View might be allowed to ask the Model for its state information directly. However, because of the nature of the Web, it's a lot easier to simply "push" the state information into the View through the Controller.

Why Use Model-View-Controller?

I'm sure we've all heard talk about separating "business logic" from "presentation logic," and MVC does exactly that. By putting the Controller in between the Model (business logic) and View (presentation), we're insuring that they're not only separate, but highly reusable because they're not specific to a particular implementation. Additional Views can use the same Model without any modification. More importantly, the internal working of the Model can change without affecting the Views. This should sound familiar: we've arrived at the power described earlier when I said I'd rather be a conductor than a one-man band. Let's revisit what I said earlier, but change a few words:

"My components could change how their internals worked, but as long as they kept the same interface, it wouldn't matter to me. More importantly, I can add new components without having to learn how they work, just how to use them."

That's a very powerful statement to make about application design. Model-View-Controller utilizes Object-Oriented Design concepts known as "decoupling" and "separation of concerns" to achieve this power. In a "traditional" ColdFusion application, each page is responsible for a good deal: business logic, data access logic, presentation logic, and data validation. By isolating the business logic in the Model and the presentation logic in the View and having the two communicate through the Controller, we've "decoupled" the business layer from the presentation layer. Each of the three layers has a responsibility, or "concern" that is separate from the others. As long as the boundaries along which the layers communicate, or "interfaces," remain the same each layer can change its internal functionality without affecting other the other two.

Let's Code: An MVC Example

Building an MVC application in ColdFusion isn't very hard. You'll just need a good understanding of ColdFusion Components (CFCs) and an open mind. If you're not up and running with ColdFusion Components, you may want to read Jeffry Houser's article entitled "ColdFusion Components" when you're through with this article (*CFDJ* vol. 6, issue 11).

First, we need a problem to Model. For a fun example, I've chosen to Model translation of English phrases into Pig Latin. For our first attempt, I'm implementing a very simple version of Pig Latin – for any given word, move all consonants until the first vowel to the end of the word, and then add "ay." For example, "Model-View-Controller" becomes "odelMay iewVay ontroller-Cay."

It's easy enough to encapsulate all the logic for this into a CFC. Listing 1 shows the entire model for our application, contained in PigLatinizer.cfc. You'll see that it has a single method ("Translate") that receives a string, and returns a translated

string. At this point, we have a working "Model" of our "problem domain" – translating phrases into Pig Latin.

Now, we need a View to interact with our model. Listing 2 shows our basic translator form. It has a form field that collects a phrase to translate, a submit button, and will optionally show results of the latest translation.

Well, that was easy. We have a Model and View. Now we need a Controller to glue the two together. One approach many have taken is to simply add pseudo-Controller code to the top of their View pages, closely mimicking the standard self-posting form architecture ColdFusion developers have used for years. However, this places the Controller code in your View, eliminating any chance of re-use. Instead of doing this, we're going to use a CFC called PigLatinController. The source code for PigLatinController.cfc is shown in Listing 3. Examining the code reveals that it's pretty simple. There's an empty constructor ("Init") and a single method ("Translate"). While our constructor here remains empty, in the real world, this may be a place to insert information, such as datasource names or other configuration data, that the controller will need to pass along to the Model.

Examining the Translate method shows a controller in action. It first creates the needed portions of the model – in this case, simply an instance of PigLatinizer.cfc. It then passes data from the View (form.phrase) to the Translate method of the PigLatinizer. The user is then redirected to the view, with the resulting translation passed as a URL variable.

Now we have a Model, a View, and a Controller. What we're still lacking, however, is a way to have them all "talk." This is a where a framework is needed.

Building a Simple Framework

While there are two popular frameworks available for what we need to do (Fusebox and Mach-II), we're going to build our own mini-framework using about ten lines of code. It's going to be bare-bones, but it should get the job done while also introducing the need for more robust, application-agnostic frameworks.

Listing 4 shows our Pig Latin's Application.cfm file, where lines 3 – 13 represent our framework. First, we see if the application is initialized. If it's not, we create an instance of our Controller, placing it into the application scope. Second, we examine the URL scope to see if a parameter named "method" exists. If it does, we CFINvoke that method of our controller.

Running Our Application

We've now built a complete MVC application for translating phrases to Pig Latin. Let's step through two page requests, initial and form post, to trace how the pieces of MVC work together.

Initially, the user requests Index.cfm. ColdFusion processes the Application.cfm file. Our framework insures that an instance of PigLatinController (the Controller) exists in the application scope, and then looks for URL.Method. Because URL.Method doesn't exist, no action is taken. Index.cfm (the View) then presents the user with the appropriate form.

The user then enters a phrase and clicks the submit button, and our next request cycle begins.

Again, ColdFusion processes the Application.cfm file and

insures that our Controller exists. This time, however, URL.Method is defined, and its value is "Translate." Our mini-framework recognizes this, and invokes the "Translate" method of the Controller.

Inside the Controller, the "Translate" method creates an instance of PigLatinizer (our Model). It passes the data from the View (form.phrase) into the Model's "Translate" method, and collects the result. The Controller then CFLocates the user back to Index.cfm, appending the result of the translation to the URL. Index.cfm, the View, shows the "state of the model" (the translated phrase), and dutifully re-displays the translation form.

Modifying Our Application

Because we've placed the Controller between the Model and the View, changes to the Model won't affect the View, and vice versa. Realizing our implementation of Pig Latin isn't complete and that there are additional rules for handling words beginning with vowels, we need to modify our Model. Luckily, there's a freely available Java class that translates English into Pig Latin. We can modify our Model to use this class instead of its own internal logic (Listing 5 shows this modification). Because we're only changing its internals, and not altering its interface, the rest of our application will remain unaffected. By isolating our changes, and decoupling our business logic from our presentation logic, we've given ourselves a both more powerful and safer way to alter or scale our application.

The Need for Frameworks

Our bare-bones framework is fine for our Pig-Latin translator, but I wouldn't want to develop an enterprise application with it. On a low level, the method of processing input in the controller then CFLocating would make it hard to return meaningful validation results – the URL string could become huge! On a higher level, what if one method of the Controller needed to call another method? It'd be easy to add the code to do so, but this may begin to fall outside of the "concern" of the controller (passing data between the View and the Model) and may begin to fall into the realm of business logic or applica-


tion design. Frameworks such as Fusebox and Mach-II help out in this problem. By removing this control of "flow" from the Controller and letting you organize your application using XML, they both allow the Controller to assume its proper role of passing data. Frameworks have a number of other advantages, but that's another article.

Summary

Model-View-Controller is an elegant way to divide your application into logical layers. Each layer has a single job, and does it well. The Model represents your application's business logic. The View presents data from the Model, and accepts user input. The Controller serves to pass data between the View and the Model. Most importantly, the Controller serves to decouple the View from the Model, isolating what changes to bring unprecedented power to your application's design.

Further Reading

This article provided a high-level introduction to Model-view-controller. Hopefully, it's gotten you interested in using MVC to simplify and enhance your application designs. However, it's glossed over a good deal of Object-Oriented concepts and terminology. There's a lot to learn about the theory of MVC and the various design patterns that are used inside of it (MVC is technically a "compound" pattern, made up of other design patterns).

If you're ready to jump into using MVC for ColdFusion applications, I'd begin by taking a look at the Fusebox (<http://www.fusebox.org>), Model-Glue (<http://www.model-glue.com>) and Mach-II (<http://www.mach-ii.com>) frameworks." 

About the Author

Joe Rinehart is an associate with Booz Allen Hamilton, a global strategy and technology consulting firm. He's written the Model-Glue framework (<http://www.model-glue.com>) to help MVC development in ColdFusion, and runs his own blog at <http://clearsoftware.net>

joe.rinehart@gmail.com

Listing 1: Model - PigLatinizer.cfc

```
<cfcomponent name="PigLatinizer">
<!--- Constructor --->
<cfunction name="init" returntype="PigLatinizer">
<cfreturn this />
</cfunction>

<!--- Translates a phrase to Pig Latin --->
<cfunction name="Translate" returntype="string">
<cfargument name="phrase" type="string">
<!--- Declare local variables --->
<cfset var result = "" />
<cfset var word = "" />
<cfset firstVowel = 0 />

<cfloop list="#arguments.phrase#" index="word" delimiters=" " >
<cfset firstVowel = reFindNoCase("[aeiouy]", word) - 1/>
<cfif firstVowel gt 0>
<cfset word = right(word, len(word) - firstVowel) & left(word, first-
Vowel) />
```

```
</cfif>
<cfset result = result & word & "ay " />
</cfloop>

<cfreturn result />
</cfunction>
</cfcomponent>
```

Listing 2: View - Index.cfm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MVC Pig Latin Translator</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
Enter a phrase, and I'll change it to Pig Latin!
```



```
<cff structKeyExists(url, "result")>
<cfoutput>
<p><strong>Result: #url.result#</strong></p>
</cfoutput>
</cff>
```

```
<cfform name="phrase" action="index.cfm?method=translate">
Phrase: <input type="text" name="phrase" /><br>
<input type="submit" value="Go" />
</cfform>
```

```
</body>
</html>
```

Listing 3: Controller – PigLatinController.cfc

```
<cfcomponent displayName="PigLatinController">
<!--- Constructor --->
<cffunction name="init" returnType="PigLatinController">
<cfreturn this />
</cffunction>

<!--- Reverses a name --->
<cffunction name="Translate">
<!--- Declare local variables --->
<cfset var reversedName = "" />

<!--- Create the needed CFCs from our model --->
<cfset var translator = createObject("component", "PigLatinizer").
init() />

<!--- Use the Model to reverse the name --->
<cfset result = translator.Translate(phrase) />

<!--- Forward the user to the appropriate page --->
<cflocation url="index.cfm?result=#urlEncodedFormat(result)#"
addToken="no" />
</cffunction>
</cfcomponent>
```

Listing 4: Framework – Application.cfm

```
<cfapplication name="IgpayAtlinlayAnslatortray">

<!--- Our mini-MVC Framework --->
<!--- Make sure a controller exists --->
<cff not structKeyExists(application, "init")
or structKeyExists(url, "init">
<cfset application.PigLatinController = createObject("component",
"PigLatinController").init() />
</cff>

<!--- Url.method states that a method of the controller should be
called --->
<cff structKeyExists(url, "method">
<cfinvoke component="#application.PigLatinController#" method="#url.
method#" />
</cff>
```

Listing 5: Java-Enhanced Model

```
<cfcomponent name="PigLatinizer">
<!--- Constructor --->
<cffunction name="init" returnType="PigLatinizer">
<cfreturn this />
</cffunction>

<!--- Translates a phrase to Pig Latin --->
<cffunction name="Translate" returnType="string">
<cfargument name="phrase" type="string">
<cfset var translator = createObject("Java", "PigLatinizer").init() />
<cfreturn translator.translate(arguments.phrase) />
<cfreturn result />
```

```
</cffunction>
</cfcomponent>
```

Listing 6: Java PigLatinizer

```
import java.util.*;

public class PigLatinizer {
    public PigLatinizer(){
    }

    public String Translate (String phrase) {
        String[] words = phrase.split(" ");
        String result = "";
        for (int i=0;i<words.length;i++) {
            result += TranslateWord(words[i]);
        }

        return result;
    }

    private String TranslateWord(String word) {
        if (StartsWithVowel(word)) {
            return word + "way ";
        } else {
            int i = FindFirstVowel(word);
            if (i != -1) {
                word = word.substring(i) + word.
substring(0,i) + "ay ";
            } else {
                word += "ay ";
            }
        }
        return word;
    }

    private int FindFirstVowel(String word) {
        String vowels = "aeiouyAEIOU";
        // if you have 200 consonants in a row, something is
        wrong.
        int pos = 200;
        int i, j;
        for(i=0;i<vowels.length();i++) {
            j = word.indexOf(vowels.charAt(i));
            if ((j < pos) && (j != -1))
                pos = j;
        }
        if (pos == 200)
            return -1;
        else
            return pos;
    }

    private boolean StartsWithVowel (String word) {
        char let = word.charAt(0);

        switch (let) {
            case 'a':case 'e':case 'i':case 'o':case 'u':
            case 'A':case 'E':case 'I':case 'O':case 'U':
                return true;
            default:
                return false;
        }
    }
}
```

Download the Code...
Go to www.coldfusionjournal.com

TheHUB – Application Framework and Development Methodology

The most straightforward way is often the best



By Neil Ross

There are times in life when the most straightforward way to do something is the best way. This axiom holds true for ColdFusion application development as well. Over the past six years I have developed

websites and web applications almost exclusively in CFML.

Like most developers, I performed the same tasks over and over and developed a library of code that can be reused when the circumstance fits. I also realized that developers have their own styles and ways of looking at application development and at actual lines of code. Conformity to rigid development methodologies is often a difficult adjustment.

Because of these issues, I developed my own application framework and way of developing code that is sound, easy to learn and apply, and has a very small footprint in a ColdFusion application, which I call “TheHUB.” It grew out of my understanding of standard ColdFusion application development practices and the concepts that I found to be the most helpful from the Allaire Spectra product and studies of methodologies like Fusebox.

A Little Background

While working at Allaire, I did a presentation for some of my peers in Allaire Consulting Services regarding modular web development. I talked about breaking down visual layout into smaller modules so that you could generate user interfaces with standard and predictable layouts. Coming to ColdFusion from a design background this was a natural way to look at things for me. During this presentation and in the conversations that followed, I began to realize that I could similarly break up application code to provide the same predictability. The application framework provided by the Application and OnRequestEnd templates did not provide enough flexibility so I worked to extend that functionality with a structure that I found intuitive.

TheHUB framework maintains application layout and style and provides a method of organizing code. I have developed all

types of applications and web sites using this technique. I have shared the principles of its use with peers, coworkers and developers that I meet at conferences like MAX, CFUN and CFEurope. I figured it was about time I shared it with **CFDJ** readers who may have never made it to any of these conferences.

The Basics

TheHUB, like other application development frameworks, utilizes the notion of a central hub template that all requests for the application pass through. Object Oriented developers typically refer to this as a “controller” pattern of some sort (there are several controller patterns – the most common known as the “Front Controller” pattern). That central hub, I refer to it as the “framework hub”, is the point or place within the application that the processing of all code hinges upon. To some it may sound complicated but believe me, that’s far from reality. The code simply checks for a query string and then reads the parameters passed through it to handle template loading and screen rendering. In my examples, the “framework hub” will be the index.cfm but you could use some custom index page and start implementing the framework layout on an interior template.

The concept builds on a structure in which each request passes a unique set of keys that relate one-to-one with a site or application directory and template within the specified directory. In applications and web sites that utilize TheHUB, you’ll notice that the query string looks like “?dsp=the_hub.” This indicates to the “framework hub” template that it should include the “the_hub.cfm” template from the “dsp” directory. I’m not sure what could be easier.

The beauty of this approach is that you do not have to maintain a configuration file as your application expands or as you add another page to a process. Also, because the query string is so easy to read, your developers have a much easier time extending, maintaining and troubleshooting the application. Because TheHUB loads a small number of templates, it has a very small footprint but allows for extension by plugging in new modules very easily.

Of course, I’ve worked with frameworks and methodologies that start with tons of code and hundreds, if not thousands of templates; many of these templates, if not most, you’ll never look at or even use. One of the nice things about TheHUB is that no unnecessary templates exist, making it simple to follow. TheHUB merely guides the direction of the development of ColdFusion applica-

tions without shoving a lot of unneeded code down the developer's throat.

Here's how it works:

1. Client requests a URL within your application:
`http://localhost/thehub/index.cfm`
2. TheHUB analyzes the URL to read the URL Parameters. If none are found a default page is loaded into the framework; usually the "main.cfm" template from the "dsp" directory.
3. Client requests another URL: `http://localhost/thehub/index.cfm?dsp=page1`
4. TheHUB analyzes the URL to read the URL Parameters and loads the "page1.cfm" template from the "dsp" directory.

Overall, what this scheme allows you to do is create the look and feel of your basic site template using the "cmn/header.cfm" and "cmn/footer.cfm" files (they are included into the index.cfm which accepts all requests as part of the framework) and then read http requests that indicate the directory/template file to load into the body area:

Include the header.cfm

Include requested template

Include the footer.cfm

This is a basic modular design. You can get as fancy as you want in the header and footer, including whatever design elements you want or need.

In addition to the main three parts of the displayed page, a simple section of code handles the processing of the dynamically requested data. I call it the VarHandler and have separated it into an include file called "varhandler.cfm" in the "cmn" directory:

```
<cfif Len(cgi.query_string) NEQ "0">
  <cfset divpt=find("=", cgi.query_string)>
  <cfset stoppt=find("&", cgi.query_string)>
  <cfset request.dir=left(cgi.query_string,
divpt-1)&"/">
  <cfif stoppt NEQ 0>
    <cfset request.tmp=mid(cgi.query_
string,divpt+1,stoppt-divpt-1)>
```

```
<cfelse>
  <cfset chars_pos=REFind("[*],\|/
:<?{};&$%]",cgi.query_string)>
  <cfif chars_pos NEQ 0>
    <cfset request.tmp=mid(cgi.query_
string,divpt+1,chars_pos-divpt-1)>
  <cfelse>
    <cfset request.tmp=right(cgi.query_
string,Len(cgi.query_string)-divpt)>
  </cfif>
</cfif>
<cfelse>
  <cfset request.dir="dsp/">
  <cfset request.tmp="main">
</cfif>
```

The varhandler evaluates the CGI. QUERY_STRING variable and identifies the first key/value pair in the string. It then sets the REQUEST.DIR variable to match the key and sets the REQUEST.TMP variable to the value. Done properly, this indicates which template to load from which directory.

—continued on page 31

Develop Powerful Web Applications with ColdFusion MX 7

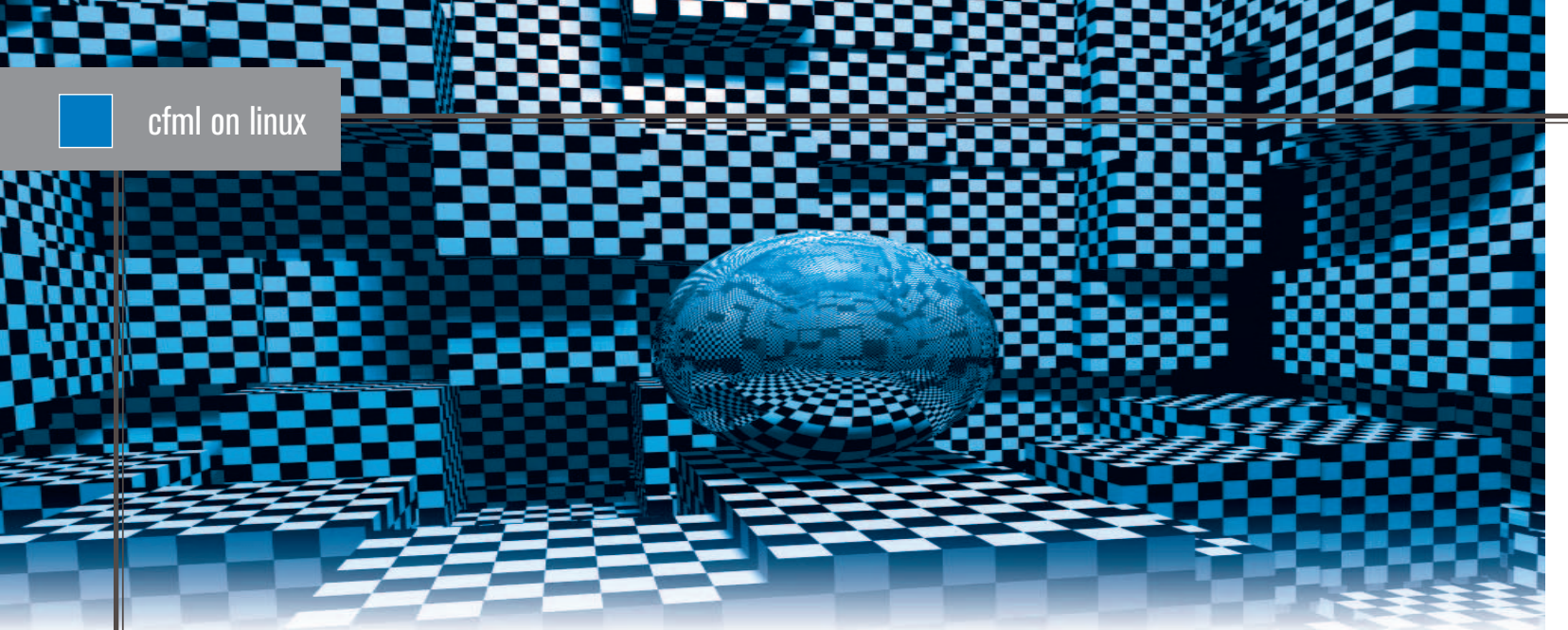
- ▶ Easily build and deploy multi-step data entry forms with CFML tags
- ▶ Solve business reporting problems with Integrated Reporting
- ▶ Manage your account with HostMySite's Control Panel
- ▶ Call 24x7x365 for expert ColdFusion support

Visit hostmysite.com/cfdj for a special offer



HostMySite.com

1-877-215-4678



LAMDA Boxes

Linux and BlueDragon for low-Cost CFML happiness

By Jeff Peters

In recent months, the term “LAMDA Box” has taken hold in the CFML community. The idea of a low-cost CFML server based on Linux, Apache, MySQL, and BlueDragon provided a solution for me in early 2003, and resulted in the invention of the term.

Let's take a look back at how that box came into existence, and what I've learned about Linux and CFML.

The Problem

In early 2003, I was looking for a way to provide a workgroup server to run internal program management applications for my team of CFML developers. The project in question provides development services for a nationally scoped application written for a federal government agency. Details of the application and agency don't matter, other than the fact that the project works essentially as a mini-IT department, taking requests for application changes and incorporating them into an ongoing development pipeline.

As the program manager, I had created a set of small applications to assist me in managing the project, in particular the pipeline of requests coming from our customer. These applications were written in CFML, and required a lightweight database to support them (at least at the outset). Wanting to prove the concept of these programs, I set out to find a low-cost solution for a reasonably high-reliability server on which to house them. As

with most proofs-of-concept, our budget for new hardware and software was very limited.

Going for low cost, I knew that Windows and ColdFusion were probably out of the question. Not only would I need to purchase a new computer to be the server, but I would need to buy the application server software. Windows and ColdFusion were out of the price range for my little concept project.

Naturally, I turned next to Linux as the operating system. Linux's reputation for high reliability was very promising, and the very low cost of a Linux installation was attractive from a price perspective as well. But Linux wasn't an instant winner, either.

Since I would not be only person needing to deal with the server in question, but was one of the only Linux aficionados at the company, I needed a distribution that provided a comfortable front-end experience. Unlike many Linux servers that run only from the command line, my server would need to have some GUI capability, and a certain degree of user-friendliness. Linux distributions like Mandrake, SUSE LINUX, and Linspire came to the forefront as having strong desktop presentations as well as the underlying power of Linux. I would have to decide which would do the job.

The final hurdle in keeping costs down was finding a low-cost computer to run the whole configuration. If there had been a spare computer available, we would have used it, regardless of processor speed. This was truly a lowball proof. The idea was to build a more expensive machine if the proof box worked.

To summarize, the problems I needed to solve were:

- Need for a low-cost, high reliability operating system
- Unfamiliarity of users with OSes other than Windows

- No budget for expensive application server software
- No budget for expensive database software
- Limited budget for hardware

The Solutions

At the time, the Lindows (now Linspire) distribution of Linux had been on the market for a couple of years, and had been progressing nicely as a desktop distribution. Having looked at Mandrake, SUSE LINUX, and Lindows, I wasn't drawn strongly to one over the others. Each used the KDE desktop environment, and seemed to have fairly reasonable installation tools. Since I could download distributions, I figured I might try each distribution and then decide which one to keep. Whichever distribution I chose, I knew I would use Apache as the web server software. So, with the Linux decision back-burnered, I moved on to the CFML server.

Fortunately, New Atlanta was in beta testing for the 6.1 release of their BlueDragon CFML application server. The free version of their software (known as BlueDragon Server) allows production instances in a workgroup environment at no cost. The compatibility notes for Linux stated that BlueDragon would work with a Debian distribution of Linux, so I didn't expect too much trouble in getting it to work.

Similarly, I chose MySQL as the database software for the project. MySQL is a solid, low-cost solution for reasonably sophisticated database problems. In fact, for my needs, it was a bit of overkill, but I was safe in the knowledge that it could grow with the applications if necessary.

Finally there was the hardware issue. Even in the age of personal computer price dives, I wasn't looking forward to trying to find a decent box on our extremely limited budget. Linux gave us a bit of leeway, in that we didn't need the same kind of processor horsepower or memory that we would have needed for decent performance out of a Windows-based platform.

I am usually very leery of introductory promotions from any company, as I don't care much for being a guinea pig. But with little to lose and much to gain, I chose to risk purchase of an extreme-low-end "white box" from a company called KooBox. The reasons were several. In conjunction with Lindows.com, KooBox offered an AMD-based box with 256MB of RAM, a 40GB hard drive, Lindows preinstalled, and a 17-inch flat panel monitor for \$500.00. With that small amount of money invested, I figured the damage wouldn't be too bad if the box turned out to be a lemon. After all, it was simply a "proof of concept" demo. If we proved we'd need to spend more money, that would at least be a valid result.

When the machine arrived, I set it up and configured it to run on our corporate intranet. This took about 15 minutes from the time I opened the boxes to the time I got a successful network connection. The excitement of a new experiment began to increase.

The next step was to download the Apache web server. Linspire includes a software management utility called Click-N-Run (CNR) which installs applications with a single click. Unfortunately, Apache was not available through CNR at the time. So I downloaded Apache 1.3 and installed it "the old fashioned way" (following the README file) in its default location (/usr/local/apache). With Apache started, I tested to see that I could get to HTML pages served on the machine.

Satisfied that everything worked so far, I downloaded BlueDragon 6.1 beta 1. I ran the installation to install to /usr/local/NewAtlanta, and checked the httpd.conf file to see

that the BlueDragon configuration lines were there. A restart of BlueDragon (/usr/local/NewAtlanta/BlueDragon_61/bin/StartBlueDragon) and a restart of Apache (/usr/local/apache/bin/apachectl stop; /usr/local/apache/bin/apachectl start) later, I requested the BlueDragon test page (<http://127.0.0.1/BlueDragon>) and got the expected response, shown in Figure 1.

I then downloaded MySQL and MySQL Control Center and installed them, this time using the CNR client. After a few hiccups with user configuration (before I read the installation manual and actually paid attention), I confirmed a successful installation by using the MySQL Control Center application. I then set up the database tables I needed for my application.

So far, so good. I went to the BlueDragon Administrator page and set up a MySQL datasource, referencing my new database. BlueDragon threw an ugly error suggesting I upgrade my client software. I recalled Charlie Arehart mentioning something about MySQL drivers on the BlueDragon interest list. I went to the list and found the message in question. Due to a licensing issue, NewAtlanta couldn't distribute the latest MySQL driver with BlueDragon. I downloaded the driver from MySQL and installed it, then went back to BlueDragon Administrator to set up the datasource again. This time it worked as expected.

With all the desired software installed and running, I realized I had created a CFML-based version of what the PHP folks call a LAMP box: Linux, Apache, MySQL, PHP. So I christened the concept LAMBDA. That is: Linux, Apache, MySQL, BlueDragon Application server.

The final step was to set up the various processes (Apache, MySQL, and BlueDragon) to autostart on bootup, and the configuration was complete. I was ready to implement my suite of management applications, and put the whole thing in front of my client.

Reliability

Reliability was one of the questions in the forefront on this test, particularly with using such a bargain-basement computer. I only wish every piece of cheap hardware worked out so well. Running the server proved to be quite easy, though after it had been up for several days we would get a "ServletExec not found" error when trying to access the pipeline management application. A restart of the machine solved the problem until the next randomly-timed occurrence. While a bit annoying, this really wasn't a show-stopper, as it usually manifested first thing in the morning so a restart wasn't a big deal. This wasn't

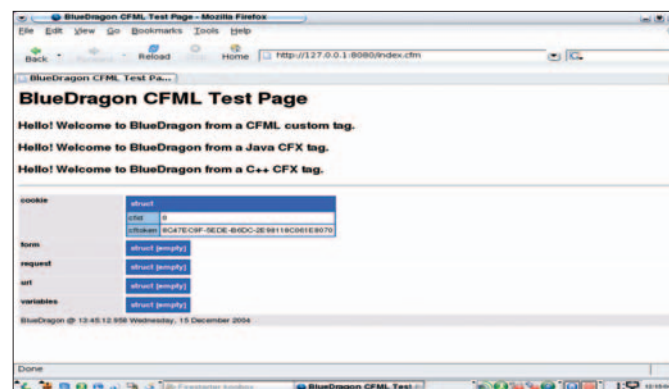


Figure 1: BlueDragon Default Page



a publicly available server, so uptime wasn't critical either. When Lindows released their 4.5 version, I reinstalled the server on the new OS with the by-then-production version of BlueDragon 6.1 and Apache 1.3.28 and the problem hasn't recurred.

As of this writing (December 2004), the pipeline server has been up for several months without complaint. It provides my development team and my customer with a resource to follow "who's doing what" with respect to development requests. Instead of being the test that would allow us to buy something more expensive, the proof-of-concept box has turned into a regular part of our daily existence, and continues to chug along, doing its assigned job. Figure 2 is a photo of the legendary beast itself.

Configuration Details

Though I can't provide all the details of configuration changes throughout the installation, testing, and upgrade of the original LAMBDA box, this section provides details of the current configuration, in case you want to build your own.

Apache 1.3.28

Apache is installed in `/usr/local/apache`. The `httpd.conf` file has the lines shown in Listing 1 to activate BlueDragon.

A startup item linked to the `apachectl` program ensures that Apache starts up if the system is restarted.

If you choose to use Linspire (as Lindows is now known, following legal action by Microsoft), you can now install Apache via CNR. If you go that route, the installation is a bit different, as the webroot is installed at `/var/www` and the configuration files are in `/etc/apache`.

MySQL 4.0.15

MySQL is installed in `/usr/local/mysql-standard-4.0.15-pc-linux-i686`. Utilities such as MySQL Control Center (`mysqlcc`) or Aqua Data Studio (www.aquafold.com) provide interfaces to the database. When configuring MySQL, pay careful attention to the User Configuration section of the User's Guide. MySQL has default users when the system is set up. You can define your own

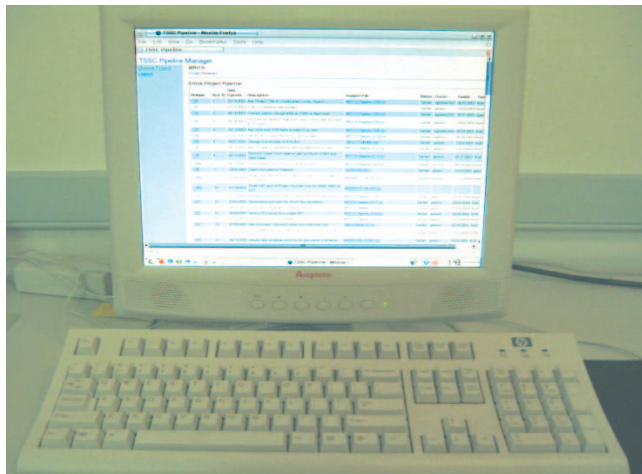


Figure 2: The Original LAMBDA Box

users after logging in as the default user "root". Note: If you use CNR to install MySQL, you'll get whichever version is most current on CNR, as opposed to the 4.0.15 described here.

BlueDragon Server 6.1

BlueDragon is installed in `/usr/local/NewAtlanta/BlueDragon_Server_61`. No modifications to BD were required following installation, beyond the usual datasource setup. A startup item linked to the `StartBlueDragon.sh` script ensures that BD starts up if the system is restarted.

I have done subsequent LAMBDA installations on Linspire boxes where the BD installer didn't notice that Apache was installed, and so didn't make it available as an installation option. If this happens, you can use the webserver configuration utility in the BlueDragon Administrator to configure Apache for BlueDragon.

Security

Followers of Linux may be aware that the Linspire distribution uses the root user as its default account. While this presents a security risk in the overall scope of Linux system administration, it's not a show-stopper in the context of a small workgroup server like the subject of this article. With implementation of a software firewall on the server (included in the Linspire distribution), we have seen no intrusion on the server throughout its service life so far.

That said, I strongly recommend use of tighter security measures for a server that is intended for the public Internet user community. Other proponents of the LAMBDA concept, notably David Epler (www.dcepler.net), have pursued using RedHat as the Linux distribution, and configuring all processes to run under non-root accounts. This takes a bit of effort to do correctly, but is well worth it if the server is going to be exposed to the public.

There, I have issued the requisite warnings.

LAMBDA Now

Since the creation of that first LAMBDA box, I've seen the interest in CFML on Linux grow at an increasing pace. The Fusebox 2004 conference included sessions specifically directed to the issues involved in running Fusebox applications under BlueDragon on a Linux box. David Epler's site, mentioned above, has some very good information on tips and tricks in these areas.


My personal interest in LAMBDA is as a low-cost developer's solution for people who are interested in the world of Linux, but haven't yet ventured forth. The maturity level of Linux as a desktop solution has reached the point where someone who has never worked in a UNIX-like environment can still use a Linux machine immediately and successfully. While you can still resort to the power of the console and command line interface if you wish, there is now a very deep wealth of GUI-based tools for Linux. These tools make it simple to do everything from typical file management to email and general office work.

Moreover, Linux offers the ability to resurrect hardware that had been relegated to the closet. I have done much of my recent LAMBDA experimentation on an old Pentium II laptop

that had become a doorstop as Windows upgrades left its 233MHz processor and 80MB of RAM gasping to keep up. With Linux installed, the same machine works quite well. Yes, there are some programs that take a few more seconds to load, but overall it's still a very serviceable machine. It's very nice to have a portable machine dedicated to tinkering with LAMBDA, without having to deal with the quirks of a dual-boot system.

Summary

The low-cost end of the workgroup application server market is often overlooked. There is a great deal of power to be leveraged within a CFML-based organization through the use of Linux, Apache, MySQL, BlueDragon application (LAMBDA) servers at the workgroup level. Expertise gained in writing in-house applications translates well when preparing to build larger scale applications intended for BlueDragon JX, BlueDragon J2EE, BlueDragon .NET, or CFMX platforms.

If you are interested in building your own LAMBDA playground, I have a class coming up called "Build Your Own LAMBDA Playground." During the two-day course, attendees will install and configure LAMBDA boxes, and get an intro to CFML applications using Fusebox. The best part is you get to keep the brand new laptop you set up during the course as part of your course materials! Details are at <http://www.protonarts.com>. 

About the Author

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

jeff@grokfusebox.com

Listing 1:

```
LoadModule servletexec_module libexec/mod_servletexec.so

<IfModule mod_dir.c>
  DirectoryIndex index.cfm index.html
</IfModule>

//At bottom of file:
ServletExecInstances default 127.0.0.1:9999
ServletExecAliases default /servlet servlet .jsp .cfc .cfm .cfml
<Location /servlet>
  SetHandler servlet-exec
</Location>
<Location servlet>
  SetHandler servlet-exec
</Location>
AddHandler servlet-exec jsp
AddHandler servlet-exec cfc
AddHandler servlet-exec cfm
AddHandler servlet-exec cfml
```

Download the Code...
Go to www.coldfusionjournal.com

What's your PDF?



Precise Document Formatting

With activePDF Server, you gain full control over your PDF output with conversion options that allow you to specify page size, compression and resolution options, embed text, create bookmarks, concatenate to existing files, and more. Licensed per server, you can easily add PDF generation to virtually any Windows application.



Populate Dynamic Forms

With activePDF Toolkit's form-filling capabilities, you can dynamically populate PDF forms with data and images from a database, allow users using only Adobe Reader to fill-in and save forms and use PDF forms as document templates to precisely control image placement and resizing. With Toolkit's robust API, the automation of virtually any PDF manipulation task becomes possible - append, stamp, stitch, merge, paint, secure PDF and more.



Promote Digital Fidelity

Do you need to standardize PDF output within your enterprise? With DocConverter, you can easily use built-in support for "watched" folders to implement server-side PDF generation in a matter of minutes, with full control over the PDF output at the server level. Or, use DocConverter's programmable COM object to integrate convert-to-PDF functionality within your enterprise application.



Present Data Fashionably

Ensuring precise layout of an HTML document can be a nightmare, especially when printing. PDF guarantees pixel-perfect layout every time as what you see is what you print. With activePDF WebGrabber, you can dynamically convert any URL, HTML stream, or HTML file to PDF on the fly, while maintaining embedded styles.



Download your
free trial version today at
www.activePDF.com

Copyright © 2004, activePDF, Inc. All Rights Reserved.
"ACTIVEPDF", "Leading the iPaper Revolution" and the
activePDF logo are registered trademarks of activePDF,
Inc. All activePDF product names are trademarks of
activePDF, Inc.





Creating Configuration Files

Make your work cleaner and easier to maintain



By Hal Helms

In their book *Head First Design Patterns*, the four coauthors lay out a series of key principles for creating robust software designs. One of the most important of these principles is “Find what varies and encapsulate it.”

In this article, let's apply this principle to the use of configuration files and explore the support for old-style .INI files and XML files.

“Find what varies and encapsulate it” is what most of us do for a living. The heart of any programming language is a variable – a name/value pair. Although we programmers are so used to a variable that it seems the most natural of things, people without a programming background don't find it natural at all.

What Exactly Is a Variable?

I learned this several years ago when my son was working on a presentation he would have to give to his fifth-grade class. At the time, he was working with a video game called “RPG Maker” (a devilishly clever introduction to computer programming that masquerades as a game) and he decided that he would explain the very basic underpinnings of computer programming languages.

“Hey, Dad – how would you explain a variable?”

*“A variable? Well, that's simple. A variable is something whose actual value may change over time. Take, for example, the notion of where you live – a notion we might call **homeAddress**. Now, the value of homeAddress may change (if you move from one house to another) but the idea of homeAddress is permanent.*

And that's a variable.”

Sometimes, I just have a way of making hard things seem simple. It's a gift.

Our son tried out his presentation on my wife. Susie isn't a programmer, but she works for IBM and so some computer basics must have worked its way into her, right?

She stared blankly at our son trying to explain how the idea

of your homeAddress is the same even though it may change – your home address, the value, not the idea (which stays the same). Or something.

I went to hear his presentation to his classmates and it was pretty clear that the idea of variables wasn't having any better success with them than it had with his mom.

And there's an epilogue to this story. Some time later, Steve Nelson (of Fusebox fame) and I were discussing some Fusebox-related matters and I happened to tell him the story of how my “homeAddress” example crashed and burned. As you would expect of a friend, Steve was properly sympathetic.

“Hal, you're an idiot,” he began. “Don't use abstract terms when trying to explain things to people. Give them something they can touch and see.”

“Like what?” I asked.

Steve was drinking a Coke at the time in a clear plastic bottle. “You see this bottle? That's the variable. See the Coke inside it? That's the value.”

He poured out the remaining liquid and dropped a pencil into the bottle. “Same variable, but different value.”

Hmmm...in his obviously simplistic way, which left out many important nuances, he did seem to have a point. Now, it was my job to crush Steve's argument underfoot so that I would emerge victorious!

“All right, smart guy,” I said. “How would you explain – an array?”

Steve stopped. It seemed that victory was mine. “An array?” he said. “Well...that's a six-pack!”

Variables in Applications

This month, I want to apply the “find what varies and encapsulate it” principle within a broader context – that of entire applications.

Almost every application of moderate size or more has certain pieces of information that the code relies on. Here are some examples:

- dsn
- log_file_path
- expiration_date
- banner_image
- verbose_messages
- required_score
- admin_email
- company_name

In each of these cases, the data is specific to a particular

installation, so that the only thing different about installation A from installation B is that the dsn or the company_name or the decision to turn verbose messages on or off (*etc.*) is different. We have, in other words, a variable.

But where should this information be stored? It might be stored in a database. Sometimes, though, this isn't a good option. It may be that there is no login that would identify a particular user. Or it may be that the information needs to be maintained by non-programmers and no user-friendly interface to a database exists. And if the needed information is so simple that a database isn't required – or so complex that modeling in a database is difficult – a better solution may be a configuration file.

Two types of configuration file have found widespread acceptance in the programming world. The first is a bit of “old school” – the INI file. First popularized by Microsoft, the ini file has a specific, simple format:

```
[SectionName]
keyname=value
;comment
keyname=value, value, value ;used when a keyname has multiple values
```

Section- and keynames cannot contain spaces and are case-insensitive.

So, a sample INI file might look like this:

```
[Configuration]
dsn=WegotWidgets
company_name =Widget Masters
verbose_messages = true
```

Notice the slight inconsistencies of treatment of spaces by the equal sign. Part of the INI specification is that parsers should be liberal in their application of rules.

How do we make use of INI files? We'll need to be able to both read and write to these files and ColdFusion makes it easy to do both of these. Let's first take the case of reading an INI file. We'll use the one shown above as our example.

To read one of the name/value pairs – say, **company_name** – we'll use ColdFusion's **GetProfileString** function.

```
<cfoutput>
  <h2>Welcome to #GetProfileString(ExpandPath('Sample.ini'), 'Configura-
tion', 'company_name')#</h2>
</cfoutput>
```

How do we get the value of “Widget Masters” into the INI file? Usually, these files are edited by hand, but ColdFusion allows you to edit them programmatically. In this example, I'll add a new name/value pair to **Sample.ini**.

```
<cfset SetProfileString(ExpandPath('Sample.ini'), 'Configuration', 're-
quired_score', 90) />
```

```
<cfoutput>
  <cfif variables.score LT GetProfileString(ExpandPath('Sample.ini'),
'Configuration', 'required_score')>
    Have you considered a career in the arts?
```

```
<cfelse>
  OK, now you can receive the secret Java decoder ring.
</cfif>
</cfoutput>
```

If you're wondering why you would need to set an INI file programmatically, consider that this file can be used to determine the next numbered item in a sequence – an order number or customer number for example.

To do this, you get, then immediately set, the number stored in the INI file.

```
<cfset invoice = GetProfileString(ExpandPath('Sample.ini'), 'Configura-
tion', 'last_invoice') />
<cfset SetProfileString(ExpandPath('Sample.ini'), 'Configuration',
'last_invoice', invoice + 1) />
```

Invoice No: #invoice#

Some configuration information is more complex and is a poor fit for the simple nature of the INI file. In such cases, XML can be the ideal solution. Here is a sample XML-based configuration file:

```
<configuration>
  <settings>
    <!-- database use = testing|production-->
    <database use="testing">
```

“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.



Once you're in it...



...reprint it!

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

Contact Kristin Kuhnle
201 802-3026
kristin@sys-con.com

REprints



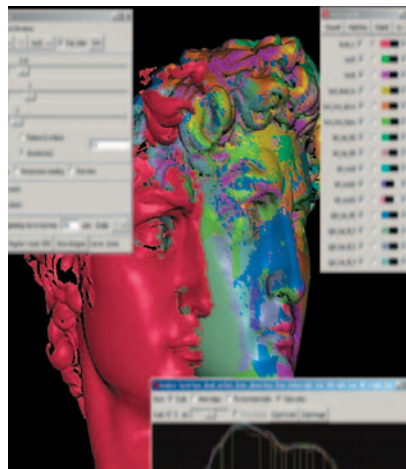
foundations

```
<testing>
<dsn
name="TestInventory" />
<db
type="msaccess" />
</testing>
<production>
<dsn
name="GlobalInventory" security="SSPI" />
<db
type="sqlserver" />
</production>
</database>
</settings>
</configuration>
```

The idea with this configuration file is that the application may be in one of two modes: testing or production. Each mode uses a separate datasource. When the "use" attribute of the "database" element is set to testing, we want to use the datasource defined in the "testing" element; in production mode, we want to use the datasource defined in the "production" element. Here is the code that reads and process the configuration file shown above:

```
<cffile action="READ" file="#ExpandPath('Sample.xml')#" variable="aFile" />
<cfset xml = XmlParse(aFile) />

<cfset databaseInfo = XmlSearch(xml, '//database') />
<cfset use = databaseInfo[1].xmlAttributes['use'] />
<cfset testingInfo = XmlSearch(xml, '//testing/dsn') />
<cfset productionInfo = XmlSearch(xml, '//production/dsn') />
```



```
<cfset testing = testingInfo[1].xmlAttributes['name'] />
<cfset production = productionInfo[1].xmlAttributes['name'] />

<cfset dsn = Evaluate(use) />

<cfquery datasource="#dsn#" name="myQ">
<!---sql here-->
</cfquery>
```

ColdFusion offers several XML tags and functions for dealing with XML. In this case, I've used only two: XMLParse and XMLSearch. The XMLParse function converts valid XML text into an XML object. Without this step, the XML text is unavailable for further processing.

The XMLSearch function provides an interface to the tremendous power of XPath, the XML-specific language for querying XML objects. For developers steeped in SQL queries, XPath may be initially hard to digest, but there are some excellent resources to help you.

Nate Weiss wrote a fine tutorial on working with ColdFusion's XML tags and functions, available at <http://www.macromedia.com/devnet/mx/coldfusion/articles/xmlxslt.pdf>. For in-depth examples of working with XPath, the tutorials at <http://www.zvon.org/xsl/XPathTutorial/General/examples.html> are invaluable.

Configuration files are very specific tools, but understanding how to use them can make your work cleaner and easier to maintain.

Further References Head First Design Patterns

By Elisabeth & Eric Freeman, Bert Bates, Kathy Sierra (O'Reilly)

About the Author

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

hal@halhelms.com

Extending

TheHUB provides developers with flexibility in the organization of code, as well. If you want to add a directory and load templates from it, your URL requests need only denote which template to load and from which directory. If you want to request the “showAllProducts.cfm” template from the “products” directory, your http request should go to “index.cfm?products=showAllProducts”. If you want to view a specific product, the URL would look something like this: “index.cfm?products=productDetails&product_id=123”.

The varhandlers.cfm template mentioned above looks for the “&” symbol as the separator of the initial key-value pair in the query string from the rest of the elements in the string. Templates that require other URL parameters will function properly by passing those parameters along the query string after the initial directory and template indicator.

You can also add child applications using an embedded TheHUB application in a sub-directory. You would copy the core files into your sub-directory and rename or delete the “Application.cfm” file from that sub-directory. Then, you would refer to that embedded application in the following way: http://localhost/thehub/child_application/index.cfm?dsp=main

Compatibility

The first thing that you will notice about the base code in TheHUB is that it’s very simple and straightforward. Because I often write applications for customers that they deploy on older ColdFusion Server installations, I needed to have something that would work across ColdFusion versions. I’ve deployed applications using TheHUB on ColdFusion Server 4.5, 5, MX and MX 6.1 in the past few months. In addition, all of my recent development and testing has been on ColdFusion MX 7. Because of this requirement to support older versions of CF, you will not find UDFs and CFCs being used as part of the core files for TheHUB but they can be developed and deployed, if they are appropriate for your environment.

At the Pennsylvania Office of Attorney General, my development team develops applications using the approach defined by TheHUB. We developed applications for ColdFusion MX 6.1 using UDFs and CFCs. All of the best practices pertaining to the development and implementation of these application elements apply and work well with TheHUB.

Summary

TheHUB provides a simple and predictable method of application development. Although it does not write code for you, nor does it alleviate the effort of

writing application code that you may have written in another application. What it does is provides a starting point and guidelines to follow for developing applications regardless of size or complexity. Furthermore, it works across many versions of ColdFusion and can be used to develop internal business applications or customer web sites. Remember, the most straightforward way is often the best.

For more information and to download the core files for TheHUB, visit my website: <http://www.codesweeper.co>



About the Author

Neil Ross is the system development manager for the Pennsylvania Office of Attorney General and owner of Codesweeper, an application and website development firm specializing in ColdFusion development and consulting. A Certified ColdFusion Developer and a ColdFusion Instructor during his days with Allaire Corporation, Neil is a frequent speaker at CFUG and MMUG meetings as well as conferences like MAX, CFUN and CF/MXEurope and an author of articles and co-author of Inside ColdFusion MX. For more information related to this article visit www.codesweeper.com.

neil@codesweeper.com

Here is example code:

```
Header.cfm
<cfinclude template="varhandler.cfm">
<html>
<head>
<title><cfoutput>#request.pagetitle#</cfoutput></title>
<link rel="stylesheet" type="text/css" href="cmn/style.css">
</head>
<body>
<table class="main-table">
<tr>
<td bgcolor="f2f2f2" align="right">
<h1 style="color:navy;"><cfoutput>#request.pagetitle#</cfoutput></h1>
</td>
</tr>
<tr>
<td bgcolor="f2f2f2">
<cfinclude template="../cmn/flatnav.cfm">
</td>
</tr>
<tr>
<td>
</td>
</tr>
</table>

Index.cfm
<cfinclude template="cmn/header.cfm">
<table width="100%" border="0" cellspacing="0" cellpadding="6">
<tr>
```

```
<td>
<cftry>
<cfinclude template="#request.dir#request.tmp#.cfm">
<cfcatch type="missinginclude">
<script language="javascript">
alert("The page that you requested has experienced an error. You'll
now be redirected back to the page you last viewed.");
history.back();
</script>
</cfcatch>
</cftry>
</td>
</tr>
</table>
<cfinclude template="cmn/footer.cfm">

Footer.cfm
</td>
</tr>
<tr>
<td class="copyright" bgcolor="f2f2f2" align="right">
<br>
Please report any problems with this application to neil.
</td>
</tr>
</table>
</body>
</html>
```

Download the Code...
Go to www.coldfusionjournal.com

Farewell, CFUN Welcome, CFUnited!

Spend up to five straight days learning and networking

By Andrew Simon

This year we bid farewell to CFUN, one of the longest running ColdFusion conferences in existence...or do we? No, not really – it's just received a facelift,

which includes a new name as well as many other compelling enhancements.

What was once CFUN is now the CFUNITED conference. CFUNITED is the realization of a vision that has been six years in the making. Started in 1999 at the National Institutes of Health (NIH) in Bethesda by TeraTech, this year is the seventh annual national ColdFusion event. Not only has the name changed, but also the anticipated attendance, session topics, extra-curricular events, venue, and nearly every other aspect of CFUN have been drastically improved. You can see some graphs at <http://www.cfunitied.com/about.cfm> for more about how the conference has grown.

This year's conference is a three-day event and includes over 40 nationally known speakers and over 1,000 attendees are expected to attend 64 sessions. This year will include Birds-of-Feather discussions in the evenings, panel discussions, a community area, and many other very good social and networking events. If three days isn't enough for you, TeraTech is going to offer a Team Macromedia and User Group Manager event on the day before the conference, and will offer full-day hands-on classes, instructed by CFUnited presenters, during the two days prior to the conference. This means that you can easily spend five straight days learning and networking! In addition, TeraTech has asked Adam Bell, the Team Macromedia member who organized "mini MAX" in New Orleans last year, to organize a "Mini MAX 2" event during the week as well.

CFUnited is being held from June 29 – July 1 just outside Washington, DC, at the Montgomery County Conference Center (MCCC). The MCCC is a beautiful new state-of-the-art facility that just opened in November 2004, and is right next to a Metro stop (White Flint metro) in Bethesda, Maryland – providing direct access from Reagan National Airport and to all downtown DC attractions. That makes CFUnited a great opportunity to also enjoy our nation's capital over the Fourth of July weekend, including the nation's best fireworks exhibition on the Capital Mall and the many restaurants and night clubs in DC.


So now you not only know when and where the conference is being held but also how the conference has evolved this year; but we still haven't really talked about the specifics of why developers should care. CFUnited is the only conference of its kind that is run by developers, for developers. What this means is that it isn't

dictated to speakers what they can and cannot talk about, and all of the speakers are encouraged to base their sessions on real-world experience and real use case studies so you can see what your colleagues at other organizations are doing. It also means that at CFUnited, the marketing fluff stops at the door so you can learn more from other developers.

There are too many different speakers talking about too wide a range of topics for me to list them all here, but in one paragraph I will try to give you a diverse *sampling* of the topics that make-up the seven tracks. Those of you new to CF will be happy to know that there's an entire track of beginner-level boot camp sessions covering everything from an introduction to the basics of CFML, to the basics of SQL, and several introductions to code reuse with UDFs and custom tags.

There is an accessibility and usability track with sessions on usability testing, search engine optimization, and CSS. There's a track on development and platforms that includes sessions on Verity, FarCry, Scaling and Tuning, and PLUM – to name a few. The CFML/.NET/Windows track features sessions on such topics as .NET, SQL Server 2005, developing Pocket PC applications, SQL Server optimization, and .NET/CFML administration. There's also an Advanced CF track, which includes sessions on Event Gateways, security, Java integration, and advanced database techniques. Similar to the advanced track is the empowered programming track that includes sessions on FLIP, MACH II, design patterns, project management, application branding, and domain models. Last but not least is the MX Integration track that features sessions on integrating ColdFusion with Flex, Flash, and Dreamweaver.

I mentioned that at CFUnited the marketing fluff stops at the door when you attend sessions. This should not be mistaken for an absence of sponsorship. In fact, far from it. There will be an exhibitor area where you can meet with sponsors in addition to other attendees and speakers. TeraTech, Microsoft, and New Atlanta are all gold-level sponsors of the event. CFDynamics, **ColdFusion Developer's Journal**, **MX Developer's Journal**, Fog Creek, Interakt, Savvy Software, Universal Mind, and PaperThin Software are all silver-level sponsors. AboutWeb and Montgomery College are bronze-level sponsors and there are several other sponsors below bronze level that have contributed products or other resources to help make the conference a success. There will be a community area in the exhibitor hall, sponsored and run by AboutWeb. There, attendees can relax, learn, play games, and win exciting prizes.

This year, if there's one ColdFusion conference not to miss, it is CFUnited. On a personal note, I've attended and presented at CFUN for the past several years and look forward to it more each year. Also, I live in Washington, DC, and my office is a 10-minute walk from the conference, so those of you who do attend should feel free to grab me if you're looking for advice or suggestions about where to go or what to do while you are in town. 

[Engage and Explore]

the technologies, solutions and applications that are driving today's **Web services** initiatives and strategies...

www.sys-con.com/edge2005

web services **EDGE**
conference & expo
FALL SERIES

**CALL
FOR
PAPERS
NOW
OPEN!**

Coming to a City Near You ▶

Web Services Edge Fall Conference Series

3 Dynamic Conference Programs Targeting Major Industry Markets

20+ seminars within 5 tracks will address the hottest topics & issues:

- ▶ Web Services: The Benefits and Challenges
- ▶ Web Services Security
- ▶ SOA (Service-Oriented Architecture) and ESB (Enterprise Service Bus) Strategies
- ▶ Interoperability, Incremental Integration, & Open Source
- ▶ The Management Process in Developing a Web Services Strategy

Why Attend:

- ▶ Improve the return on your technology investment
- ▶ Develop & sharpen your strategy and identify key action steps
- ▶ Find new ways to reach and impress customers with Web services
- ▶ Maximize the power of your enterprise
- ▶ Protect your business from security threats
- ▶ Assess Web services as a viable option

Program Features:

- ▶ Keynotes
- ▶ Tutorials
- ▶ Panel Discussions



Attention Exhibitors:

- ▶ An Exhibit-Forum will display leading Web services products, services, and solutions

Register Today! www.SYS-CON.com/Edge2005

Sponsored by

Web Services
JOURNAL

XML
JOURNAL

NET
JOURNAL

eclipse
developer's journal

WebSphere
JOURNAL

information
STORAGE + SECURITY
JOURNAL

wildj
JOURNAL

JDJ

Linux
WORLD

MX
developer's journal

asp.net
PRO

SDTimes

CoDe

Software Test
& Performance

*Call for Papers email: grisha@sys-con.com

For Exhibit and Sponsorship Information ▶ **Call 201 802-3066**



**The Westin
Washington, D.C.**
Washington, D.C.
September 7-8, 2005



**The Westin Santa Clara
Convention Center**
Santa Clara, CA
October 24-25, 2005



Hyatt O'Hare Airport
Chicago, IL
Oct 31-Nov 1, 2005

Produced by **SYS-CON**
EVENTS

© 2005 WEB SERVICES EDGE. ALL RIGHTS RESERVED



ColdFusion in Education

Developing a web-based portal for East Carolina University



By Steven Forehand



By Phil Hulsey

We have always thought that one of the best-kept secrets amongst the

ColdFusion “universe” is its tremendous potential and success within select educational institutions. Those that work with or for public educational institutions are very familiar with the staffing and budget issues that often cut resources, making it difficult to provide services to the communities that support them. However, as East Carolina

University found, all that is needed is a little

funding, some talent, and a willingness to learn.

As subscribers to *CFDJ* since the first issue, we have often read many “how-to”s with sections of code and invaluable “tips and tricks”. While we couldn’t be nearly as productive without these articles, we find it extremely beneficial at times to learn from other implementations and methods used to solve potential chal-

lenges. At risk of providing an effective sleep-aid, we would like to attempt to share the experiences of implementing ColdFusion at East Carolina University, an emerging national research university with enrollment of nearly 22,000 located in Greenville, North Carolina (<http://www.ecu.edu>).

Background

Hired by East Carolina University in the spring of 1997 by the Department of Human Resources as a Director of Information Processing, Steven worked with a tremendous group of supportive staff with our primary objective being the entry of a large volume of paperwork. His secondary duties included the development and implementation of several online applications including a web-based employment application.

At that time, he was armed with only a Bachelor’s degree in Computer Science and a solid knowledge of HTML and general programming. Development of a “web based” employment application seemed a somewhat intimidating task. He sat down, scratched his head, and began designing the employment application. Several chapters into a Perl manual that would make a dictionary seem like interesting reading, he figured there must be a better way to develop this application. As if by magic, a colleague called one afternoon and suggested trying a newly discovered product called (or so he thought at the time) “CoolFusion.” After borrowing the manuals, Steven started to dig deep, looking for that better way.

Just a few minutes of reading resulted in actually entering data in an HTML form and submitting, retrieving and displaying that data through a database connection. While this example is extremely elementary, it was nothing short of a miracle at the time. How could a few minutes of reading and programming

replace days of reading a Perl manual? Within two weeks, an application and associated database tables were designed and developed, allowing potential applicants search and apply for vacant University positions.

An Early Success

Within a few months, a system to allowing potential applicants to view and apply for vacant positions as well as online administrative tools allowing staff to easily track applicants was completed. Since then, a robust report generation module as well as EEO specific reporting have been added. Over time, the number of applicants applying for vacant positions within the University far exceeded those applying by traditional paper methods.

The use of this ColdFusion-based system offered cost savings across the board. Not only did fewer paper applications have to be printed and made available to the general public, but also less staff time was spent processing incoming paper applications as well as mailing paper applications to interested applicants. In fact, nearly 60% of the employment applications are received via this solution. The implementation of this type of solution also allowed for a more diverse applicant pool, as information was made available to a much larger audience and was not restricted to a local area.

Nearly seven years later, with nearly zero maintenance, the same application is running strong and has received nearly 150,000 employment applications from over 20,000 applicants applying for over 2,500 vacancies. In addition, the application has the distinction of having been selected as one of two ColdFusion applications developed by East Carolina University as national "best practices" by a major college and university personnel association, with the other being an employee performance evaluation system.

Next Steps

After nearly a year with the Human Resources department, Steven moved on to the campus Information Technology department as a software developer, with his primary responsibilities being the development of "web based" applications for students. It didn't take long after joining a tremendously talented team with excellent leadership and support to realize that the same "magic" experienced earlier with

ColdFusion could definitely be applied.

At the time, the two or three applications available to students via the web were HTML pages generated by a COBOL batch program. While effective in providing information on a daily basis, it did not provide students with the dynamic information they required.

It wasn't long before we were able to develop a system that allowed students to log in and obtain a great deal of personal and academic information via a web interface. This web interface was called "The Student Desktop," and it allowed students to perform many standard student tasks such as check their grades, register for courses, view course offerings, update address information, and so on. The system offered personalized content to the degree that it even "sang" Happy Birthday. Corny? Maybe. Effective? Definitely.

Over a lifetime of nearly two years, the ECU Student Desktop enjoyed usage by nearly 90% of the student community. Though the Student Desktop was functional and provided a robust set of applications for the campus community, it was lacking in several areas. Most importantly, while students could benefit from the information provided, few applications existed to help the academic and administrative staff. In addition, usage peaked during time periods such as final grades and course registration, though the system saw little usage otherwise. From a development perspective, the current system did not make use of a specific development methodology or allow for easy extensibility or customization.

Development Group Established

During the time we were considering how to best address the needs of the academic and administrative community, we were able to expand our number of staff and put together a small team consisting of a team manager and four software developers devoted entirely to development of these "web enabled" applications. The team, known as the New Technologies Development Group, received a modest amount of funding from the state of North Carolina for the purpose of providing a determined "baseline" of web-enabled services for the campus community.

While we knew that we wanted to re-engineer some of the applications made available via the Student Desktop, we needed a delivery method that would give

all members of the campus community a centralized location for access to the information and applications needed. We were also determined to design a system that provided greater security while also providing extensibility and a set of APIs to our development team. If we could possibly accomplish all of these tasks while also finding a method to offer maximum code reuse, we would have a winner. This seemed like a rather tall order at the time.

By 1999, the portal buzzword was gaining momentum. We began brainstorming on how to best implement needed capabilities for the university through other portal implementations. However, the best fit we could find was a higher education portal project with a timeline that far exceeded our requirements... not to mention the overly complicated code that would be required for creating new portal channels. While the overall concept was useful, we felt we needed more functionality than any of the current models provided. We wanted a portal that would give developers the capability to quickly write an informational channel while giving them the ability to develop and integrate more complex applications. Naturally, ColdFusion came to mind.

After brainstorming, we felt that we could take the base concept of a portal and create an environment providing an easy-to-use development framework. It would be a secure environment with functionality available via "tabs." Each tab would provide separate areas of content with at least one of the tabs providing typical "channelized" portal content. For example, you could simply click on one tab to view typical portal channels such as weather, stock quotes, current courses, and so on and then click on another "tab" to gain access to another area such as users tools. After much investigation and contact with Macromedia, we decided all this could be accomplished with ColdFusion version 5.0 (which was in alpha testing at the time), utilizing some of the newer features promised (e.g. UDFs, query of queries, etc).

Discovering Fusebox

Phil accepted the role of coordinating the portal framework development efforts. And, thus began a search that ultimately led to the Fusebox methodology. At that time, Fusebox 2.0 was just getting a good start within the development community. However, it was clear that the concepts



were solid, easy to understand and simple to implement. It was a breath of fresh air to happen upon a methodology that felt simply transparent and provided a level of abstraction suitable for our needs. Therefore, after a few e-mails with Steve Nelson and many cups of coffee, we were encouraged to take the methodology as a starting point for our efforts and modify it to suit our needs. That's exactly what we did.

The Fusebox specification and development methodology gave us a way to organize project files and abstract global functionality for our developers. Security, customer preferences and organization are all handled by the portal framework. In the past, we simply did not have a good method to accomplish these goals.

Though some additions to the standard Fusebox specification were made to file prefixes, the most drastic deviation from the specification was actually locating all fuses such that they are not directly embedded within other fuses as added functionality. Thus, control of what goes where and who gets what isn't implied by the browser location. Rather, it is managed by the framework that maintains roles-based security information about each fuse (i.e. students and alumni only have access to view their grades and instructors only have access to submit final grades for their courses).

Any fuse can call any other fuse. As this is done, the framework maintains a "breadcrumb" trail that gives the feel of depth through navigation. There are four types of fuses that give developers the capability of creating everything needed within the Fusebox methodology. The four types of fuses are hidden or system, channels, applications, and tabs.

- **Hidden** or system fuses are used by other fuses for functionality. These are never listed as part of the "breadcrumb" trail.
- **Channel** is probably the most familiar fuse and the one most recognized when referring to a portal. Channels are what you would see in most typical portal implementations. We are all familiar with channels that display the local weather, stock tickers, or even custom information such as course schedules or login information. They have a channel interface that is displayed in a "channelized" layout. They

can also contain edit capabilities as well as expand to full size (no longer in a "channelized" layout) when appropriate.

- **Application** fuses are usually associated with information that is rarely updated or actions not requiring a channel display. Linking these together in numerous ways can give added functionality and intuitiveness. Applications are associated with categories that give them the capability of being displayed as groups in a large listing. This type of fuse would become a direct link to a self-contained application such as course grades for students, advisee listings for advisers, course registration, etc.
- **Tab** is the final type of fuse. The use of tabs provides intuitive navigation within a portal environment. The tab fuse contains information about what falls within each tab type and how to display that information (channelized layout with preferences, application listing and single application interface).

Design Work Begins

With a firm development methodology in place and a solid development platform in ColdFusion 5.0 (by then in beta testing), we had a base allowing us to begin design work on a prototype. Though most of our enterprise data collected through our currently implemented COBOL legacy system resided in an IBM DB2 database on an IBM mainframe, we chose to develop the databases to support our new environment in Microsoft SQL on an independent server.

After approximately ten months of planning, arguing, and long nights that ended with sleeping on the floor (well, that was mainly Phil, hehe), we emerged with some carpet fuzz, a strong coffee smell, and our first working prototype. While needing refinement, we knew instantly that this would be a success.

During the development of the portal software and into the beta testing phase, the members of our talented development team were creating the applications using ColdFusion that would run within our managed environment and ultimately breathe life into our software drawing customers back again and again. Whether visiting to check grades or discuss cur-

rent events using the threaded discussion forums, it was the applications that were to bring the campus to our software. Without the talent of our developers and the functionality of these applications, the portal would simply be an empty shell.

Beta Testing and Launch

During load testing prior to release to the campus, we became concerned about performance and failover given our 20,000 students during peak access times. To solve this problem, we consulted with our enterprise server vendor, which resulted in our portal being deployed across a load-balanced and high availability server cluster consisting of six dual processor Dell servers managed by a BIGIP hardware load balancing solution (developed by F5). We instantly saw a performance improvement that would be able to handle our highest of loads as well as gaining the ability to guarantee uptime even when individual servers were removed from the cluster.

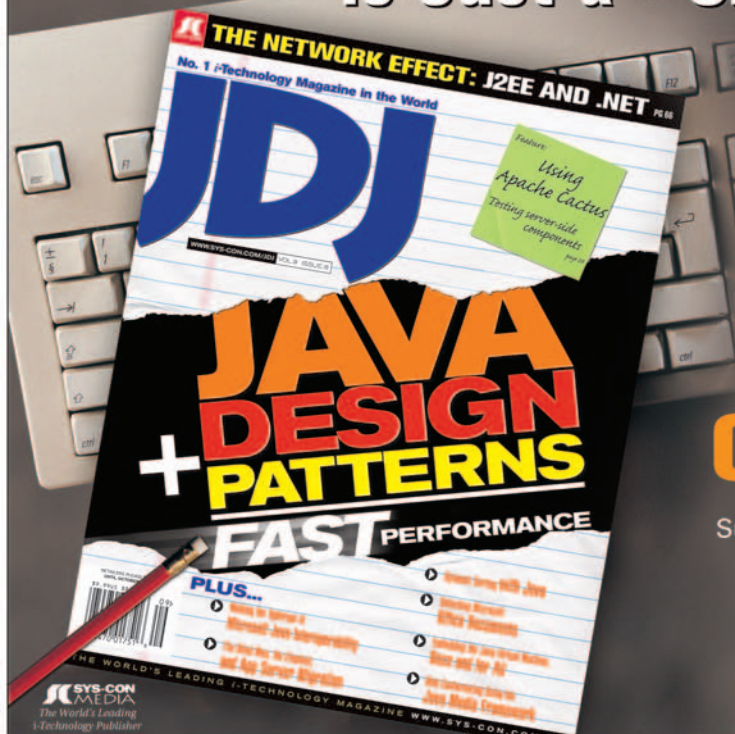
As we brought together our software in preparation for the final release, testing returned such impressive results that we actually planed to release our first version of software using the ColdFusion 5.0 beta. With little announcement and fanfare, we released our portal software as the "ECU Onestop" to the campus community.

The Portal Itself

So how does all this work? The first time anyone visits the portal, they see the "home" tab. This has been designated as the start tab for everyone. And, of course, it is associated with the "everyone" role (more about roles later). Within this tab are channels. Thus, this tab coordinates channels by using a system (hidden) fuse. Within this fuse, all channels that are related to this tab are called and organized.

In addition, the home tab contains a login channel. This channel passes authentication credentials back to the portal, which in turn verifies and forwards the customer to the appropriate location via the framework's security. In our case, authentication consists of multiple sources including a <cldap> call to Microsoft Active Directory and uses failover to insure authentication if at

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



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

Only **\$69⁹⁹** ONE YEAR 12 ISSUES

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

www.SYS-CON.com/JDJ
or **1-888-303-5282**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.ACTIVEPDF.COM	866-468-6733	27
BLOG-N-PLAY	WWW.BLOG-N-PLAY.COM	888-303-5282	49
CFDYNAMICS	WWW.CFDYNAMICS.COM	866-233-9626	4
CFUNITED	WWW.CFUNITED.COM	301-424-3903	COVER III
COLDFUSION DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/COLDFUSION	888-303-5282	39
EVI SERVERS	WWW.EVISERVERS.NET	800-504-SURF	3
HAL HELMS, INC	WWW.HALHELMS.COM		29
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM/CFDJ	877-215-4678	23
INTERAKT ONLINE	HTTP://KTL.INTERAKTONLINE.COM/	4031 401.68.19	6
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800-379-7729	COVER IV
JAVA DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/JDJ	888-303-5282	37
LINUXWORLD MAGAZINE	WWW.LINUXWORLD.COM	888-303-5282	44
MACROMEDIA	WWW.MACROMEDIA.COM/go/video5	415.252.2000	COVER II
MACROMEDIA WPS	WWW.MACROMEDIA.COM/go/webupdate	415.252.2000	11
MX DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/MX/SUBSCRIPTION.CFM	888-303-5282	41
SAVVY SOFTWARE	WWW.BESAVVY.COM	866-870-6358	17
SEAPINE SOFTWARE	WWW.SEAPINE.COM/WHITEPAPER.PHP	888-683-6456	9
SYS-CON E-NEWSLETTERS	WWW.SYS-CON.COM	888-303-5282	41
SYS-CON PUBLICATIONS	WWW.SYS-CON.COM/2001/SUB.CFM	888-303-5282	45
SYS-CON REPRINTS	WWW.SYS-CON.COM	201-802-3026	30
WEB SERVICES EDGE 2005	WWW.SYS-CON.COM/Edge2005	201-802-3066	33
WEBAPP CABARET	WWW.WEBAPPCABARET.COM/cdj.jsp	866-256-7973	13

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

Don't Miss CFDJ's Next Issue!



Macromedia's own CF charting engineer, Erik Tierney, speaks out about **Charting Capabilities in CFMX 7**
Leveraging Objects within Macromedia Flex Applications by Dennis Baldwin
Guy Rish Revisits his Cold Cup O'Joe series on Java
Jared Rypka-Hauer looks at interapplication messaging with publisher/subscriber patterns
Rob Munn talks about using Java and ColdFusion to solve complex Web Services consumption problems and much more...

all possible. Using the same authentication source as our campus e-mail system gives our customers one less password to remember. Once a successful login is provided, everyone is forwarded to the "tools" tab containing a listing of all applications based on a customer's role(s).

Permission Structure

Before we continue, we must mention the roles-based permissions structure used. When a customer is successfully authenticated, his or her role or combination of roles is determined. Unlike some portal adaptations, our primary roles of larger groups (i.e. students, employees, advisers, instructors, etc.) are not manually maintained or batch updated. Instead, we developed a method of dynamic roles. These roles are dynamically updated based on designated information queried from various databases (i.e. human resource DB, student DB, etc.). However, if needed, roles can be added manually as is typical in most portals.

A typical dynamic role would be that of a student. Once a login is authenticated, the information collected from the LDAP directory is used to execute a series of dynamic role routines. At this point, queries are made to the student database and others to determine what roles this particular customer may have (i.e. freshman, current student, graduate, instructor, staff). Of course anyone may have one or more roles. For example, it is possible for someone to be a student and an employee. The value of dynamic roles is that it provides updated permissions in real-time without having to manually maintain data for large groups. If a student were to be hired by a department, by the time he or she returned to the portal, their new role would be instantly available. The obvious question that arises from this type of update is overhead and delay at time of authentication. However, this was built with a feature that, based on an easily changed parameter, will limit how often these roles are refreshed. So, if need be, this can be increased for high loads. Also, we have a mechanism that allows anyone to update their roles at any given time.

Tools and Applications

As mentioned previously, it is these

roles that determine the applications available to a customer via the "tools" tab. Typical tools available to someone with student permissions would be course schedules and grades, course registration, exam schedules, etc. Instructor tools would include course rosters, final grade entry, etc. Tools available to employees would include access to his or her pay information, employment opportunities, service request system, etc. At this point, the total number of applications available to the campus community is very close to the century mark.

Many of these applications are written completely in ColdFusion using Fusebox, given the simplicity of the methodology. However, some are simple fuses functioning as an interface to in-house or third-party J2EE applications, COM objects or Web services. This allows us to maintain a diverse development team and allow freedom to develop applications in a variety of ways. For example, "course schedules and payments" was written using ColdFusion/Fusebox, which also relies on a J2EE solution to for e-commerce.

The "my page" tab gives authenticated customers the ability to add, remove and organize their own personally preferred channels. The capability to have more of this type of tab is available. However, we have discovered that simplicity is not only a valuable lesson in art... it's a necessity in software usability (just ask our help desk staff). Default settings for what are considered the most appropriate channels for this tab are loaded when an account is automatically created. This type of tab's interface gives the capability to minimize, detach, edit and remove channels designated as having these capabilities. Some of our most popular channels are Current Courses (Students only), Personal Bookmarks, Weather, University Newspaper Top Stories, News Feeds and People Search. Many of these are syndicated XML content consumed from other sources. Most content from other sources is cached for performance purposes.

Personalization

An always-available "personalize" link gives customers the ability to personalize their personal my page tab or change the look and feel of the entire portal.

Personalization options for this type of tab include adding/removing/rearranging channels and choosing between a two or three column layout. Customers currently using a typical portal will find this feature most familiar. In addition, customers have the ability to select a "theme" for the portal. A wide variety of themes with custom graphics and colors give customers the ability to change the entire look of the portal with a simple click of the mouse.

Other tabs within this implementation include "community," which offers a place for anyone to communicate through various roles-based discussion lists, a "profile" tab which allows anyone to see a quick snapshot of their standing at the university and an "e-mail" tab which is obviously a single application tab that offers an IMAP interface to our university mail servers.

Lessons Learned

While the implementation of our ECU Onestop portal has provided benefits to our campus community, our development staff has also reaped rewards. The use of ColdFusion has allowed us to develop powerful channels and applications in relatively short periods of time. This fact, in conjunction with some development features and APIs made available within our portal, have provided a great combination.

While searching for a solution, we found the difficulty in development and deployment of a channel in other portal implementations a flaw. Developing a channel for this portal is extremely simple and could be done with only two files (application.cfm and index.cfm). However, since we display information and are using Fusebox, we like to break out any and all display into a third file (dsp_whatever.cfm). There are specific fuseactions required by the channel standard depending on the functionality of the channel. More complex channels may require edit and other additional capabilities. These all give the channel focus (no longer in a channelized layout) after which the channel functions much like an application. Whether developing a channel that interacts with an internal application or a channel that processes and displays a typical RSS feed, channel development time is minimal.

Subscribe Today!

SAVE 16%

12 Issues for **\$89⁹⁹**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE



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

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

ColdFusion Developer's Journal





Much like the simplicity of developing an ECU Onestop channel, a simple application could be developed with as little as four files. These include `application.cfm`, `app_locals.cfm`, `index.cfm` and a single display file. There are no required fuseactions for applications. A default fuseaction is set within the `app_locals.cfm` file through a custom tag that is used to maintain the ever helpful “bread crumb” trail.

Main APIs

One of our goals in the beginning was to extend the functionality of our portal and applications through the use of programming interfaces. Though they are continually growing, the main APIs available are for security, customer or person, permissions or roles and display (allowing applications to utilize the “themes” available providing a consistent “look and feel”).

The security API provided by the portal framework gives each channel or application the ability to check the current customer’s permissions for any other channel or application. This prevents the mistake of sending someone to a dead-end security error. In addition, this security API, among other things, gives the ability to encrypt and decrypt sensitive data that may sometimes need to be passed between applications through the SSL client connection. Developers also have the ability to check for specific roles for the customer. The “`IsDemo()`” function is a perfect example of the need for this. Other examples would be the need to check and allow more granular options available within an application (administrator, etc.).

The customer or person API provides access to current customer information. The notion of supplying APIs to developers provides an easy way to reference much needed information while maintaining a level of abstraction that drastically simplifies maintaining an environment such as this. For example, application developers no longer have to worry with developing queries to identify the user of the application as common information such as name and id are provided by this API.

The display API gives developers easy access to the “themes.” Inclusion of these APIs allows the developer to develop applications with a consistent look and feel and improved usability.

Implementation of such interfaces drastically reduces development time and eliminates redundant code. Additional APIs are constantly being added for a variety of purposes including content formatting. As with most, our ultimate goal is to entirely abstract our display from our “business logic” while providing developers access to common functionality. This gives application developers a continually growing list of functionality simply built in for their convenience. Need the current semester? How about the department of the employee? Maybe I would like to know if a student is senior. All of these are a simple function call away.

The Future

To date, the ECU Onestop (now running smoothly using ColdFusion MX 6.1) has experienced over 15 million individual logins by nearly 83,000 unique customers. A snapshot of our current customer base shows usage by nearly 98% of the student population and nearly 93% of our remaining campus community. While all is not fuzzy kittens and rainbows, our implementation has been successful. Though it would be easy to enjoy this success, we are now doing what developers do best...developing the next version of our portal.


As before, we have chosen the newest beta release of ColdFusion, “Blackstone” (now CFMX 7 of course) as our foundation. In addition, the latest evolution from Fusebox, Mach II, is taking shape to become the methodology upon which our newest portal and internal applications will be built. We are also developing methods allowing content display to be abstracted and managed more by the environment giving back even more valuable time to the developer. Other features include a single-sign-on module allowing single login access to external applications as well as JMS integration giving the portal and internal applications the ability to send and consume pertinent messages. These are truly exciting times! We are currently planning a prototype release during the spring of 2005. Carpet fuzz and coffee makers get ready.

Conclusion

In the end, two main ingredients made the success of the ECU Onestop

portal possible: a dedicated, talented staff – and Macromedia ColdFusion. An additional testimony to a successful implementation surrounds the current purchase of a new, campus-wide, enterprise solution. As with most, this particular company offers an enhanced version of an open-source portal to provide access to their web-enabled software. At the end of our preliminary analysis, it was determined that our “in-house” portal provided greater levels of functionality, flexibility and customer service, while also providing a better environment for software development at a much lower cost and with less overall maintenance.

Therefore, the ECU Onestop remains and will provide access to this new enterprise system. Again, the talent of a small group and ColdFusion was able to outperform a product produced by a multi-million dollar corporation with more resources (and probably more coffee).

Whether it is the access to information or customization features desired by the campus community or the simplified, “drop in and run” approach to applications and channels, the ECU Onestop has proven a huge success. Though presented at many national conferences, nothing is more gratifying and an indication of success than walking through campus and hearing people call what we have created “their” Onestop. 

About the Authors

Steven Forehand is the team manager for the New Technologies Development Group, a team of twelve talented Internet application developers, at East Carolina University located in Greenville, North Carolina. He has been using Macromedia ColdFusion since just prior to version 2 and has over nine years of software development experience.

Phil Hulsey serves as project coordinator and lead developer for the OneStop portal project at East Carolina University, Greenville, North Carolina. He has nine years of software development experience and over seven years experience working with Macromedia ColdFusion.

forehands@mail.ecu.edu

hulseyj@mail.ecu.edu

Looking to Stay Ahead of the *i*-Technology Curve?

Subscribe to these **FREE** Newsletters >

Get the latest information on the most innovative products, new releases, interviews, industry developments, and *i*-technology news

Targeted to meet your professional needs, each newsletter is informative, insightful, and to the point.

And best of all – they're FREE!

Your subscription is just a mouse-click away at www.sys-con.com

IT solutions
JOURNAL

NET JOURNAL

JDA
JOURNAL

WebServices
JOURNAL

information
STORAGE
SECURITY
JOURNAL

LINUXWORLD
MAGAZINE

wireless
BUSINESS TECHNOLOGY

LINUX BUSINESS
WEEK

XML JOURNAL

MX
developer's journal

WebSphere
JOURNAL

wldj
JOURNAL

ColdFusion
Developer's Journal

SYS-CON
MEDIA

The World's Leading *i*-Technology Publisher

A new tool for MX professional developers and designers...

ADVERTISE

Contact: Robyn Forma
robyn@sys-con.com
(201) 802-3022
for details on rates
and programs

SUBSCRIBE

[www.sys-con.com/
mx/subscription.cfm](http://www.sys-con.com/mx/subscription.cfm)
1 (888) 303-5282

SYS-CON
MEDIA



MX
developer's journal



ColdFusion + Model-View-Controller = CFMVC

Developing a new framework for a membership organization by tying together CF & MVC



By Sam Farmer

In any membership organization with many in-house applications, the ability to share members' data among applications is vital. Previously, applications would duplicate functionality or make members leave one application to update their

new address. This didn't work well for members, nor was it easy to maintain by developers.

Using a new framework, CFMVC, allowed us to make improvements. Our new membership system will allow our paper submission application to let members change their institution with the same front and back end code the membership application uses.

In developing a new framework we needed it to handle the integration of over twenty different databases and applications, work with multiple developers, have low processing and be straightforward. We achieved it by utilizing the benefits of ColdFusion and the Model-View-Controller (MVC) architecture.

ColdFusion

Though it's arguable whether or not ColdFusion is an object-oriented development environment, it does support many object-oriented features and has many constructs that allow developers to reuse code and functionality. ColdFusion Components (CFCs) are ideal for all data interactions and business logic while normal ColdFusion pages, User Defined Functions (UDFs) and Custom Tags are ideal for handling display portions.

All applications, regardless of size, contain the following pieces:

- There will be some way of displaying information to a user – most likely, for ColdFusion developers, via a Web browser.
- Another piece will process any changes the user makes generally by writing to a database.
- A third piece will direct the user to a new item to be displayed.

Sound like the pieces of your applications? Now it's not just web applications that have these tiers, pretty much all applications contain them – even going back to the days when computers were the size of rooms and disco was cool for the first time.

Model-View-Controller

The Model-View-Controller (MVC) architecture was designed over 25 years ago by Trygve Reenskaug. Originally designed for Smalltalk programmers, it has been used in many languages from Visual Basic to Java, and is especially common among Object Oriented programmers. Companies embracing it include Apple and IBM.

MVC separates an application into the three pieces or tiers mentioned earlier. What exactly does MVC consist of?

- The model layer handles all core data interaction whether with a database, XML files or any other data store along with other business interactions.
- The view layer displays data to the user.
- Any action the user takes goes through the controller layer, which calls the needed parts of the model layer before calling the next view layer to display.

Make sense? If not, bear with me and it soon will!

By combining ColdFusion's best practices and the proven architecture of MVC, a framework emerges that modularizes code and allows for it to be used in multiple applications. (Benoit

Hediard also has a similar idea for merging together MVC and CF, see <http://www.benorama.com/>.)

Tying Together CF and MVC

Where does ColdFusion code live and where does it use components, pages and custom tags?

Model

Most applications use at least one database. The model tier retrieves data from the database, stores this data, and provides functionality for interacting with the database. If your application uses text files or some other data storage mechanism, then your model tier will interact with that entity. My MVC implementation involved creating an entity CFC for each table in the database (for example, PeopleEntity.cfc) that handles select (get), update and insert statements (set) and delete if required. A utility CFC is used when returning multiple records, joining other tables and processing functions that require more than one entity, like adding a new member – which requires inserts into a members and an addresses table. All code in the model layer must be written so it can be reused in other applications and should not use session, application or request variables; rather, values should be passed in via attributes. Other CFCs, to be shared, are also added in the model layer for e-mail, SOAP calls, LDAP, and so on.

Each CFC should be a object and contain only functions (methods) that relate to it. Functions should either perform one action only or be a wrapper function and call other functions.

View

The view tier is where all display files reside. Custom tags are used extensively for presentation code and can be called from any application if needed. Like the model layer, they must not include session, application or request variables. For some presentation purposes UDFs may also be needed.

We did two things that are not part of MVC but were needed for ColdFusion implementation. First, all forms use the post method and custom tags that contain a form accept two attributes: formAction and formMethod, which are placed in the form to pass data to the correct controller. This allows for calls from anywhere and the capability to post to anywhere. Second, all applications have at least one applicationSkin file (see membershipSkin.cfm) in the view layer to allow for an application appearance to be set. Multiple skins can then be used if, say, an admin version is intended to look different to a normal version.

Controller

The controller tier is on the web root and has no pretense of being reused. Indeed they should only be used by the application. There are two types of controller files: view-controller and model-controller. View-controller files call code in the view tier and sometimes add customization themselves. Model-controller, as the name suggests, handle all interaction with the model layer but it's important to remember that code here is not intended for reuse and extra features for this application can be added for example an e-mail or calling a web service.

As part of the ColdFusion implementation, view-controller files are standard cfm files that call custom tags. Model-controller files are CFCs, are accessed remotely but as are below an Application.cfm (or Application.cfc) file fit into whatever security system an

application deploys. Once a model-controller file has finished processing, it uses cflocation to call the next view-controller file to ensure that data is processed only once if a user reloads.

If you have trouble separating your files into their appropriate areas of the MVC framework, the code in that file is probably doing too much. Try breaking it down into smaller parts.

Folder Structure

Where does all this code sit? As controller files are used only by the application they all belong in the application web root, i.e. webroot/membership/. View-controller files, one for each area of the application, sit in this folder (see Figure 1). This allows for freedom in naming and short URL's. Features that apply to that area only, such as navigation, can also be added easily. Model-controller files are all placed in a controller folder i.e. webroot/membership/controller/.

The rest of our code, in the model and view tiers, is intended to be shared. Below the webroot we set up a folder called cfshare and within that a CFC, tag, and UDF folder (see Figure 2). A merging of ColdFusion, MVC and our desire to share code amongst applications led to such a folder set up. All model files are CFCs leading to a natural home for them, so beneath the CFC folder an additional folder is created for each application to store Entity, Utility and other components.

A similar set-up occurs for view files, all of which are custom tags. Likewise within the tag folder is an additional folder for each application. For the times when UDFs are used they follow the exact same structure under the UDF folder.

To make accessing easier in our code, a ColdFusion mapping is set up to the cfshare folder. Once achieved to access the peopleEntity in the model tier simply call the component:

```
<cfinvoke component="cfshare.cfc.membership.peopleEntity" method="init"></cfinvoke>
```

And to access the applicationSkin (using CFMX 6.1):

```
<cfimport prefix="/cfshare/tag/membership/" taglib="view">
<view:membershipSkin>
</view:membershipSkin>
```

Or for 6 and backwards:

```
<cf_membershipSkin>
</cf_membershipSkin>
```

Code

Let's look at two code examples, which can be downloaded from www.sys-con.com/coldfusion/sourcecode.cfm; first the display

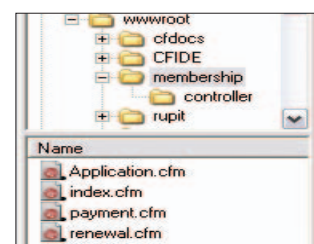


Figure 1:

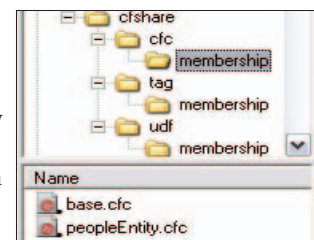


Figure 2:



page for a member to join and second the processing of the collected data.

Code Example One: index.cfm

The first example uses a view-controller called index.cfm.

We start by importing a taglib and giving the prefix of view to enable the calling of custom tags (such a method shows future developers exactly where the custom tag is). Then we param the URL display to a default value of "join". Next we open up the membershipSkin call inside of which is a cfswitch statement which will call the desired display. The default value of join is picked and the "people" custom tag is called with some variables passed in. The cfswitch statement and membershipSkin custom tags are then closed.

Let's step back and look at the people custom tag. First it performs a check to see if it has already been called and if so exits, to prevent duplicate output. Then it calls the peopleEntity cfc and outputs a simple form to allow for the collecting or editing of data. The three attributes it accepts are important: nextAction and nextMethod work in tandem – nextAction contains the path to the processing controller ("controller/membershipController.cfm" in this case) and nextMethod is the name of the function that will be called in the controller file. Passing in a value of -1 for peopleID allows for blank values to be created, but if a value of a real member had been passed in then those values would have been displayed.

Code Example Two: process-controller

How does this get processed? This is an example of a process-controller with the membershipController.cfm being called.

The nextMethod attribute in the people custom tag was set to a hidden field called method which when passed to a cfc remotely will trigger that a function of the provided name. For this example it is join. The access attribute in join function is set to remote to allow for it to be called over the web. The function creates an object to the peopleEntity in the model layer and then calls the set function with the passed in values. The peopleEntity then either runs an update if a valid peopleID is passed or an insert. Just like that, we have a new member! The join function then uses cflocation to relocate the user to the next page.

Summary

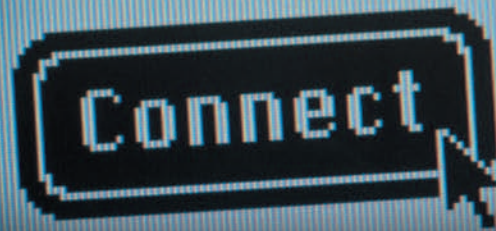
The CFMVC framework discussed above is based on a proven, successful architecture in MVC and has a low processing overhead for ColdFusion. The effect of splitting an application into tiers – model, view and controller – allows us to write applications that can share both data and functionality easily. But the benefits will also extend to complete standalone applications.



About the Author

Sam Farmer has been working with ColdFusion since version 3.1 and is currently a senior developer and architect for the American Society for Engineering Education in Washington, DC.

s.farmer@asee.org



Subscribe Today!

Connect online

for fastest service...

don't miss another

issue of LWM!



SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY
\$49⁹⁹ 12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

LOG ON TO www.LinuxWorld.com



The World's Leading IT-Technology Publisher

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO \$340 AND RECEIVE UP TO 3 FREE CDs!*



RECEIVE
YOUR DIGITAL
EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to **\$210⁰⁰**
Pay only \$99 for a 1 year subscription plus a **FREE CD**

- 2 Year – \$179.00
- Canada/Mexico – \$189.00
- International – \$199.00

6-Pack

Pick any 6 of our magazines and save up to **\$340⁰⁰**
Pay only \$199 for a 1 year subscription plus **2 FREE CDs**

- 2 Year – \$379.00
- Canada/Mexico – \$399.00
- International – \$449.00

9-Pack

Pick 9 of our magazines and save up to **\$270⁰⁰**
Pay only \$399 for a 1 year subscription plus **3 FREE CDs**

- 2 Year – \$699.00
- Canada/Mexico – \$749.00
- International – \$849.00

CALL TODAY! 888-303-5282

☐ LinuxWorld Magazine

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 /	Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Int'l - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Int'l - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ JDJ

U.S. - Two Years (24) Cover: \$144	You Pay: \$99.99 /	Save: \$45 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$69.99 /	Save: \$12
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$89.99 /	Save: \$40
Int'l - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE \$198 CD
Int'l - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ Web Services Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Int'l - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Int'l - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ .NET Developer's Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Int'l - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE \$198 CD
Int'l - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

☐ Information Storage + Security Journal

U.S. - Two Years (24) Cover: \$143	You Pay: \$49.99 /	Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$39
Can/Mex - Two Years (24) \$168	You Pay: \$79.99 /	Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 /	Save: \$34
Int'l - Two Years (24) \$216	You Pay: \$89.99 /	Save: \$126 + FREE \$198 CD
Int'l - One Year (12) \$108	You Pay: \$59.99 /	Save: \$48

☐ Wireless Business & Technology

U.S. - Two Years (24) Cover: \$120	You Pay: \$49.00 /	Save: \$71 + FREE \$198 CD
U.S. - One Year (12) Cover: \$60	You Pay: \$29.99 /	Save: \$30
Can/Mex - Two Years (24) \$120	You Pay: \$69.99 /	Save: \$51 + FREE \$198 CD
Can/Mex - One Year (12) \$60	You Pay: \$49.99 /	Save: \$10
Int'l - Two Years (24) \$216	You Pay: \$99.99 /	Save: \$20 + FREE \$198 CD
Int'l - One Year (12) \$108	You Pay: \$69.99 /	Save: \$2

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.

TO ORDER

• Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

☐ MX Developer's Journal

U.S. - Two Years (24) Cover: \$143	You Pay: \$49.99 /	Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$79.99 /	Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 /	Save: \$34
Int'l - Two Years (24) \$216	You Pay: \$89.99 /	Save: \$126 + FREE \$198 CD
Int'l - One Year (12) \$108	You Pay: \$59.99 /	Save: \$48

☐ ColdFusion Developer's Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Int'l - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE \$198 CD
Int'l - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

☐ WebSphere Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129.00 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Int'l - Two Years (24) \$264	You Pay: \$189.00 /	Save: \$75
Int'l - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

☐ PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Int'l - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Int'l - One Year (12) \$180	You Pay: \$179 /	Save: \$1

☐ WLDJ

U.S. - Four Years (24) Cover: \$240	You Pay: \$99.99 /	Save: \$140 + FREE \$198 CD
U.S. - Two Year (12) Cover: \$120	You Pay: \$49.99 /	Save: \$70
Can/Mex - Four Years (24) \$240	You Pay: \$99.99 /	Save: \$140 + FREE \$198 CD
Can/Mex - Two Year (12) \$120	You Pay: \$69.99 /	Save: \$50
Int'l - Four Years (24) \$240	You Pay: \$120 /	Save: \$120 + FREE \$198 CD
Int'l - Two Year (12) \$120	You Pay: \$79.99 /	Save: \$40

*WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm

SYS-CON
MEDIA



ColdFusion U

For more information go to...

U.S.

Alabama
Huntsville
Huntsville, AL CFUG
www.nacflug.com

Alaska
Anchorage
Alaska Macromedia User Group
www.akmmug.org

Arizona
Phoenix
www.azcfug.org

Arizona
Tucson
www.tucsoncfug.org

California
San Francisco
Bay Area CFUG
www.bacflug.net

California
Riverside
Inland Empire CFUG
www.sccflug.org

California
EL Segundo
Los Angeles CFUG
www.sccflug.org

California
Irvine
Orange County CFUG
www.sccflug.org

California
Davis
Sacramento, CA CFUG
www.saccflug.org

California
San Jose (temporary)
Silicon Valley CFUG
www.siliconvalleycfug.com

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

California
Long Beach
Southern California CFUG
www.sccflug.org

Colorado
Denver
Denver CFUG
www.denvercfug.org/

Delaware
Kennett Square
Wilmington CFUG
www.bvcfug.org/

Delaware
Laurel
Delmarva CFUG
www.delmarva-cfug.org

Florida
Jacksonville
Jacksonville, FL CFUG
www.jaxfusion.org/

Florida
Winter Springs
Gainesville, FL CFUG
www.gisfusion.com/

Florida
Plantation
South Florida CFUG
www.cfug-sfl.org

Florida
Tallahassee
Tallahassee, FL CFUG
www.tcfug.com/

Florida
Palm Harbor
Tampa, FL CFUG
www.tbmmug.org

Georgia
Atlanta
Atlanta, GA CFUG
www.acflug.org

Illinois
East Central
East Central Illinois CFUG
www.ecicflug.org/

Indiana
Avon
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana
Mishawaka
Northern Indiana CFUG
www.ninmug.org

Iowa
Johnston
Des Moines, IA CFUG
www.hungrycow.com/cfug/

Kentucky
Louisville
Louisville, KY CFUG
www.kymug.com/

Louisiana
Lafayette
Lafayette, LA MMUG
www.cflib.org/acadiana/

Maryland
Lexington Park
California, MD CFUG
<http://www.smdcfug.org>

Maryland
Rockville
Maryland CFUG
www.cfug-md.org

Massachusetts
Quincy
Boston, MA CFUG
www.bostoncfug.com

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

Minnesota
Brooklyn Park
Twin Cities CFUG
www.colderfusion.com

Missouri
Overland Park
Kansas City, MO CFUG
www.kcfcfusion.org

Missouri
O'Fallon
St. Louis, MO CFUG
www.stlmmug.com/

New Jersey
Princeton
Central New Jersey CFUG
<http://www.cjcfug.us/>

Nevada
Las Vegas
Las Vegas CFUG
www.snfcug.com/

New York
Albany
Albany, NY CFUG
www.anycfug.org

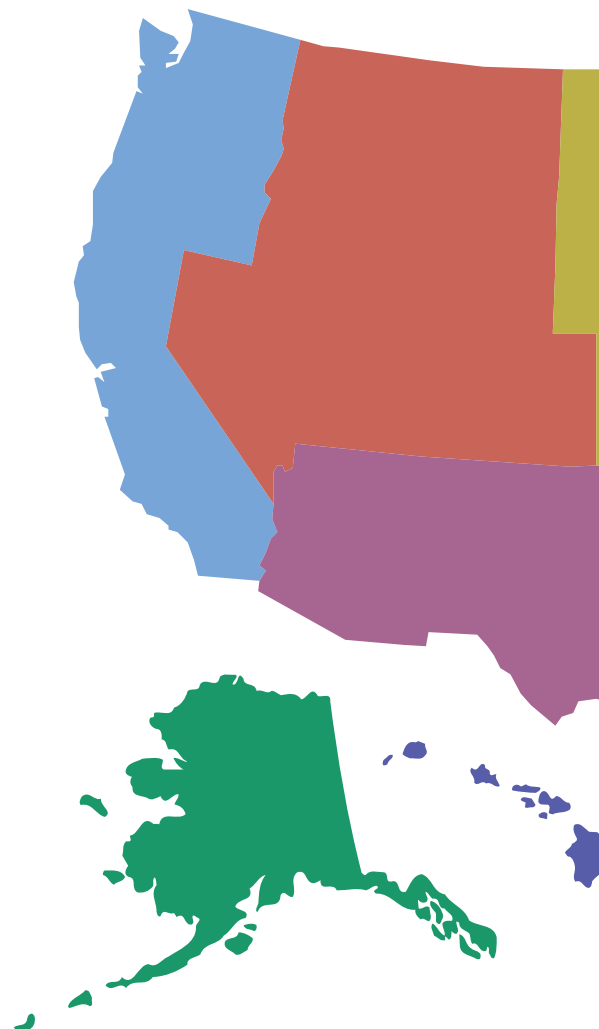
New York
Brooklyn
New York, NY CFUG
www.nycflug.org

New York
Syracuse
Syracuse, NY CFUG
www.cfugcny.org

North Carolina
Raleigh
Raleigh, NC CFUG
www.ccfug.org

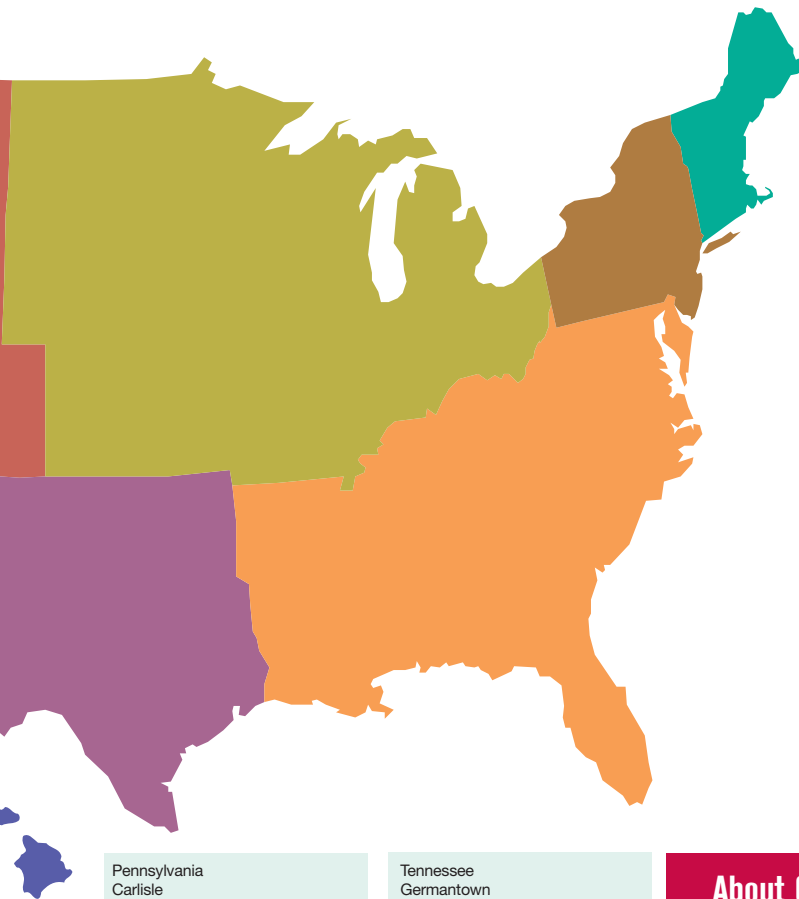
Ohio
Dayton
Greater Dayton CFUG
www.cfd Dayton.com

Oregon
Portland
Portland, OR CFUG
www.pdxcfug.org



User Groups

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



Pennsylvania
Carlisle
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Exton
Philadelphia, PA CFUG
www.phillycfug.org/

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

Rhode Island
Providence
Providence, RI CFUG
www.ricfug.com/www/meetings.cfm

Tennessee
LaVergne
Nashville, TN CFUG
www.ncfug.com

Tennessee
Germantown
Memphis, TN CFUG
www.mmug.mind-over-data.com

Texas
Austin
Austin, TX CFUG
www.cftexas.net/

Texas
Corinth
Dallas, TX CFUG
www.dfwcfug.org/

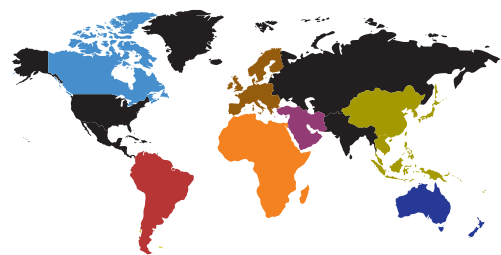
Texas
Houston
Houston Area CFUG
www.houcfug.org

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

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
www.actcfug.com

Australia
Queensland CFUG
www.qld.cfug.org.au/

Australia
Southern Australia CFUG
www.cfug.org.au/

Australia
Victoria CFUG
www.cfcentral.com.au

Australia
Western Australia CFUG
www.cfugwa.com/

Italy
Italy CFUG
www.cfmentor.com

Japan
Japan CFUG
cfusion.itfrontier.co.jp/jcfug/jcfug.cfm

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

Brazil
Brasilia CFUG
www.cfugdf.com.br

Brazil
Rio de Janeiro CFUG
www.cfugrio.com.br/

Brazil
Sao Paulo CFUG
www.cfugsp.com.br

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Ireland
Dublin, Ireland CFUG
www.mmug-dublin.com/

Spain
Spanish CFUG
www.cfugspain.org

Switzerland
Swiss CFUG
www.swisscfug.org

Thailand
Bangkok, Thailand CFUG
thaicfug.tei.or.th/

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org



Climbing Mountains – Challenging Yourself with Design Patterns

Introducing...the resource pool



By Simon Horwith

Novice programmers often ask for advice about the best way to learn ColdFusion. Fortunately for them, there is a wealth of curricula, web sites, knowledge bases, books, tutorials, and other materi-

als to help a budding young developer blossom into a decent ColdFusion developer.

What about the more seasoned developers? What resources are there to help them improve? One thing that I always suggest to experienced developers looking to improve their skills, is to try their hand at solving puzzles or at writing generic, flexible solutions to common problems.

A good way to do this is by coding generic implementations of common object-oriented design patterns. I've written several APIs and design pattern implementations for fun in the past. When I realized that other developers were interested in doing the same and in seeing how others approached this, I began writing articles about some of the generic and not so generic APIs I have written. Most recently I wrote about creating an SVG Gantt Chart API [*CFDJ* volume 7 issue 1], and in the October 2004 issue I wrote about creating a generic implementation of the Data Transfer Hash (DTO) J2EE design pattern [*CFDJ* volume 6 issue 10]. I received a lot of positive feedback from readers who want to see more articles that not only explain design patterns but also how to implement them. This month's issue being focused on architecture, I thought it only appropriate that I try my hand at implementing yet another design pattern in CFML and writing about it. This time around I decided to introduce the Resource Pool design pattern.

Resource Pools in General

The idea behind the resource pool design pattern is relatively simple. You have a central object that contains many threads that other objects can request and release. It's similar to a "factory" pattern, only in a factory pattern when an object asks the

factory for a specific object, the factory creates a new object instance and returns it. A resource pool is different in that it is just that – a "pool" of resources that are already instantiated and waiting to be put to use.

Like the factory pattern, you need look no further than under your nose (actually, under ColdFusion's hood) to see examples of resource pools at work. One classic example of the resource pool pattern at work is in the way the ColdFusion Application Server handles database connectivity. Have you ever noticed areas in the ColdFusion Administrator where you can define the number of available threads to perform actions such as the number of available connections to a datasource as well as thread behavior such as whether or not to maintain those database connections? Well these settings are really just parameters that define resource pools that ColdFusion uses to handle requests for certain functionality or outside resources. The example of database connection threads is the classic one and is easy to understand – CF has a certain number of connections to the database that it keeps alive and ready to accept requests to pass SQL to a datasource. This resource pool of database connections optimizes (speeds up) database connectivity in our applications because each time ColdFusion processes a CFQUERY tag it does not have to look-up the datasource information and establish a connection to the database – it simply requests a connection from the pool.

Resource Pools in a CFML App

There is still the question of what use a resource pool would be in a CFML application – and it's a good question indeed. The truth is that, by and large, ColdFusion applications don't need a resource pool implementation. The application server itself has many underlying resource pools so that you don't have to – it handles database threads, threads for asynchronous HTTP request processing, memory access, and much more. That said, a resource pool implementation in CFML is not a complete waste. I am usually the first person in a room to claim that ColdFusion Components are by far the most significant feature ever to be introduced to the CFML programming language; however, that's not to say that CFCs don't have any negative qualities. Specifically, there is a lot of overhead that comes with the actual instantiation of a CFC. In fact, if it weren't for the ability to persist CFCs in the application and session scopes, I wouldn't recommend their widespread use.

In a high traffic site or an application with the demand for an unusually large number of objects in memory at any

moment, reusing CFC instances that are already in memory can offer significant performance benefits. An example of this might be a high traffic eCommerce site – having a pool of available shopping cart objects in memory might drastically speed things up.

The resource pool design pattern can also be used to restrict access to resources in an application. One example would be to prevent simultaneous log-ins in a secure application. Supposing I had an application with many different user accounts that can authenticate, but I want to prevent the same username and password from being used to log in simultaneously - I want to prevent one user from creating many sessions either from several different browsers or several different physical machines at once. Another example of controlling access to objects would be in a content management system (CMS). A CMS allows people to log in as content authors; a resource pool with one object instance for each editable piece of content would ensure that no two authors could attempt to edit the same content at once. These are just a few examples – there are many other scenarios where a resource pool pattern could be used to enhance performance and/or thread access to system objects.

Implementing a Resource Pool

So how do you actually implement a resource pool in code? There are several ways to build a resource pool – most should involve a CFC that internally stores an array or structure of object instances. At the very least this CFC should have a method for requesting a new object instance from the pool. There are many other features one could add. The following describes how I implemented a generic resource pool.

My resource pool application consists of two files: a resource pool CFC and a custom tag, which gives applications an easy interface to use the CFC. Any implementation of the resource pool API that I created also requires a third file – an XML file that defines all of the resource pools that you want to instantiate. The resource pool CFC has an init method that requires that you define for each pool: the (full package) name of the CFC that this pool contains

instances of, the default minimum size (number of instances) for the pool, the growth rate for the pool, the maximum size the pool is allowed to grow to, and a timeout. The timeout is a number, representing the maximum number of seconds that may pass between one request for a specific instance and the next. The growth rate is a number representing how many additional object instances to create in the pool when all of the threads in the pool are currently in use. If the maximum size is 0 then growth is never restricted, otherwise the number of object instances in the pool will never exceed this number.

When the resource pool CFC is instantiated, it stores all of the parameters passed to its init method in its private variables scope and creates a private structure that is the actual pool. It then loops from 1 to the default pool size value and calls an “addResource” method. Add resource doesn’t simply create an instance of an object. In order to persist, control access to, and track timeouts for instance there is some additional information required about each instance in the pool. Add resource creates a new key (with a UUID value) in the “pool structure.” This new key is also a structure containing a “dts,” “vacant,” and “object” key. The “object” key value is a new object instance, the “vacant” key value is a Boolean representing whether or not that object is currently in use, and the “dts” key is a timestamp used to track the date of last access. Before creating a new key in the pool structure, the add resource method first calls a private method that returns a Boolean value specifying whether or not the pool will accept a new thread (i.e. has the maximum size been reached). That method also calls a “clean up” method that sets the “vacant” property to “true” for any instance that has timed out and deletes any object instance that is currently not in use until the minimum pool size is reached.

In writing the resource pool implementation I had to violate one of my own best practices. As a best practice I try never to use the “this” scope in component instances, preferring to use the private “variables” scope instead. Unfortunately, the need arose to set

THREE REASONS TO

blog-n-play.com

1 Get instantly published to 2 million+ readers per month!

blog-n-play™ is the only **FREE** custom blog address you can own that comes with instant access to the entire i-technology community. Have your blog read alongside the world's leading authorities, makers and shakers of the industry, including well-known and highly respected i-technology writers and editors.

2 Own a most prestigious blog address!

blog-n-play™ gives you the most prestigious blog address. There is no other blog community in the world that offers such a targeted address, and comes with an instant targeted readership.

3 Best blog engine in the world...

blog-n-play™ is powered by *Blog-City*™, the most feature rich and bleeding-edge blog engine in the world, designed by Alan Williamson, the legendary editor of *JDJ*. Alan kept the i-technology community bloggers' demanding needs in mind and integrated your blog page to your favorite magazine's Web site.



www.TAMI.linuxworld.com

"Many blogs to choose from"

PICK YOUR MOST PRESTIGIOUS ADDRESS

IT Solutions Guide	MX Dev. Journal
Storage+Security Journal	ColdFusion Dev. Journal
JDJ: Java	XML-Journal
Web Services Journal	Wireless Business &Tech.
.NET Dev. Journal	WebSphere Journal
LinuxWorld Magazine	WLDJ: WebLogic
LinuxBusinessWeek	PowerBuilder Dev. Journal
Eclipse Dev. Journal	

3 MINUTE SETUP

Sign up for your FREE blog Today!

blog-n-play.com
i-Technology Blogs Read by Millions *beta*

— This site will go beta February 15, 2003

a property in the “this” scope of each instance in the pool. Though every instance in the pool occupies a unique key in a pool structure, that ID alone is not adequate to identify the object.

The reason for this is as follows: imagine a user requests a thread. On each request they ask the resource pool for the thread they are currently using – identifying it by the UUID name of the pool key in which it exists. A second person comes to the site and also asks for a new thread. The first user’s object, which is in use, has timed out and so the resource pool gives our second visitor the first user’s object. Then the first user does ask for their object (suppose they’d just gone to get a cup of coffee and the pool has short timeouts). How does the resource pool know that it has given that user’s object to another user? When an object instance is returned to the pool to be reused, I cannot simply change that UUID value. The only way to do this would require deleting the key and creating a new one, which defeats the purpose of the pool – remember, you can’t “duplicate()” CFC instances.

In order to get around this, I create an “rPoolID” property on each object instance – assigning it a UUID value. Clients use this UUID value to specify which object in the pool they would like. Whenever a resource is returned to the pool for reuse, the instance is kept in memory – only the UUID on its public property is reset to a new UUID. In our scenario above, when our first user asks for their object instance (which has been reallocated to another client) the resource pool will return another vacant instance from the pool. Yes, this does mean that any data that was in the old instance will be lost, but that’s what happens when things time out. It’s important to note that if the object that the pool contains instances of is one that holds client-specific properties, then when that object is returned to the pool (and subsequently given to another client) it will still contain the last client’s data. In order to keep the resource pool generic and flexible it does not call an “init” method or any other constructor on an object instance when it’s put to use. To

get around this, either call the init method from the client the first time an instance is requested (or whenever the instance’s public ID value changes) or modify the resource pool itself. Note that modifying the resource pool itself will most likely make it less (or not at all) generic.

The resource pool component also contains methods for getting and setting certain global pool parameters such as the default minimum thread size, a method for getting an array of structures that contains information about the current pool member statistics, and a method for manually returning an instance to the pool rather than waiting for the timeout period to expire.



The custom tag is relatively simple – it gives a simple interface for requesting an (existing or new) object from the pool, releasing an object instance back to the pool, and for reinitializing one or all resource pools. On every call, the tag expects the path to an XML file containing definitions for all of the resource pools to create as well as the fully qualified package name of the resource pool CFC. On each request if the “application” scope does not contain a key called “stRPool”, the tag will create that key as a new structure, open and parse the XML file, and will create a new resource pool as the value of a key in “application.stRPool”. The name of each pool is specified by a “name” attribute in each <pool> node in the XML file (a <pool> node defines one resource pool).

Summary

So that’s about it – one relatively simple CFC, one very simple custom tag, and an XML file definition are all it

takes to implement a flexible and generic resource pool implementation in any CFML application running on CFMX.

I didn’t take advantage of any new features in CFMX 7 (nor did I need to), so that people could use the resource pool on CFMX 6.1 as well as CFMX 7. In any application that uses this API, I would most likely persist the ID of a user’s object instance retrieved from the pool in the session scope. In an application running on CFMX 7, you could take advantage of the new event framework and have threads automatically retrieved and returned to the pool in the “onSessionStart” and “onSessionEnd” methods in Application.cfc (see my article in last month’s issue for more on the event framework).

The code for my generic resource pool application, along with a simple sample that allows you to monitor and interact with several pools at once, is available for free download from <http://www.horwith.com/downloads/rpool.zip>. I suggest looking for other design patterns to try your hand at implementing in CFML, and give it a shot! Let me know what you come up with and what the development process was like. If you think you’d like to write about it in an upcoming issue of *ColdFusion Developer’s Journal*, don’t hesitate to let me know.

...

On a side note, I will be speaking about API creation and code reuse at the Powered by Detroit conference in April and will be giving a six hour hands-on session on implementing design patterns and writing APIs to maximize code reuse at the CFUnited conference at the end of June. I am also currently in the process of writing a book as well as a class curriculum that teaches developers everything that they need to know about software architecture with ColdFusion. If you still want to know more, are interested in finding out more about the class, book, or CFUnited session, or you have some suggestions or advice to offer regarding the topic, please don’t hesitate to send me an e-mail at simon@horwith.com.



simon@horwith.com

Recent News:
 Microsoft is Gold sponsor
 Joel Spolsky (of Joel on Software)
 to give keynote
 Pre-Conference classes added for
 Monday and Tuesday
 Hotel block price expires 4/15/05

Accessibility / Usability

Raymond Camden
 Michael Smith

Advanced CF

Michael Dinowitz
 Geoff Snowman

coldfusion

Sean Corfield

Deployment / Platform

Tim Buntel

Christian Cantrell

Matt Liotta

Simon Horwith

Charles Arehart

Jeff Peters

Manager
 Empowered Programming

Steve Drucker

Ben Forta

MX Integration

Hal Helms

Dave Watts

Shlomy Gantz

Boot Camp



June 29 - July 1, 2005

Washington DC Area

7th Annual ColdFusion Conference

3 full days!

New Venue

75% Bigger



www.cfunitied.com

Produced by
 TeraTech
 Programming
 www.teratech.com
 301.424.3903
 info@cfunitied.com



CFUN has become the premier CF specific event, and Michael Smith and his team deserve all sorts of praise for their hard work in pulling it all off yet again.

Ben Forta

"...this event really is the best gathering in the world for people developing or managing CF systems. It's here that we can understand what happened, hear what's happening, and learn what's going to happen. You can't beat it."

Chuck Hoffman

"Great place to network yourself and pick up new techniques and ideas. Also to meet ones peers, and see what the future holds for all those involved with ColdFusion"

Daniel Gregorio

"Introductions to the latest developing techniques. Get inspiration in new ways to develop projects. Ge"erally, to "re-ignite" your cf "fire" by being part of a group excited by and interested in cf development."

Kathleen Ballard

Intermedia.NET...

Now serving the hottest ColdFusion.

At Intermedia.NET we go beyond the industry standard by supporting the hottest new Coldfusion software, offering power like never before. For nearly a decade, we've been providing reliable, secure hosting to thousands of companies across the globe. We can do the same for you.

Intermedia.NET's premier hosting services include:

- ColdFusion MX hosting
- Custom tag registration
- Competitive plans
- Verity collections search engine
- Security sandboxes
- Guaranteed service levels

Unprecedented power, unmatched reputation...
Intermedia.NET is your hosting solution.



INTERMEDIA.NET

Call us at: **1.888.379.7729**

e-mail us at: **sales@intermedia.NET**

Visit us at: **www.intermedia.NET**