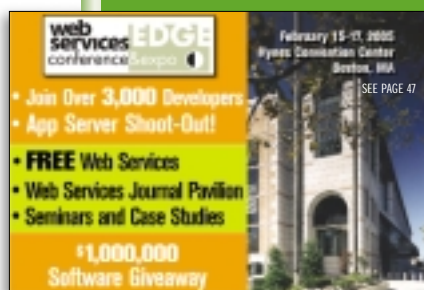


COLDFUSION Developer's Journal

ColdFusionJournal.com

December 2004 Volume: 6 Issue: 12



February 15-17, 2005
Plymouth Convention Center
Boston, MA

- Join Over 3,000 Developers
- App Server Shoot-Out!
- FREE Web Services
- Web Services Journal Pavilion
- Seminars and Case Studies
- \$1,000,000 Software Giveaway

SEE PAGE 47

Editorial

Change Is in the Air

Simon Horwith
page 5

Community Focus

Blackstone Talk Abounds

Simon Horwith
page 7

<bf>on<cf>

Blackstone at MAX

Ben Forta
page 24

CFUGs

page 34

RETAILERS PLEASE DISPLAY
UNTIL FEBRUARY 28, 2005



SYS-CON
MEDIA



FLEX YOUR COLDFUSION MUSCLES

By Matt Woodward page 12

Frameworks: Forums onTap

A framework case study



Isaac Dealey
8

Stored Procedures: Procedures, Wizards, and Dangerous Things: What Categories

Reveal About the Mind Designing and building stored procedures and transforming those procedures into ColdFusion



Joseph Flanigan

26

CF101: Creating a Component to Help You Collect Addresses

Squaring away address fields



Jeffry Houser

36

Techniques: Thinking Outside the Table PART 2 Storing search results

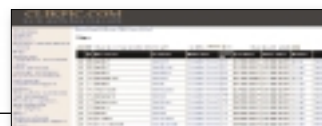


Tom Peer

40

Foundations: Designer Coffee and the Decorator Pattern

Avoiding the problem of class explosion



Hal Helms

44

macromedia®





This is:

☐ Video

☐ Flash

☐ Don't bother me. I'm staring.

The line between Flash and Video is forever gone.

Travel with us to Scandinavia for a behind the scenes tour of the award-winning, genre-bending Volvo V50 website.

macromedia.com/go/volvo





Dedicated Server Packages Starting at \$189/mo.

All dedicated servers include:

- ▶ FREE STATS SOFTWARE!
- ▶ No long term commitments!
- ▶ FREE SQL server access!
- ▶ FREE MAIL SOFTWARE!
- ▶ Fair and simple pricing!
- ▶ Optional server maintenance!

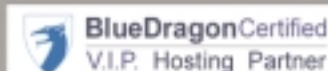
As one of the premier ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to provide **better service** to ensure your satisfaction. Not only do we offer the **finest in shared hosting plans**, but we now offer the **finest in 100% dedicated server plans**! Now you can afford the freedom of having your own dedicated server!

When your needs have outgrown shared hosting look to CFDynamics for **total freedom**. With dedicated server packages they're not offering an oxymoron; "virtually private" or "virtually dedicated" is **NEITHER** private nor dedicated. CFDynamics offers a solution that is **100% completely dedicated**. They don't play games with the fake stuff; CFDynamics only offers the real deal. Real Service. Real Satisfaction. Real Value.

Real Freedom.



Paper|Thin Partner



Visit us online or call to order!

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

executive editor

Jamie Matusow jamie@sys-con.com

editor

Nancy Valentine nancy@sys-con.com

associate editors

Gail Schultz gail@sys-con.comNatalie Charters natalie@sys-con.com

assistant editor

Seta Papazian seta@sys-con.com

research editor

Bahadır 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.comRichard Silverberg richards@sys-con.comTami Beatty tami@sys-con.comAndrea Boden andrea@sys-con.com

contributors to this issue

Isaac Dealey, Joseph Flanigan, Ben Forta, Hal Helms,
Simon Horwith, Jeffry Houser, Tom Peer, Matt Woodward

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 © 2004 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.comFrank 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

Change Is in the Air

As I write this, I am beginning to box up my belongings and prepare for another move across the Atlantic. I've accepted the position of CIO at AboutWeb – a staff augmentation and software development company based out of Washington, DC. I'm moving back into one of the richest ColdFusion communities and couldn't be happier about it. In addition to changes in my personal life, I am trying to introduce changes to the magazine. Beginning with next month's issue you will start to see these changes go into effect.

For one thing, I am going to try to gather articles that focus on one general subject area each month. I'd like readers to consider whether they'd be interested in writing for us – it's really not very difficult to do as long as you can make time to write the piece and exchange a couple of e-mails with me.

If you are interested in writing, the editorial calendar for 2005 is at www.sys-con.com/coldfusion/CFDJ_2005_Edi_Cal.pdf, and lists the topic of focus for each month as well as the deadline. That deadline is for the final copy, so articles are actually due anytime prior, preferably a week beforehand. Also, you can find guidelines at www.sys-con.com/coldfusion/writers.cfm.

If you don't want to write or don't know what to write about, let me know what you think of the new format(s) being introduced, and also if you'd like to write but aren't sure what to write about. I get a lot of e-mail to this effect, and I try to take into account the opinions expressed in every one of them.

With the pending release of Blackstone, there's sure to be a ridiculous number of topics waiting to be explored, examined, and explained for our readers. I should mention that though the issues will focus on specific topics, we will still be running other (off-topic) articles in each issue as well, so don't let that discourage you.

This month, we have some terrific content. Isaac Dealey has written a great overview of the onTap Framework (which he wrote and distributes free of charge). I've received a lot of e-mail about framework coverage and would like to see more coverage each month of




By Simon Horwith

MACH II, FuseBox, onTap, and any other framework you would like to know more about. Joseph Flanigan has written a good introduction to the CFSQLTool and stored procedures. Both of these articles describe one approach to automating or handling the process of implementing CRUD (Create, Read, Update, Delete) in CF applications.

Hal Helms has written a good introduction to decorator design patterns. In addition to a design pattern-focused issue next year, I foresee more discussion of architectural issues in general – especially after the release of Blackstone. Speaking of Blackstone, this month Ben Forta tells us all about what was seen and heard at this year's MAX convention.

Jeffry Houser has written an article for his "CF-101" column that examines ColdFusion Components in depth. We also feature part two of an article from Tom Peer, which explores the ways that developers can leverage their database. Matt Woodward explores Flex and walks you through rewriting a ColdFusion application user interface with Flex. In the Community Focus column, I offer a little bit of advice on preparing yourself for the release of Blackstone by way of reading blogs.


There's an old adage that says change is good. Well, I'm banking on it. *CFDJ* is your magazine. If you don't like something about it, tell me. If there's something missing or you feel you have something to offer, tell me. With all the changes coming up, I can't emphasize enough how important it is to let me know what you think of the changes and what other changes you'd like to see. My e-mail address is simon@horwith.com and I'm waiting to hear from you. 

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

Nothing beats our racks

Absolutely nothing



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!

Over 20,000 Servers!

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.

Blackstone Talk Abounds

I'll be honest – I wasn't terribly thrilled with the amount of coverage given to the new features in Blackstone. Don't get me wrong, between a full-day of hands-on sessions, sneak peeks, a couple of features shown off in the keynotes, and a "Birds of a Feather" one evening, there was plenty of information about Blackstone to be seen and heard. The fact that the entire server development team was there helped a lot. That said, if you were looking for new information about Blackstone that hadn't already been mentioned by Ben Forta on his whirlwind Blackstone tour, then there wasn't too much there for you. Obviously, by the time MAX 2005 happens Blackstone will be released and we'll see and hear it all, but that's almost a year from now. Fortunately, a few Macromedia employees have been granted permission to blog about new features in Blackstone and have volunteered their time and energy to that end.

You can learn about new features currently being planned for the release, hear about how they work, and sometimes can even see code and/or working examples! Never before has so much information been made available about an upcoming release – and personally, I like that personal blogs are being used as the format for delivering some of this information. Here are a few URLs, information on who owns the blog, and what type of Blackstone content has been blogged recently:

- **Ben Forta** (www.forta.com/blog/): Ben has long been the senior evangelist for ColdFusion at Macromedia and boy, has he been evangelising! Ben has blogged about so many new features in Blackstone, I can't list them all here. Recently, he's mentioned support for XML as a <CFFUNCTION> return type or argument type, and enhancements to the <CFPARAM> tag.
- **Damon Cooper** (www.dcooper.org/): Damon is in charge of the ColdFusion development team at Macromedia. Who better to talk about new features in Blackstone than the guy rolling up his sleeves and coding and testing on a daily basis? Damon has recently blogged about the power of an "Asynchronous Gateway" (supplying sample files and installation




By Simon Horwith

instructions), a Blackstone SMS application, and about other ideas for SMS use in ColdFusion applications.

- **Tim Buntel** (www.buntel.com/blog/): Tim is the product manager for ColdFusion at Macromedia. I'm no expert, but I believe product managers are largely responsible for deciding what features will go into a product. So we should all

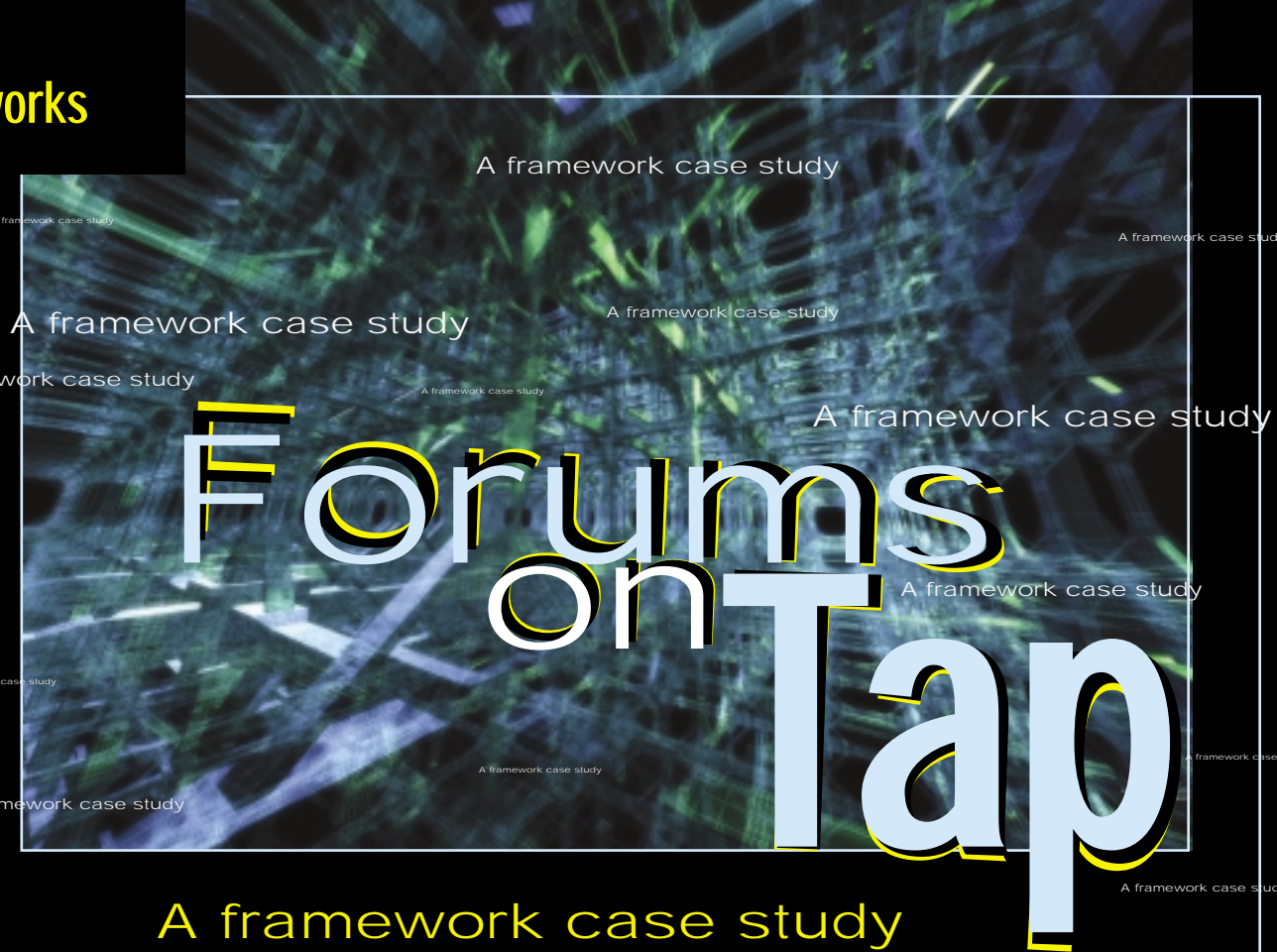
thank Tim for the cool new stuff it seems Blackstone will be able to do. Tim is spending a lot of time blogging about all sorts of great things to do with Blackstone. At the time of this writing, Tim blogged about the FlashPaper and Flash Forms application used in part of the keynote at MAX – and he has made some code examples and live demos available as well.

A few other Macromedia employees, such as Christian Cantrell (www.markme.com/cantrell/) and Sean Corfield (www.corfield.org/blog/), are also blogging about new features.

If you're interested in knowing more about new features planned for the next release of ColdFusion, I suggest that you visit the blogs mentioned in this article on a daily basis. In addition, expect to see more material about Blackstone in **CFDJ** as we approach the release – particularly upon its official launch. Next month begins a year of more in-focus issues each month...and Blackstone is sure to be a large part of that as well. 

About the Author

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal. He is a freelance software architect currently consulting with companies in London. Simon is a Macromedia-Certified Master Instructor and a member of Team Macromedia. He has been using ColdFusion since version 1.5 and consults on ColdFusion applications that leverage 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



Programming is hard work, or it can be. For developers there seems to be a tendency to write and rewrite the same code. Often it seems most of our work boils down to the fundamental four: Create/Read/Update/Delete (CRUD).

This portion of the application development life cycle can be so time consuming, so repetitive, and so tedious at times that it's all we can do to keep from getting ourselves into a rut, or continue working once we're already in one. With the volume of code this represents, it's no wonder that we turn to frameworks and life cycle processes like FLiP to provide consistent structure, accelerate development times, and bring manageability to what can otherwise be a surrealistic management nightmare of cut-and-pasted code.

For a long while the only ColdFusion framework with a following has been Fusebox. Since August of 2003 (very nearly coinciding with the appearance of the Mach-II framework) I've been working hard developing the onTap framework. Unlike other frameworks, onTap focuses on much more than the controller layer in a model-view-controller (MVC) approach to application development, including many custom tags and function libraries to extend the ColdFusion server's native APIs. All this work on low-level functionality like e-mail, caching,



By Isaac Dealey

database analysis tools, database agnostic querying, modular display components, etc., has kept me pretty busy in the past year and it's superseded any work on complete applications until very recently. I knew the framework community needed a forum – I'd known this for quite awhile but hadn't found the time to build one. I didn't want to use a tool created with another framework for reasons I hope are obvious. Now that I've finally created the Forums on Tap application and I've implemented it on the framework site, I can say that I'm quite satisfied with all the underlying work that's gone into the framework up to this point.

I won't say Forums on Tap required less time to develop due to the onTap framework because it hasn't. What I will say is that I've achieved several objectives with this application by using the onTap framework that I most likely would not have achieved if I'd used another framework or no framework at all. The time savings as a result of using the framework have, however, allowed me to achieve these additional objectives in the same amount of time that I would normally spend developing a similar application.

The Best Buggy Whips Money Can Buy

A forum application in and of itself is nothing special. There are dozens, maybe hundreds of free forums available these days for various Web development platforms. If I'm going to develop something so common, I want to make it

better than the others. With this in mind I had several objectives:

- Browser-driven installation
- Common security API
- Rich user interface
- Extensible forum API

What? No Coding?

The first objective of browser-driven installation is something I've written and spoken about before and is part of the fundamental objectives of the onTap framework. As application developers, the reasons for this may not be immediately obvious to you. If you imagine for a moment that you aren't a skilled application developer but merely an intrepid businessman who's acquired a ColdFusion-enabled shared hosting account and are setting up your first dynamic Web site, the reasons might become more obvious. As developers we've become accustomed to manually installing and configuring applications by hand-editing templates and text files with file paths, datasource names, and other information. This situation is unheard of amongst traditional desktop applications. Imagine the horror of a typical computer user when told that he or she must manually edit an ini file or Windows registry entries before Winamp, AIM, or Microsoft Word will function. In the world of desktop applications this would never be tolerated, and with onTap it's no longer necessary with dynamic Web applications.

In order to achieve this objective, I began with a copy of the onTap framework core components and added to that the onTap framework Plugin Manager application (see Figure 1). Both of these components may be downloaded from the framework Web site at www.fusiontap.com. The Plugin Manager component must be extracted or copied into the framework root directory and should then be immediate-



Figure 1: Add/Remove Components

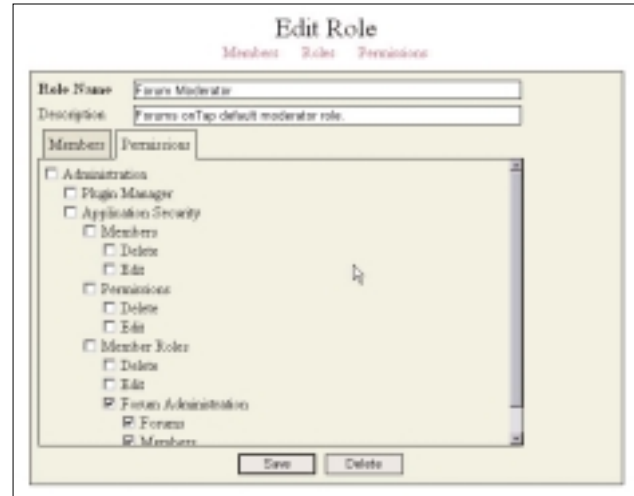


Figure 2: Roles onTap

ly functional. Unfortunately the scope of this article doesn't allow me to offer an in-depth analysis of the Plugin Manager or how plug-ins can be installed and managed. What I can say is that the installed Plugin Manager supplies a rich API that provides common tools for installing and configuring individual plug-ins like the Forums onTap application. Each individual plug-in must include its own installer to utilize the Plugin Manager API in much the same way that traditional desktop application installers interact with their target operating systems like the Windows Add/Remove Programs Wizard.

Of course creating an installer for the forum application does increase the amount of time required to develop the forum, although I think this is time well spent. Ideally a well-written installer will mean less time spent answering e-mails from intrepid developers when they're confused about how to install and configure your application.

Logins and Trade Secrets and Encryption Keys – Oh My!

When was the last time this happened to you? You visit a Web site and discover it has a members-only area. You need some information from the site so you fill out their sign-up form, check your e-mail, and log in. You look to find their forum to ask your question from others on the site and discover that you now need a membership! But wait! Didn't you just sign up to be a member on this site just a few minutes ago? What gives?! Forum applications are often installed into existing Web sites that already have some other dynamic services as well. The site owners understand the need for a forum though they don't feel they have the time to design and develop one, and why should they? With plenty of existing forums available in the community they usually choose one they like, install it as is, and leave it alone. The task of integrating the forum with their existing application member-management and security if considered is rarely undertaken. But why should visitors or users be required to have two separate logins for the same Web site?

The scenario I just described is the motivation behind creating the Members onTap and Roles onTap plug-ins for the onTap framework (see Figure 2). These two plug-ins provide a rich API for managing members (new member signups, member preferences, etc.) and roles-based permissions for complex Web applications. This allows the forum to utilize a common API for creating and managing its members and permissions, and this same API can be used with any other applications installed in your onTap framework site. Since the member and security management API is installed separately from any applications (like the forum), all plug-in applications can share the same set of members and roles, eliminating the need for multiple logins or memberships on your site.

Both the Members onTap and Roles onTap plug-ins can be downloaded from the framework site and like the forum application are installed through the Plugin Manager interface.

I Only Have I's For U

There's been a lot of talk in the community and from Macromedia in the last two years about rich Internet applications. This is the term Macromedia uses to describe applications that use Flash (or now Flex) in an application's presentation layer to offer the application users a friendlier interface that more closely resembles a traditional desktop application. This is important because while scriptable Web applications are less expensive to develop than traditional desktop applications, the limitations of working in a multibrowser environment have been a significant handicap to developing clean, intuitive applications that are pleasing to use rather than annoying or challenging for the user.

Although Flash and Flex are still challenging and expensive to implement, Web standards like CSS, XHTML, and DOM are fortunately achieving better support with the latest breed of browsers. As modern browsers like Opera, the latest Firefox, and other recent Gecko-based releases become available, it is an increasingly viable option to use DOM to create rich user interfaces inexpensively. While DOM may still have its limitations compared to Flash or Flex, it can provide most of the same functionality for free to almost as many users (or browsers if you like), it's often faster to load, and the onTap framework makes it very easy to design.

The forum application uses two user interface elements or



Figure 3: A recent forum thread

techniques in particular, which makes it unusual amidst the morass of Web forum applications. The first element is an implementation of the onTap framework's HTML forms library to display form validation (and some other less obvious niceties like clickable labels for radio buttons and checkboxes). When the forum application is installed you are given the opportunity to select client-side (JavaScript/DOM), server-side (ColdFusion), or a combination of form validation. The default is for forms to be validated both in the client browser and on the ColdFusion server.

In itself this is moderately unusual, although what's really unique about this implementation is the display of error messages. We've all seen Web forms that use either client-side validation (with its annoying JavaScript alert windows), or some form of server-side validation presenting an error or list of errors at the top of a form. Even if a form uses both forms of validation, the error messages supplied are usually delivered in their native mediums (HTML for server validation or a JavaScript alert for client-side validation). Forums onTap however delivers both server-side validation and client-side validation in the HTML medium so that there is no variation in display between errors occurring before or after the form is submitted. As an added bonus these error messages are not accompanied by a nerve-jarring alert sound effect.

The second interface element can be seen when viewing the list of messages in an individual forum thread (see Figure 3). By default the forum offers four modes in which to view forum threads. The first is a simple linear mode in which the messages are displayed in chronological order. The print mode is very similar only excluding some graphical elements. The remaining two views are described as "top" and "left" indicating the location of a DOM tree presenting a list of the messages in the thread sorted by branch. In these modes only one message at a time is visible on the page and the user may select messages to read from the tree. Recent messages are marked in the tree with a graphic and any branches containing new messages are open by default when displayed. Although obviously not for everyone, this provides a very clean and intuitive way to examine longer and more active forum threads, which often fragment into several distinct conversations making them difficult to navigate.

Both the rich forms interface and the tree navigation are examples of the onTap framework's HTML library – a very rich and powerful tool for creating modular display and an important part of achieving the next objective. Although the HTML library seems daunting at first glance (particularly to developers who are accustomed to seeing templated display code), its principals are ultimately very simple, based on the existing standards for CSS, XHTML, and DOM. Once you have a good grasp of the fundamental principles of these Web standards, the HTML library becomes quite easy to understand and use.

A Horse of a Different Color

I've saved the best for last: extensibility. The word extensibility has become something of a buzzword, immediately conjuring up images of XML or XHTML for many developers, in spite of the fact that simply using XML doesn't make an application extensible. In fact I would argue that the majority of

applications using XML today are not any more extensible than their non-XML cousins. When I say extensibility, I'm talking about something much more tangible. This element alone was the thrust of my first *ColdFusion Developer's Journal* article entitled "Features Without Fixtures." It has become my mantra for application development and the driving force behind the development of new features for the framework. What I mean when I say the forums are extensible is that I can make customizations to them, add features, remove features, change not only the formatting (as with CSS), but also mold the content or completely rearrange the interface – and I can do this all without editing a single line or even a single character of the forum code. I can...and I have!

The Forums onTap plug-in application is designed with a wider audience in mind and as such its interface offers no considerations for the formatting, layout, or navigation of any site or application it might be installed within. In my case I installed the forum application within the onTap framework documentation site – which serves a double purpose as both the documentation included with the core application (also an exercise in extensibility) and as the official onTap framework Web site. This documentation site uses a pair of iframes for its primary navigation, one to the left containing a tree with links to various documents, and one to the right containing the content. Above these frames is the site title and an area containing a hierarchical view of your current position in the framework documentation.

The forum application also has a similar hierarchy-driven navigation built into its displays (typically referred to as breadcrumbs). Although both sets of navigation work, I would much rather have a single contiguous hierarchy rather than confuse visitors with two separate hierarchies. So by using the extensibility features of the framework, I've been able to merge the forum navigation with the documentation site navigation. I've done this without editing a single line or a single character of either the forum application code or the documentation site code.

This grail of extensible development is made possible by two things in particular. The first is the framework's core feature – directory-based automation of template execution. In short, the framework examines the directories you create to locate templates to execute. New templates may be executed in any given page merely by adding the template in the appropriate directory. Each request is divided into several stages (application, onrequestend, htmlhead, local, etc.) and your new templates may be executed in any of these stages simply by knowing which directory to place your template in. In the case of the forum application I wanted to move links from the forum display into the frame-parent document. Since the forum navigation is created in the local stage, I have to add my template in the local directory after this navigation is created.

The second key to this process lies in the HTML library described above. The HTML library creates all of the elements in an HTML page as a series of structures and arrays prior to display. This provides a very easy way to make even the most complex modifications to the display at any time before the elements are written to the ColdFusion server's output buffer. If you don't like the label next to a form element, simply


locate the structure representing the table cell (or other HTML element) containing the text and change the text. Or better yet, use one of the HTML library's built-in functions designed to do just that. Simple, clean, and safe: no need for complicated regular expressions to modify HTML content created with the cfset or cfsavecontent tags.

Knowing that the navigation would be an element people might want to modify, I gave it a distinctive name to make it easy to identify in the code. I then created my new template in the forum application's local directory and applied an event-filter to the navigation element. This event filter (in the form of a ColdFusion user-defined function) is called just before the navigation element is written to the ColdFusion output buffer. Receiving the navigation element as an argument, this event filter then loops through the navigation links one at a time and appends them to the navigation element in the frame-parent document using a handy function declared in the documentation site's API. Once the documentation site's navigation is updated, it's a simple matter to suppress display of the original navigation element in the forum display using a little CSS.

This hierarchy navigation wasn't the only customization for the official framework site. I also wanted to include the forum in the framework's left-pane tree; the site's primary navigational element. This is a similarly easy task performed in the same manner by adding a template in the appropriate directory to take advantage of functions available in both the framework documentation site API and the forum API. This new template displays a branch in the tree showing the forum home page and allows users to expand the tree to see forum categories and individual forums.

If you're asking yourself why I've gone to all this trouble, the answer is simple. When I publish new versions of the framework or of the forum application, I can install them cleanly over the current versions without worrying about my modifications because the new versions don't impact any of the existing code. These are "features without fixtures".

Conclusion

I could talk for quite some time about the various merits of the onTap framework and how they've influenced this project. Obviously I have a bias and the medium of a magazine article such as this offers me little room for in-depth analysis. In summary, I've been able to meet my four major objectives of extensibility, security, rich user experience, and easy installation all by using the features available in the onTap framework – and while I haven't saved time doing this, I believe I've produced superior results. 

About the Author

Isaac Dealey has worked with ColdFusion since 1997 (version 3) for clients from small businesses to large enterprises, including MCI and AT&T Wireless. He evangelizes ColdFusion as a volunteer member of Team Macromedia, is working toward becoming a technical instructor, and is available for speaking engagements.

info@turnkey.to



Flex Your ColdFusion Muscles

Building Rich Internet Applications couldn't be easier with ColdFusion and Flex

In the beginning of the World Wide Web there was HTML, and it was good. HTML provided an easy, structured way to present data and images, and hyperlinks gave users access to other pages with the simple click of a mouse.

As time went on, however, users demanded more and more from HTML, which led to the rise of Web applications. At first the users were pleased with the ability to shop and do their banking online, but they soon became frustrated. "These Web applications don't behave like my desktop applications!" they cried. "I'm sick of waiting for page reloads every time I click on something. I want a better experience!"

Web developers have done their best to meet these increasing demands through a rather clumsy mix of HTML and other technologies such as JavaScript, DHTML, and Flash. They have met with varying degrees of success. Because of the nature of these technologies and their inherent limitations, developers are still prevented from providing desktop-like application functionality and rich user experiences within the Web browser. The time has come for a completely new Web application paradigm; one that combines the power of the Internet with the user experience of a desktop application.



By Matt Woodward

Let There Be Flex

Flex is the newest addition to Macromedia's line of server products, and it's an example of a relatively new type of server software known as a "presentation server." The sole job of a presentation server is to provide a framework on which Web applications with rich user interfaces can be built. These applications are typically referred to as Rich Internet Applications (RIAs).

Macromedia's RIA initiative began with the evolution of the Flash IDE into a stronger application development tool. Flex is another step forward for RIAs and offers many development advantages over the Flash IDE. Using Flex, developers who may be uncomfortable with the Flash IDE can quickly build RIAs in their favorite code editor. Because of the code-centric nature of Flex applications, there's no need to sacrifice enterprise application development methodologies and architectures. Best of all, Flex applications run in the extremely ubiquitous Flash player so we can be certain our Flex applications will run practically anywhere.

As if this weren't compelling enough, Macromedia recently announced Flex 1.5. This adds exciting new features to the already-rich Flex toolset such as vastly improved charting and graphing capabilities, more advanced UI components, and simpler application skinning. The performance has also been greatly improved, and with Flex 1.5, developers can also integrate their Flex applications with Macromedia Central. If you

haven't already checked out Flex, now is a great time to do so. For more information on the new features of Flex 1.5, visit www.macromedia.com/software/flex/.

In this article I'll briefly address RIAs and discuss the types of applications for which they are especially well-suited, introduce you to Flex, and discuss how Flex and ColdFusion can be used in conjunction with one another. I'll also explain how Flex can be used to put a completely new face on an existing ColdFusion application.

Improving the User Experience

The overriding goal of RIAs is to give users a richer, more interactive, and more immediate experience when using Web applications. One of the chief complaints users have about Web applications is the page-based model. Every time something is clicked or new data is needed, the user has to wait for a page refresh. Even if the page loads relatively quickly, it's still a less-than-optimal experience for the user. Particularly in the case of a multistep process such as making an online purchase, the continual reloading of pages tends to make users uneasy, and as a result, many users actually abandon the process because the experience is so poor.

Flex solves this problem by allowing developers to create a complete Web application in a single-screen model. Flex applications are able to refresh discrete portions of the screen or change from one view to another without having to load a new page in the browser. Flex also offers more advanced UI components when compared with HTML, allowing for vastly improved functionality that would be difficult or impossible to achieve with HTML. Also, because Flex applications run in the Flash player, the headaches of browser inconsistencies with DHTML are completely avoided. The end result is a far richer, more interactive experience for your users. For additional information on RIAs, visit the RIA section of Macromedia's Web site at www.macromedia.com/resources/business/rich_internet_apps/.

Obtaining Flex

You can get your hands on Flex and Flex Builder (Macromedia's Flex IDE) by downloading a trial version from www.macromedia.com/software/flex/trial. Or if you're a DevNet subscriber, you may download Flex and Flex Builder from your DevNet subscription page.

Macromedia also recently announced that a noncommercial, noninstitutional Flex license will soon be made available at no cost. This means you will be able to use Flex as a learning tool or for building personal applications and make these applications accessible to others. This is fantastic news, and I encourage you all to take advantage of this. Read Macromedia's FAQ for more information (www.macromedia.com/software/flex/productinfo/faq/#ancni). You may apply for a noncommercial, noninstitutional Flex license at www.macromedia.com/go/flexlicense.

Because there are several excellent resources covering the installation and configuration of Flex, I will not be addressing these issues. If you need assistance installing Flex, visit the Flex section of Macromedia DevNet at www.macromedia.com/devnet/flex. Or you can go to the Flex Documentation page at www.macromedia.com/support/documentation/en/flex/.

Flex Basics

At a high level, a typical Flex application consists of MXML and ActionScript code (more on this next). This code is compiled into a Flash movie (SWF) by the Flex server at runtime. Flex applications are launched in a Web browser using a URL just as with traditional Web pages. When the MXML file is requested, the Flex server either compiles the code if necessary or serves up a precompiled SWF to the user.

Concerning the languages used to write Flex applications, in similar fashion to ColdFusion, Flex applications are created by using a rich set of extremely powerful tags. These tags are written in a flavor of XML called MXML, and this serves as the base language for the creation of Flex UIs. To unleash the true power of Flex, however, ActionScript 2.0 is also used. Think of MXML as the tags used to build the basic structure of a Flex UI and ActionScript as the language that brings the UI to life (although there's a lot you can achieve with MXML alone).

If you have experience with scripting in Flash, you will already be familiar with ActionScript. If not, ActionScript is an ECMAScript-compliant language, so the Java or JavaScript skills possessed by many Web developers will dramatically ease the learning curve. Macromedia's LiveDocs for Flex is at http://live-docs.macromedia.com/flex/15/flex_docs_en/index.html. This is an extremely handy reference as you build Flex applications.

Where Does ColdFusion Fit In?

One frequent point of confusion surrounding Flex is that many ColdFusion developers see Flex as a replacement for ColdFusion. Nothing could be further from the truth. As mentioned previously, Flex is a presentation server. Thus, on its own Flex cannot do many of the things we ColdFusion developers do every day, such as retrieve data from a database. Database connectivity simply doesn't exist in Flex because this isn't Flex's job. Flex can, however, easily communicate with any number of back-end technologies such as simple HTTP services, Web services, and remote Java or Flash objects.

ColdFusion, on the other hand, excels at providing powerful back-end functionality, but the presentation layer is not addressed directly by ColdFusion. This is precisely where ColdFusion fits into the Flex equation. Through the use of CFML pages and ColdFusion Components (CFCs) exposed as Web services, we can use our existing ColdFusion skills to create back-end components and easily leverage these components with Flex. Because Flex functionality can be exposed as a Java tag library, you can also combine Flex and JSP or CFML code within a single page, but this is beyond the scope of this article. You can read more about this on Christopher Coenraets's blog: www.markme.com/cc/archives/004021.cfm. As you'll soon see, using Flex with ColdFusion is an extremely powerful combination of technologies that allows us to build compelling RIA experiences with a minimal amount of code.

Case Study: A Forum Facelift with Flex

Volume 8 of Macromedia's DevNet Resource Kit (DRK) includes an excellent ColdFusion forum application by Ray Camden called Galleon. Because this is an existing application that's available to DevNet subscribers (hopefully that includes you!), Galleon provides a nice code base on which we can build a

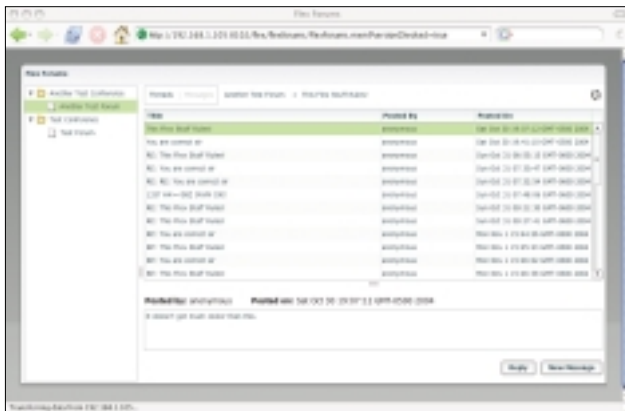


Figure 1: A new Flex face for Galleon

new Flex user interface. In addition, because of the numerous screens involved with an HTML-based forum application, it's particularly well-suited to illustrate the dramatic difference between an HTML UI and an RIA built in Flex.

Galleon comes complete with user-management features and an administration module. However, in the interest of focusing on Flex basics, these portions of Galleon have been omitted from the Flex application. Session management and some of the finer points of Flex development will be left for a future article. But armed with the knowledge you'll gain from this article, you can complete the Flex UI for Galleon on your own! (Consider that a homework assignment.)

The ColdFusion Code

The existing Galleon code provides an excellent head-start in creating a Flex UI because Galleon already leverages CFCs quite heavily. This makes exposing the CFC methods to Flex as simple as setting the access attribute of the methods to "remote," which instantly creates a Web service that can be called from our Flex application. Although the original Galleon application includes several CFCs, for simplification purposes I have extracted the relevant functions and put them all in a single CFC called FlexForums.cfc (see Listing 1).

We will also be using a simple CFML page in the Flex application to illustrate how Flex can leverage CFML pages as HTTP services. There is also a "gotcha" in Flex when XML is retrieved via a Web service call. To make a long, hair-tearing story short, if you're using ColdFusion to generate XML and you want to use this XML in your Flex application, you must use a CFML page as opposed to a CFC function that returns XML. (This is a known issue that should be fixed in future versions of Flex.) We'll use an HTTP service call to a CFML page to populate the forum navigation tree in our Flex application (see Listing 2).

The Flex Code

In addition to our CFC and CFML page, building our entire Flex UI takes only a little over 200 lines of MXML and ActionScript code. That's not a typo. The power of Flex allows developers to be extremely productive and build impressive applications with impressively little code.

Our Flex application is broken down into four files.

FlexForums.mxml (see Listing 3) is the main application file, and this is what's called in the Web browser to launch the application.

FlexForums_script.as (see Listing 4) is an ActionScript file that contains all the functions that are used in the application. This could have been included in a script block within FlexForums.mxml. However, separating the ActionScript code from the MXML file makes for cleaner code organization. Two additional MXML files, ThreadForm.mxml and MessageForm.mxml (see Listings 5 and 6), contain the forms used for adding new threads and posting messages to the forums.

Setting Sail on the Galleon RIA

To see the HTML version of Galleon, visit Ray Camden's Web site at www.camdenfamily.com/morpheus/forums/. We're going to use this exact same back-end code to drive our new Flex UI. Due to the current licensing restrictions with Flex, unfortunately I can't point you to an online version of the Flex application, but you can see a screenshot in Figure 1.

The new Flex face we've put on Galleon is the familiar interface of many desktop newsreader programs. The default "Halo" style of Flex creates an attractive, clean UI with no graphic design work whatsoever. All the Flex components as well as the application itself are quite easily skinnable using standard CSS syntax. Thus, you're by no means stuck using this look for all your applications.

Apart from the rich interface itself, the most compelling aspect of the Flex application is its behavior. When the user clicks on a specific forum, for example, the messages are retrieved from the database and the data grid is updated without requiring a refresh of the entire screen. When new threads or messages are posted by the user, the data grids automatically refresh immediately and display the latest data. Navigating between forums, threads, and messages is all immediate and again requires no reloading. Even retrieving specific messages is a simple, quick call to the server that updates one portion of the Flex display without having to refresh the entire page. This creates a far more cohesive, seamless experience for the user.

Retrieving Data From ColdFusion

As mentioned previously, Flex can retrieve data from ColdFusion in two ways: CFML pages can be called as an HTTPService, or CFCs can be exposed as Web services. (A third method is to call a CFC as a RemoteObject, but for this article we'll focus on HTTP services and Web services.) For security reasons, access to remote services from Flex is controlled via a white list. This is simply a section of Flex's XML configuration file that dictates what remote services may be called and how they may be called. Services can either be unnamed (called via a URL directly) or named (called via an alias), and we'll see an example of each. Flex configuration is beyond the scope of this article, so for more information about configuring the white list see www.macromedia.com/support/flex/ts/documents/whitelist.htm.

Calling a CFML Page as an HTTPService

The navigation tree in the left panel of the Flex UI provides

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



easy access to each of the conferences and forums. The data in the tree comes courtesy of an HTTPService call from Flex to the getConferences.cfm page (see Listing 2). The HTTPService is defined in Flex as follows:

```
<mx:HTTPService id="conferenceFeed" url="http://192.168.1.105:8103/cfusion/forums/getConferences.cfm" showBusyCursor="true" />
```

The ID attribute simply gives the service a handle that can be used within Flex, the URL is the location of the CFML page, and the showBusyCursor attribute tells Flex to change the cursor into an hourglass while the service is being called to give visual feedback to the user. The XML returned from the CFML page is then set as the data provider for the navigation tree and Flex handles the rest.

```
<mx:Tree id="forumTree" height="100%" width="20%" showDataTips="true" dataTipField="description" change="handleTreeChange(event);" dataProvider="{conferenceFeed.result.nodes}" />
```

The other item of interest is the change attribute, which tells Flex to call a function when a change is made to the tree. When the user clicks on a forum in the tree, this change triggers the handleTreeChange function, and this function subsequently calls the method that loads the thread data. The details of this operation can be seen in the handleTreeChange function in the FlexForums_script.as file (see Listing 4).

Calling a CFC Method as a Web Service

The second way to get data from ColdFusion into Flex is to invoke methods in a CFC that have been exposed as a Web service. This is the method most frequently used in our sample application. By using CFCs, we create better code encapsulation in our ColdFusion application while simultaneously making it easier to access this functionality from outside sources such as Flex.

As an example of this, let's take a brief look at how we use a call to a CFC to populate a Flex DataGrid (see Figure 2). The

Thread	Originator	Replies	Last Post
Does this REALLY work?	anonymous	4	Thu Nov 4 09:30:33 GMT-0800 2004
This Flex Stuff Rules!	anonymous	30	Mon Nov 1 22:39:55 GMT-0800 2004
Another New Thread for Testing	anonymous	2	Mon Nov 1 22:37:51 GMT-0800 2004
I think I have this thing figured out!	anonymous	2	Mon Nov 1 22:33:06 GMT-0800 2004
Does it still work?	anonymous	1	Mon Nov 1 22:32:34 GMT-0800 2004
Should work now	anonymous	1	Mon Nov 1 22:32:19 GMT-0800 2004
Re:thread	anonymous	1	Mon Nov 1 22:32:29 GMT-0800 2004
Another New Thread	anonymous	1	Mon Nov 1 22:34:15 GMT-0800 2004
Re:thread Test for Threads	anonymous	1	Mon Nov 1 22:32:49 GMT-0800 2004
Time for Yet Another New Thread	anonymous	4	Mon Nov 1 22:37:25 GMT-0800 2004
The Thread to End All Threads	anonymous	3	Mon Nov 1 22:36:56 GMT-0800 2004
what's Another New Thread	anonymous	1	Mon Nov 1 22:45:49 GMT-0800 2004
Time for a New Thread	anonymous	2	Mon Nov 1 22:43:49 GMT-0800 2004
Test of Word Wrap in Thread Posts	anonymous	9	Sat Oct 31 07:27:18 GMT-0800 2004
Test Another Flex Test	anonymous	6	Sat Oct 31 07:03:53 GMT-0800 2004
Better Test Another Thread Add Operation	anonymous	3	Sat Oct 31 07:09:42 GMT-0800 2004
Another New Thread from Flex	anonymous	1	Sat Oct 30 19:00:17 GMT-0800 2004
New Thread from Flex	anonymous	1	Sat Oct 30 18:59:45 GMT-0800 2004
This is a New Topic	admin	1	Thu Oct 28 09:25:00 GMT-0800 2004
Test Topic	admin	2	Thu Oct 28 09:29:38 GMT-0800 2004

Figure 2: The Flex DataGrid

DataGrid is an extremely rich component with a large amount of out-of-the box functionality such as sortable and resizable columns, so it provides a very powerful way to present data.

To fill the DataGrid with data, we simply set the result of the call to the Web service (which returns a ColdFusion query object) as the data provider for the DataGrid. Flex dutifully displays the data retrieved from ColdFusion. In our main MXML file we must first define our Web service, which in this case is a service that has been named "FlexForums" in the Flex configuration file as opposed to being called by a URL:

```
<mx:WebService id="forumWS" serviceName="FlexForums" showBusyCursor="true" />
```

Once the Web service is defined, in our ActionScript code we simply call the Web service method that returns the thread data:

```
forumWS.getThreads(forumId);
```

One item to note is that because ColdFusion is case insensitive, it returns the column names of the ColdFusion query to Flex in caps, so we must use all capital letters to refer to the column names in our DataGrid columns (see Listing 7).

Again, because we have set the dataProvider of the DataGrid to be the result of the getThreads call, the DataGrid is automatically updated with new data upon completion of the Web service call. Simple yet extremely powerful!

Next Steps

Using a relatively small amount of code, we've created a very compelling, interactive Flex interface for an existing ColdFusion application. Unfortunately, there are some configuration issues and code details that I didn't have space to discuss here, so for a more detailed analysis of the code please visit my blog at www.mattwoodward.com/blog. I'll try to have something posted by the time this article is published.

Conclusion

I obviously couldn't possibly cover everything you need to know about Flex in this brief article, but I hope this sample application has shown you how Flex can be used to build highly interactive RIAs extremely quickly and easily. Using your existing ColdFusion skills in combination with Flex, you can put a new face on your existing ColdFusion applications, or build completely new experiences for your users with this powerful combination of technologies. Now get started on that homework assignment!

Resources

Books and Articles

- Forta, Ben (2004). "On ColdFusion and Flex." *ColdFusion Developer's Journal* 6(5), 40-43.
- Horwith, Simon (2004). "Introducing ... Flex!" *ColdFusion Developer's Journal* 6(6), 8-10.
- Moock, Colin (2004). *Essential ActionScript 2.0*. O'Reilly.
- Webster, Steven and McLeod, Alistair (2004). *Developing*

THE GRASS REALLY IS GREENER ON THE **SERVERSIDE.**

SUPERIOR MANAGED HOSTING

- Intelligent Routing
- Redundancy
- Network Security
- Service Level Agreement
- Environmental Controls
- Network Uptime Guarantee
- Scalability
- OC3/SONET Backbone
- Backup Power
- Physical Security
- Fire Protection
- Server Hardware Guarantee

IF YOU'VE BEEN SEARCHING for a reliable managed hosting partner, your search for greener pastures is over. There is no longer a need to settle for inferior support and lack of accountability. ServerSide takes the guess work, and the associated hassles, out of working with a technology partner.

ServerSide provides managed web hosting solutions for a wide range of customers including; Fortune 500 corporations, small and medium sized businesses and non-profit organizations located across the United States and around the World.

We provide a single point of accountability for all your web hosting infrastructure needs — while adding value, not cost.

>> SPECIAL OFFER

Sign up online for a new shared hosting account now at www.serverside.net and ServerSide will waive the set-up fee.

ENTER CODE: mxd2004

The grass is greener at
www.serverside.net
888.682.2544
hosting@serverside.net



Rich Clients with Macromedia Flex. Macromedia Press.

Macromedia Web Resources

- Macromedia Flex Product Site: www.macromedia.com/software/flex/
- Macromedia DevNet: www.macromedia.com/devnet/flex/
- Installing Flex on ColdFusion: www.macromedia.com/support/documentation/en/flex/1_5/flexforcf.html

Blogs

- Matt Chotin's Blog: <http://markme.com/mchotin/>
- Christophe Coenraets's Blog: www.coenraets.com/index.jsp
- RIA Blog: www.richinternetapps.com/archives/cat_macromedia_flex.html
- Brandon Purcell's Blog: www.bpurcell.org/blog/index.cfm?mode=cat&catid=14

- Waldo Smeets's Blog: www.waldosmeets.com/index.cfm?categoryID=27
- Rich Internet Blog: www.richinternet.de/blog/index.cfm?mode=cat&catid=C4CE4B51-0C1B-190A-34750825DCF11D68. 

About the Author

Matt Woodward is a Web application developer for i2 Technologies in Dallas, Texas, and also works as a consultant through his company, Sixth Floor Software. He is a Macromedia Certified ColdFusion Developer, a member of Team Macromedia, and has been using ColdFusion since 1996. In addition to his ColdFusion work, Matt also develops in Java and PHP.

matt@sixthfloorsoftware.com

LISTING 1: FlexForums.cfc

```
<cfcomponent displayname="FlexForum" output="false" hint="CFC for use
with the Flex interface to Ray Camden's Galleon forum software">
    <!-- THREAD METHODS -->
    <cffunction name="getThreads" access="remote" returntype="query"
output="false" hint="Returns a list of threads.">
        <cfargument name="forumId" type="numeric" required="false"
default="0" />

        <cfset var qGetThreads = "" />

        <cfquery name="qGetThreads" datasource="galleon">
            SELECT threads.id, threads.name, threads.readonly,
            threads.active, threads.forumidfk, threads.useridfk,
            threads.datecreated, forums.name as forum,
            users.username, max(messages.posted) as lastpost,
            count(messages.id) as messagecount
            FROM threads, forums, users, messages
            WHERE threads.forumidfk *= forums.id
            AND threads.useridfk *= users.id
            AND threads.active = 1
            <cfif arguments.forumid gt 0>
                AND threads.forumidfk = <cfqueryparam value="#argu-
ments.forumId#" cfsqltype="cf_sql_integer" />
            </cfif>
            AND messages.threadidfk *= threads.id
            GROUP BY threads.id, threads.name, threads.readonly,
            threads.active,
            threads.forumidfk, threads.useridfk, threads.date-
created, forums.name,
            users.username
            ORDER BY lastpost DESC
        </cfquery>

        <cfreturn qGetThreads />
    </cffunction>

    <cffunction name="addThread" access="remote" returntype="void"
output="false" hint="Adds a thread.">
        <cfargument name="threadName" type="string" required="true"
/>
        <cfargument name="threadBody" type="string" required="true"
/>
        <cfargument name="forumId" type="numeric" required="true" />

        <cfset var qAddThread = "" />
        <cfset var qAddMessage = "" />
        <cfset var success = true />

        <!-- add thread data to threads table -->
```

```
<cftry>
    <cfquery name="qAddThread" datasource="galleon">
        SET NOCOUNT ON
        INSERT INTO threads (
            name,
            readonly,
            active,
            forumidfk,
            useridfk,
            datecreated
        ) VALUES (
            <cfqueryparam value="#arguments.threadName#"
cfsqltype="cf_sql_varchar" maxlength="100" />,
            <cfqueryparam value="0" cfsqltype="cf_sql_bit"
/>,
            <cfqueryparam value="1" cfsqltype="cf_sql_bit"
/>,
            <cfqueryparam value="#arguments.forumId#" cfsql-
type="cf_sql_integer" />,
            <cfqueryparam value="3" cfsqltype="cf_sql_inte-
ger" />,
            <cfqueryparam value="#Now()"#
cfsqltype="cf_sql_timestamp" />
        )
        SELECT @@IDENTITY AS newId;
        SET NOCOUNT OFF
    </cfquery>
    <cfcatch type="database">
        <cfset success = false />
    </cfcatch>
</cftry>

    <!-- since this is a new thread, add a message to the mes-
sages table as well -->
    <cfif success>
        <cftry>
            <cfset success = addMessage(arguments.threadName,
arguments.threadBody, qAddThread.newId) />
            <cfcatch type="any">
                <cfset success = false />
            </cfcatch>
        </cftry>
    </cfif>
</cffunction>

    <!-- MESSAGE METHODS -->
    <cffunction name="getMessages" access="remote" returntype="query"
output="false" hint="Returns a list of messages.">
        <cfargument name="threadid" type="numeric" required="false"
default="0" />
```

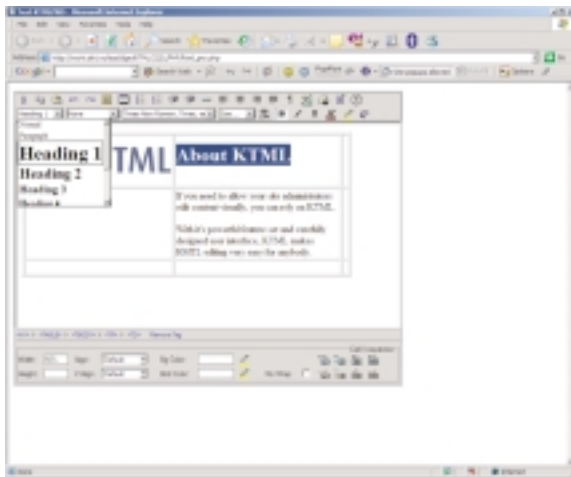
Web (r)evolution:



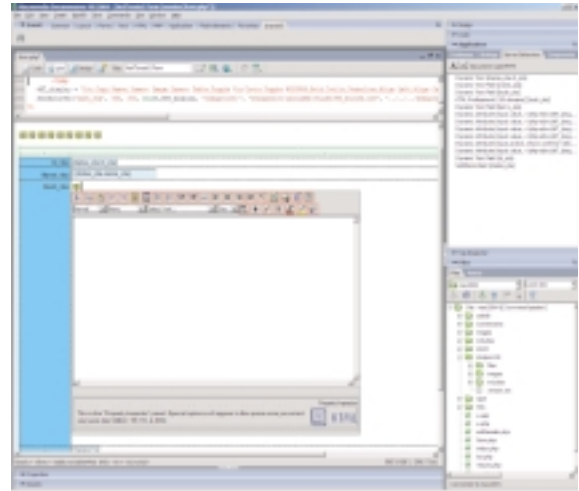
KTML

Let your clients edit their Web sites content through the browser

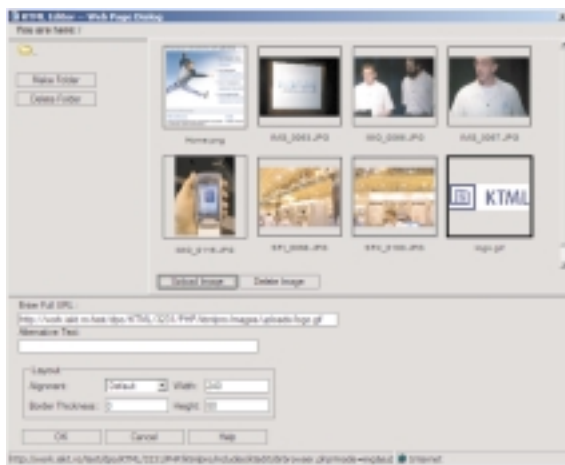
Visually use CSS styles (Word like)



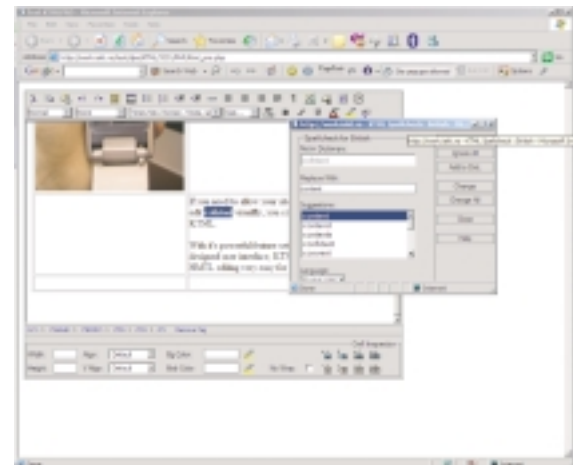
Native Dreamweaver MX integration



Easy-to-use image explorer



Multi-language spell-checker



KTML 3.5 - HTML online editor

See all the features, benefits and a live demo at:

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

```

<cfset var qGetMessages = "" />

<cfquery name="qGetMessages" datasource="galleon">
    SELECT messages.id, messages.title,
           messages.posted, messages.threadidfk, messages.userid-
fk,
           threads.name as threadname, users.username
    FROM messages, threads, users
    WHERE messages.threadidfk *= threads.id
    AND messages.useridfk *= users.id
    <cfif arguments.threadid gt 0>
        AND messages.threadidfk = <cfqueryparam value="#argu-
ments.threadid#" cfsqltype="cf_sql_integer" />
    </cfif>
    ORDER BY posted ASC
</cfquery>

<cfreturn qGetMessages />
</cffunction>

<cffunction name="getMessage" access="remote" returntype="query"
output="false" hint="Returns a single message.">
    <cfargument name="messageid" type="numeric" required="true" />

    <cfset var qGetMessage = "" />

    <cfquery name="qGetMessage" datasource="galleon">
        SELECT messages.title, messages.body, messages.posted, mes-
sages.useridfk,
               users.username, users.datecreated
        FROM messages, users
        WHERE messages.useridfk *= users.id
        AND messages.id = <cfqueryparam value="#arguments.mes-
sageid#" cfsqltype="cf_sql_integer" />
    </cfquery>

    <cfreturn qGetMessage />
</cffunction>

<cffunction name="addMessage" access="remote" returntype="void"
output="false" hint="Adds a new message.">
    <cfargument name="messageTitle" type="string" required="true"
/>

    <cfargument name="messageBody" type="string" required="true" />
    <cfargument name="threadId" type="numeric" required="true" />

    <cfset var qAddMessage = "" />

    <cfquery name="qAddMessage" datasource="galleon">
        SET NOCOUNT ON
        INSERT INTO messages (
            title,
            body,
            useridfk,
            threadidfk,
            posted
        ) VALUES (
            <cfqueryparam value="#arguments.messageTitle#" cfsql-
type="cf_sql_varchar" maxlength="100" />,
            <cfqueryparam value="#arguments.messageBody#" cfsql-
type="cf_sql_longvarchar" />,
            <cfqueryparam value="3" cfsqltype="cf_sql_integer" />,
            <cfqueryparam value="#arguments.threadId#" cfsql-
type="cf_sql_integer" />,
            <cfqueryparam value="#Now()#" cfsqltype="cf_sql_time-
stamp" />
        )
        SELECT @@IDENTITY AS newID;
        SET NOCOUNT OFF
    </cfquery>
</cffunction>
</cfcomponent>

```

LISTING 2: getConferences.cfm

```

<cfsetting enablecfoutputonly="yes" />
<cfquery name="qGetConferences" datasource="galleon">
    SELECT conferences.id,

```

```

           conferences.name,
           conferences.description,
           forums.id AS forumId,
           forums.name AS forumName,
           forums.description AS forumDescription,
           forums.readonly
    FROM conferences
    LEFT JOIN forums
    ON conferences.id = forums.conferenceidfk
    ORDER BY conferences.name, forumName ASC
</cfquery>

<cfif qGetConferences.RecordCount GT 0>
    <!-- build xml to provide to Flex Tree -->
    <cfoutput>
        <cfxml variable="conferences">
            <node>
                <cfloop index="i" from="1"
to="#qGetConferences.RecordCount#" step="1">
                    <cfif i EQ 1 OR (i GT 1 AND qGetConferences["name"][i]
NEQ qGetConferences["name"][i -1])>
                        <node label="#qGetConferences['name'][i]#"
data="#qGetConferences['id'][i]#"
description="#qGetConferences['description'][i]#" type="conference">
                            <node label="#qGetConferences['forumName'][i]#"
data="#qGetConferences['forumId'][i]#"
description="#qGetConferences['forumDescription'][i]#" type="forum" />
                        <cfelse>
                            <node
label="#qGetConferences['forumName'][i]#"
data="#qGetConferences['forumId'][i]#"
description="#qGetConferences['forumDescription'][i]#" type="forum" />
                        </cfif>
                    <cfif (i + 1 LTE qGetConferences.RecordCount AND
qGetConferences["name"][i + 1] NEQ qGetConferences["name"][i]) OR i EQ
qGetConferences.RecordCount>
                        </node>
                    </cfif>
                </cfloop>
            </node>
        </cfxml>
    </cfoutput>

    <!-- no conference/forum data retrieved, so build dummy xml that
indicates this -->
    <cfelse>
        <cfoutput>
            <cfxml variable="conferences">
                <node>
                    <node label="No Data Available" data="0" descrip-
tion="No data was retrieved from the server." type="conference" />
                </node>
            </cfxml>
        </cfoutput>
    </cfif>

<cfoutput>#conferences#</cfoutput>

```

LISTING 3: FlexForums.mxml

```

<?xml version="1.0" encoding="utf-8"?>
<!-- the main application tag; the initialize attribute tells the
application to call
the getConferences function as the application initialized so
we have data in the
navigation tree immediately after the application loads -->
<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml"
pageTitle="Flex Forums" initialize="getConferences();">

    <!-- include the ActionScript functions -->
    <mx:Script source="FlexForums_script.as" />

    <!-- create handle for getConferences.cfm file -->
    <mx:HTTPService id="conferenceFeed"
url="http://192.168.1.105:8103/cfusion/forums/getConferences.cfm"
showBusyCursor="true" />

    <!-- create handle for FlexForums.cfc; this is a named service in
the Flex whitelist -->

```



```

<mx:WebService id="forumWS" serviceName="FlexForums"
showBusyCursor="true">
  <!-- declare these methods of the CFC specifically so we can
tell it
to refresh the display upon completion of these method
calls -->
  <mx:operation name="addMessage" result="refresh()" />
  <mx:operation name="addThread" result="refresh()" />
</mx:WebService>

  <!-- the main application panel -->
  <mx:Panel id="mainPanel" width="100%" height="100%" title="Flex
Forums">
    <!-- horizontal divided box that allows user to adjust width
between the left tree
and the right-hand application panel -->
    <mx:HDividedBox id="leftNav" width="100%" height="100%">
      <!-- the navigation tree; data comes from call to the
conferenceFeed
      HTTPService, which in turn calls the
getConferences.cfm page -->
      <mx:Tree id="forumTree" height="100%" width="20%"
showDataTips="true" dataTipField="description"
change="handleTreeChange(event);"
dataProvider="{conferenceFeed.result.node}" />

      <!-- vertical box that contains the right-hand area of the
application -->
      <mx:VBox width="80%" height="100%">
        <!-- horizontal box to contain the view stack control,
navigation labels, and
refresh button -->
        <mx:HBox height="28" width="100%" verticalAlign="mid-
dle">
          <!-- horizontal box to contain just the view

```

```

stack control and
navigation labels -->
        <mx:HBox width="100%" verticalAlign="middle">
          <!-- link bar controls the thread/message
viewstack -->
          <mx:LinkBar dataProvider="gridViewStack"
borderStyle="solid" />
          <!-- text labels to show the user where they
are -->
          <mx:Label id="forumTitle" text="Please Select
a Forum From the Tree On the Left" />
          <mx:Label id="threadTitle" text="" />
        </mx:HBox>

        <!-- horizontal box for the refresh button so we
can have it right align -->
        <mx:HBox verticalAlign="middle"
horizontalAlign="right">
          <mx:Link id="refreshLink"
icon="@Embed('images/syncicon.png')" click="refresh()" tooltip="Refresh
Content" />
        </mx:HBox>
      </mx:VBox>
    </mx:HDividedBox>
  </mx:Panel>
</mx:Application>

```

Simplify Content Management With



View complete history of any task

An Ektron CMS helps organizations to effectively manage people, process, and information on the Web while driving successful Web strategies.

ektron CMS300

- GoldFusion
For - ASP
- ASPX
- PHP

Use Ektron CMS300 to:

- ✓ Rapidly deploy content management as a Web site component
- ✓ Streamline work with advanced CMS workflow and collaboration
- ✓ Assign, manage and monitor Web tasks in real-time
- ✓ Globalize sites with multilingual content management
- ✓ Bring true control and accountability to your Web efforts



Try it for FREE

Download a free 30-day, fully-functional trial or request a personalized demonstration at

www.ektron.com/cfdj

Ektron - Redefining Web Content Management

```

change="threadSelected(threadGrid.selectedItem.ID,
threadGrid.selectedItem.NAME)">
    <mx:columns>
        <mx:Array>
            <mx:DataGridColumn
columnName="NAME" headerText="Thread" />
            <mx:DataGridColumn
columnName="USERNAME" headerText="Originator" width="40" />
            <mx:DataGridColumn
columnName="MESSAGECOUNT" headerText="Replies" width="40" />
            <mx:DataGridColumn
columnName="LASTPOST" headerText="Last Post" width="70" />
        </mx:Array>
    </mx:columns>
</mx:DataGrid>

    <!-- horizontal box below data grid for new
thread button -->
    <mx:HBox id="threadButtonBox" width="100%"
height="40" horizontalAlign="right" verticalAlign="middle">
        <mx:Button id="newThreadButton"
label="New Thread" click="showNewThreadForm();" visible="false" />
    </mx:HBox>
</mx:VBox>

    <!-- vertical divided box for message list and
details allows user to control
height between the message data grid and
the message detail area -->
    <mx:VBox width="100%" height="100%"
id="messages" label="Messages">
        <!-- the data grid for the message data -->
        <mx:DataGrid id="messageGrid" width="100%"
height="50%" dataProvider="{forumWS.getMessages.result}"
change="showMessage(messageGrid.selectedItem.ID);">
            <mx:columns>
                <mx:Array>
                    <mx:DataGridColumn
columnName="TITLE" headerText="Title" />
                    <mx:DataGridColumn
columnName="USERNAME" headerText="Posted By" width="40" />
                    <mx:DataGridColumn
columnName="POSTED" headerText="Posted On" width="50" />
                </mx:Array>
            </mx:columns>
        </mx:DataGrid>

        <!-- vertical box for the message headers and
the message contents -->
        <mx:VBox height="50%" width="100%">
            <mx:HBox id="messageHeader" height="30"
width="100%" visible="false" verticalAlign="bottom">
                <mx:Label htmlText="&lt;b&gt;Posted
by:&lt;b&gt; {forumWS.getMessage.result[0].USERNAME}" />
                <mx:Label htmlText="&lt;b&gt;Posted
on:&lt;b&gt; {forumWS.getMessage.result[0].POSTED}" width="400" />
            </mx:HBox>

            <!-- text area displays the message con-
tent; note that when a CF query
is returned to Flex it comes
back as an array with one array
element per query record -->
            <mx:TextArea id="messageDetails"
width="100%" height="100%" wordWrap="true" editable="false"
text="{forumWS.getMessage.result[0].BODY}" />

            <!-- horizontal box for message buttons
(new and reply) -->
            <mx:HBox id="messageButtonBox"
width="100%" height="40" horizontalAlign="right" verticalAlign="middle">
                <mx:Button id="replyMessageButton"
label="Reply" click="showMessageForm(true);" visible="false" />
                <mx:Button id="newMessageButton"
label="New Message" click="showMessageForm(false);" />
            </mx:HBox>
        </mx:VBox>
    </mx:VBox>

```

```

    </mx:ViewStack>
</mx:VBox>
</mx:HBox>
</mx:Panel>
</mx:Application>

```

LISTING 4: FlexForums_script.as

```

// get conference data for tree
function getConferences() {
    conferenceFeed.send();
}

// handle tree change
function handleTreeChange(event) {
    /* Get the type of item the user clicked on (forum or conference).
If it's a
conference, we don't do anything, but if it's a forum, we display
the threads. */
    var type = event.target.selectedItem.getProperty("type");

    // if user clicked on a forum, load the messages
    if (type == "forum") {
        // set forumId from the id of the item the user selected in
the tree
        var forumId = event.target.selectedItem.getData();

        // set label above grids to forum name
        forumTitle.text =
event.target.selectedItem.getProperty("label");
        threadTitle.text = "";

        // set selected item in view stack to threads, clear out old
data, and show new
        // thread button
        gridViewStack.selectedChild = threads;
        newThreadButton.visible = true;
        messageGrid.removeAll();
        messageDetails.htmlText = "";
        messageHeader.visible = false;

        // load the threads; the result of this call is bound to the
datagrid
        if (forumId != undefined) {
            forumWS.getThreads(forumId);
        }
    }

    // item clicked in thread grid, so load messages
function threadSelected(threadId, title) {
    // the result of this call is bound to the datagrid
    forumWS.getMessages(threadId);
    gridViewStack.selectedChild = messages;
    threadTitle.text = "> " + title;

    // clear out any existing message data
    messageHeader.visible = false;
    messageDetails.text = "";
}

    // item clicked in message grid, so show message
function showMessage(messageId) {
    // this returns a cf query which comes back to flex as an array
with one array element
    // per row
    forumWS.getMessage(messageId);

    // show message header and reply button
    messageHeader.visible = true;
    replyMessageButton.visible = true;
}

    // show new thread form in popup window
function showNewThreadForm() {
    var threadForm = mx.managers.PopUpManager.createPopUp(_root,
ThreadForm, true, {deferred: true});
}

```

```

/* add new thread; the result attribute of the mx:operation tag in
the main
  MXML file will refresh the thread grid once the add operation
has completed */
function addThread(title, body) {
    forumWS.addThread(title, body,
forumTree.selectedItem.getData());
}

// show message form in popup window; reply boolean indicates whether
this is a reply
// or a new message
function showMessageForm(reply) {
    // open the popup
    var messageForm = mx.managers.PopUpManager.createPopUp(_root,
MessageForm, true, {deferred:true});

    // if this is a reply as opposed to a new message, pre-populate the
subject field
    if (reply) {
        var subject = "";

        // only add RE: to subject if it isn't already there
        if (messageGrid.selectedItem.TITLE.substring(0, 3) != "RE:") {
            subject = "RE: " + messageGrid.selectedItem.TITLE;
        } else {
            subject = messageGrid.selectedItem.TITLE;
        }

        messageForm.messageSubject.text = subject;
    }
}

// add message
function addMessage(subject, body) {
    forumWS.addMessage(subject, body, threadGrid.selectedItem.ID);
}

// refresh messages and threads as needed
function refresh() {
    if (gridViewStack.selectedChild == threads &&
forumTree.selectedItem.getData() != undefined) {
        forumWS.getThreads(forumTree.selectedItem.getData());
    } else if (gridViewStack.selectedChild == messages) {
        forumWS.getMessages(threadGrid.selectedItem.ID);
    }
}

```

LISTING 5: ThreadForm.mxml

```

<?xml version="1.0" encoding="utf-8"?>
<mx:TitleWindow xmlns:mx="http://www.macromedia.com/2003/mxml"
title="New Thread">
    <mx:Form>
        <mx:FormItem label="Title" required="true">
            <mx:TextInput id="threadTitle" width="300" />
        </mx:FormItem>

        <mx:FormItem label="Body" required="true">
            <mx:TextArea id="threadBody" height="300" width="300"
wordWrap="true" />
        </mx:FormItem>
    </mx:Form>

    <mx:ControlBar horizontalAlign="right">
        <mx:Button label="Cancel" click="this.deletePopUp();" />
        <mx:Button label="Submit"
click="_root.addThread(threadTitle.text,
threadBody.text);this.deletePopUp();" />
    </mx:ControlBar>
</mx:TitleWindow>

```

LISTING 6: MessageForm.mxml

```

<?xml version="1.0" encoding="utf-8"?>
<mx:TitleWindow xmlns:mx="http://www.macromedia.com/2003/mxml"
title="Post Message">
    <mx:Form>
        <mx:FormItem label="Subject" required="true">

```

```

            <mx:TextInput id="messageSubject" width="300" />
        </mx:FormItem>

        <mx:FormItem label="Body" required="true">
            <mx:TextArea id="messageBody" width="300" height="300"
wordWrap="true" />
        </mx:FormItem>
    </mx:Form>

    <mx:ControlBar horizontalAlign="right">
        <mx:Button label="Cancel" click="this.deletePopUp();" />
        <mx:Button label="Submit"
click="_root.addMessage(messageSubject.text,
messageBody.text);this.deletePopUp();" />
    </mx:ControlBar>
</mx:TitleWindow>

```

Listing 7: Use capital letters to refer to the column names

```

<mx:DataGrid id="threadGrid" width="100%" height="100%"
dataProvider="{forumWS.getThreads.result}"
change="threadSelected(threadGrid.selectedItem.ID,
threadGrid.selectedItem.NAME)">
    <mx:columns>
        <mx:Array>
            <mx:DataGridColumn columnName="NAME" headerText="Thread" />
            <mx:DataGridColumn columnName="USERNAME"
headerText="Originator" width="40" />
            <mx:DataGridColumn columnName="MESSAGECOUNT"
headerText="Replies" width="40" />
            <mx:DataGridColumn columnName="LASTPOST" headerText="Last
Post" width="70" />
        </mx:Array>
    </mx:columns>
</mx:DataGrid>

```

Download the Code...

Go to www.coldfusionjournal.com

Don't Miss CFDJ's Next Issue!



SPECIAL FOCUS ON PRESENTING DATA

Optimize, Extend and Enhance the Search Functionality in ColdFusion MX: If your end users can't find what they're looking for with your application's search, you're not realizing the full value of your ColdFusion investment

Reusable (and Maintainable) Presentation Code: Here are a few techniques that are helpful in writing reusable code for applications

"Traditional" IDEs: A look at Plum, a great product for the ColdFusion development community

The Alagad Image Component: A case study shows how this single CFC is used to create and manipulate image files directly from ColdFusion

Blackstone at MAX

The future of ColdFusion

MAX 2004 easily ranked as one of our best conferences ever. Between product announcements, the sessions, increased attendance, and the New Orleans scene, it set a new standard for us to beat next year.

For the benefit of those of you who did not attend (or who were too hung-over to pay attention), what follows is a recap of some of the MAX 2004 highlights from a very Blackstone perspective. (Note: I stated this at MAX and must do so again here. ColdFusion "Blackstone" is in beta, and is not yet a finished shipping product. As such, the features and technologies discussed in this column are subject to change. In other words, I want no flame-wars if stuff does indeed change.)

Blackstone Powers MAX SMS

Blackstone played a very important role at MAX this year, especially a brand new Blackstone feature, the SMS gateway. I'll be covering Blackstone and SMS in detail in a future column, but here are the basics. SMS stands for Short Messaging Service, the service used to send text messages between phones and devices. The Blackstone SMS gateway allows ColdFusion to send SMS messages and to respond to inbound SMS messages. Blackstone's SMS support was used in two ways at MAX:

- MAX session schedules were available electronically, allowing attendees to check their schedules as needed. This year the session scheduling application allowed attendees to sign up for SMS notification by providing their phone number (SMS ID). Fifteen minutes before each session, Blackstone sent SMS reminders to attendees letting them know where they needed to be next.
- As MAX coincided with the U.S. presidential elections, Steven Elop introduced an application allowing MAX attendees to vote via SMS. During his keynote segment he provided an SMS ID, and asked attendees to text the letter B, K, or N (for Bush, Kerry, and Nader respectively) to place a vote. Blackstone received those votes via SMS, and then passed the data to a Flex application that charted the stats in real time. (And no, the MAX vote did not match the actual election results; at MAX, Senator John Kerry won by a significant margin). About a third of the 2000 present at the keynote voted via SMS, an impressive display of SMS use and of Blackstone's planned SMS support.



By Ben Forta

The Blackstone Keynote Segment

Last year at MAX 2003 I announced Blackstone for the very first time, and presented a single slide listing the themes for that release. Since that time we have been very busy turning those themes into actual features and technologies. Here are the stats I presented:

- 15 months in development
- Over 20,000 development hours
- Over 2,000 beta testers
- 7 months of customer testing
- Over 15,000 regression tests

This year, with Blackstone now in public beta, Tim Buntel and I used the opening keynote to demonstrate some of what the ColdFusion engineering team has been working on.

Tim started by incrementally building a Flash-based electronic voting application, using just CFML tags. He started with this code:

```
<cfform format="flash"
      action="process.cfm"
      width="554"
      height="500">
</cfform>
```

This code created an empty Flash SWF in the ballot page using the existing <cfform> tag with format="flash". He then added a login form:

```
<cfformgroup type="panel"
            width="300"
            height="200">
  <cfinput type="text"
          name="login_id"
          value="" label="Voter Registration ID:"
          mask="A99-999-9999"
          required="yes">
  <cfinput type="password"
          name="login_password"
          value=""
          label="Voter Registration Password:"
          required="yes">
  <cfinput type="button"
          name="login_sbmt"
          value="Login and Vote!">
</cfformgroup>
```

The form is enclosed within a <cfformgroup> which

defines the group type as “panel”, making the login form appear in a pop-up-type panel. Notice also that the login_id field uses a character mask to enforce correct data entry.

Tim then wrapped this page into a tab navigator control using this code:

```
<cfformgroup type="tabnavigator"
             height="450"

style="backgroundImage: '#imageUrl#';">
...
</cfformgroup>
```

This placed the login screen on the first page of the tab navigator, and then additional pages were added for president selection (using buttons containing pictures of the candidates), ballot initiatives, and a final confirmation page.

When finished, Tim had created an impressive Flash forms application using only CFML tags, demonstrating how Blackstone will help developers build richer, more engaging data-entry forms, leveraging Flash without needing anything more than Blackstone itself.

Then I took over. Electronic voting critics demand a printed paper trail, and printing Web pages is not trivial. I showed how Microsoft Internet Explorer often cannot print Web pages properly, even simple pages (truncating data, losing data, and worse).

To create a printable version of Tim's ballot I simply wrapped his code using the following tags:

```
<cfdocument type="flashpaper">
...
</cfdocument>
```

I then voted using Tim's form, and when I was done, Blackstone generated a perfect printable FlashPaper ballot. I also created one other printable page, taking that same Web page that Internet Explorer could not print, and wrapping it in the following code to generate a perfect printable PDF version of the page:

```
<cfdocument type="pdf">
<!-- Get page using CFHTTP -->
<cfhttp ...>
</cfdocument>
```

The next thing we needed was a way to run a report on all votes to determine who won the election. To do this I used

the ColdFusion Report Builder, a tool that creates report templates (.cfr files) that are then processed by the ColdFusion <CFREPORT> tag. Using the Report Builder I did the following:

- Ran the report wizard
- Used the SQL Query Builder to select data from three tables, using joins and an aggregate function to calculate totals
- Grouped data by candidate and state
- Selected formatting and presentation options
- Embedded a pie chart showing voter breakdown by candidate

I also pointed out an important distinction between the ColdFusion Report Builder and third-party reporting tools in that the ColdFusion Report Builder is incredibly ColdFusion aware:

- CFML expressions (using CFML functions, variables, and more) could be embedded in reports
- Data need not be embedded in a report, and any query could be passed to the <CFREPORT> tag at runtime

Like <CFDOCUMENT>, <CFREPORT> generated both PDF and FlashPaper output.

The last demo in the Blackstone segment showed how a ColdFusion-powered SMS application could be built. Tim fired up the SMS server and gateway (included with Blackstone) and ran the SMS client (a phone emulator). He then wrote a few lines of code in a CFC method allowing him to send a request from the SMS client to obtain voting tallies.

We wrapped up by telling the attendees that every one of them had a copy of Blackstone Beta 2 in their bags.

Meet the ColdFusion Team BOF

BOF stands for “Birds of a Feather,” informal chats or panel discussions. This year I once again moderated the “Meet the ColdFusion Team” BOF, and a standing-room only crowd used the opportunity to interrogate the ColdFusion engineering, QA, business, and marketing teams. Here are some of the questions and answers:


- **What will the Blackstone upgrade cost?** And how will features be packaged and editioned? Sorry, we can't discuss that yet.
- **Will subscription customers get the Blackstone update?** Yes!
- **How can CFML code be executed**

asynchronously? Via the new CFML gateway in Blackstone.

- **Will ColdFusion support null?** JavaCast() in Blackstone makes it possible to pass null to Java calls, but support for null throughout CFML is on the wish list, but not implemented at this time.
- **What application events are supported by the new application event model?** OnApplicationStart, OnApplicationEnd, OnSessionStart, OnSessionEnd, OnRequestStart, OnRequestEnd, OnRequest OnError.
- **Will the new spell checking in Verity support custom dictionaries?** That is not necessary, Verity (in Blackstone) performs heuristic matching against text in indexed data, and does not actually use a dictionary at all.
- **Can the Blackstone beta coexist with CF5 or CFMX?** Yes, absolutely, we designed Blackstone so that it can live alongside current versions, so there is no reason not to try the beta.
- **What must we do to use the Blackstone beta CD provided to all MAX attendees?** Install it, agree to the terms, and provide us with feedback.

There were lots more questions too, but unfortunately I did not take notes.

Summary

MAX 2004 was a blast, and if you did not attend, don't make that same mistake next year. Blackstone played a prominent role in this year's conference, both as a product and as the power behind the scenes. I can best sum it up by sharing the following statement extracted from an e-mail I received from a very happy (and tired) attendee: “I no longer have any doubt about the future of ColdFusion, and of Macromedia's commitment to the product. My only complaint is that I want Blackstone now!” 

About the Author

Ben Forta is Macromedia's senior product evangelist and the author of numerous books, including ColdFusion MX Web Application Construction Kit and its sequel, Advanced ColdFusion MX Application Development, and is the series editor for the new “Reality ColdFusion” series. For more information visit www.forta.com. ben@forta.com

Procedures, Wizards, and Dangerous Things: What Categories Reveal About the Mind

Designing and building
stored procedures
and transforming
those procedures into
ColdFusion

Stored procedures are dangerous things. They are programs. They are a collection of pre-compiled SQL statements stored in the database. This article presents the techniques I used with the Switch-box CFSQLTool (free download available at www.switch-box.org) to remove the grief of designing and building stored procedures and then transforming those procedures into ColdFusion.

If you have the feeling stored procedures are dangerous, take a look at CFC in Listing 4. You may get the idea your feelings are correct. But wait. All that code was generated from CFSQLTool, and it's not as dangerous as it seems.

The code in Listing 4 is typical of the amount of programming required to build a simple data access layer (DAL) that bridges ColdFusion applications with the database. The code shown in Listing 4 is the goal of this article. We will look at the approach CFSQLTool's code generation wizards use by reverse engineering the SQL (all SQL examples and references use Microsoft SQL Server 2000) metadata once for building the stored procedures, and once again for generating CFC functions based on the stored procedures.

The Database Design

Usually business rules become entity-relationship models in



By Joseph Flanigan

the database. For example, the business rule for a user is expressed as *each user must have a unique identifier, user name, password, and date created. A user will be of a user type and will have a status. Every user shall be a contact. A user may have a reminder phrase for the password. Other rules might express more constraints such as each user name is unique, passwords have to be at least a certain length, and new users always have an active status.*

Figure 1 shows an example of an entity relationship diagram that models business rules for the user and the user relationships with Contacts, Status, and UserType. Listing 1 shows the User table data definition.

In the diagram, the names in bold underscore are primary keys. Transforming the entity relationship diagram into SQL data definition, the primary key (PK) [L1 15] is an integer that auto-increments its value as a new record is inserted in the table. This key is the unique identifier referred to in the business rule.

Names in bold require values and cannot be null. Names in normal print are not required values and can be null. Extending from the User table are three lines with arrows that diagram the "will" and the "shall" statements in the business rule. These lines illustrate the foreign key relationship. A foreign key (FK) [L1 22, 28, 34] is a reference to a fact about the user that does not exist in the User table itself. The "foreign" part to the term indicates not being in the User table, and the "key" part is the pointer value that indicates the value of the primary key in the foreign table.

If you notice in the diagram, the names next to the FK labels are in bold, which means the foreign key value is required. In

some other business rules, the foreign key relationship may be expressed, but it may not be required. For example, the rule could be, "The user may have a status." In this case, the FK label would be there, but the name would be in normal print.

DateCreate [14] is required, but it has a default value, get-date(), that the DB (database) inserts when a new user is added. This means the application inserting a new user does require passing in the DataCreate value.

The SQL Insert Procedure

Create, Read, Update, and Delete (CRUD) are the basic programming operations on tables. Create means inserting new data into a table. Read means selecting data from the table.

Stored procedures work especially well with CRUD operations that change the content of data in the table. If you are planning a database design, consider using stored procedures for inserts, updates, and deletes. Listing 2 is an example of a stored procedure, pr_tUser_INS, for inserting a new record in the user table. This is a stored procedure that would implement inserting a new user following requirements of the previous business rules.

The first part of the procedure [L2 3-7] declares the name and space required for variables the procedure will use. Notice these names are the names found in the business rule: UserName, Password, ContactID, StatusID, and UserID. The data for these variables must be passed into the procedure to do the insert. Datecreate is also required for the business rule but not coded as a variable. This is because in the table data definition, the DateCreate field has a default value.

MS SQL Server has a feature called identity property for columns that can be used for primary key values [L1 2]. This feature makes it very easy to add a new user and get the primary key. As part of the stored procedure, this primary key value can be returned to the calling application [L2 29].

The CF StoredProc Function

Once the stored procedure is in the database, ColdFusion has tags, <cfstoredproc>, that make calling the procedure easy, but the coding can be dangerous. In fact, it causes a lot of grief to code. The first wizard I wrote for CFSQLTool was just to handle insert procedures and the <cfprocparm> tags. CFMX

removed some of the grief because the cfstoredproc could be put in a cffunction and values could be passed in by reference rather than coded as values for each cfprocparm.

Listing 3 shows the cffunction that has the cfprocparm for each of the variables in the procedure pr_tUser_INS. The cfprocparm tag has several attributes that must be coded correctly to bind the CF with the stored procedure. It is very easy to mistype something when a procedure has so many values to pass in. Insert and update procedures can really take lots of time to hand code. All that metadata about tables is just sitting in the database waiting for wizards to do the coding.

For a CRUD application update, select and delete functions are necessary. Most likely several different functions and procedures will be needed for selects and updates. For a data access layer, all these functions can be constructed in a CFC referred to as the wrapper CFC.

The Switchpanel Wizard

When I first started CFSQLTool, I wanted a new feature to make the coding a lot faster and easier. One of the ideas was to think of a CRUD as being a suite of procedures rather than just one of each, like most SQL tools support. I wanted to be able to write four procedures at a time where I could see the interaction at design time. I wanted to see what column needed values at insert time and what values complemented the insert with updates. Also, I wanted many of the functions of Enterprise Manager but within a CF environment. My thinking was that if the application would be in CF, then build the SQL in CF.

The Switchpanel Wizard Interface (see Figure 2), is the basis of CFSQLTool for program design. It's a spreadsheet of switches and metadata about tables.

Across the top [1] are the check boxes for CRUD procedures. At design time, you may want to build alternate procedures. The wizard prenames the CRUD stored procedure [5] using a text input form. To build multiple procedures, just uncheck the unwanted CRUD and change the proposed procedure name.

The column metadata [3] provides essential design information. Right there as part of the spreadsheet are the facts about a column. There is no need to look in multiple places to make sure a column is not null and has a default value. If the

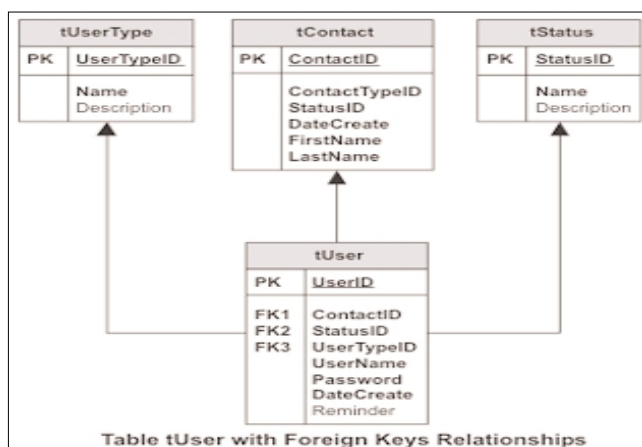


Figure 1: tUser entity relationship diagram

Figure 2: Switchpanel stored procedure design interface

table being programmed has an identity column as the primary key, the column is highlighted with yellow across the switchpanel. Foreign and primary key labels make it easier to decide the implications of procedure actions.

Each CRUD has several check box or pull-down options [2]. Create, Read, and Update have result set (RS) options. Checking an RS box will cause the code generator to build the procedure that returns the checked column. The Read procedure switchboxes [4] provide basic formats for sorting and clause generation. Select SQL can be very complicated, and CFSQLTool basic features do not replace advanced query building tools. But if the advanced query is defined as view, CFSQLTool can provide the wrapper code based on the view.

If the table uses identity columns as the primary key and if the design checks the identity column as a result set, the code generation will add a select statement to the insert transaction to return the new identity value [L2 29].

Meta Magic

The insert procedure can be one of the more difficult procedures to get correct by hand coding. CFSQLTool does something I have not found in other tools. It does insert procedure proposals. The tool looks at the metadata for the table and, using a few simple rules, predicts the check boxes for an insert procedure.

In Figure 2, when the switchpanel opened, the boxes checked in the Read Insert columns were prechecked by the insert predictor. The tool does proposed check boxes for updates, selects, and deletes as well.


The metadata predictors do a pretty good job for this ver-

sion of the tool. As the tool matures, I expect the predictors will improve to support UUID and complex primary keys.

Pop Goes the CFC

Getting back to Listing 4, we have built the stored procedures and put them in the database. Now we need the CFC to run the procedures. The user procedure wizard does this function. It's so easy we don't need an illustration. First select the wizard, then select the list of procedures, and hit generate. All done. Copy and paste into Studio.

Wizards Are Dangerous


One thing I really like about CFSQLTool is that the wizards are so fast. I found that the data access layer wrapper CFC can expand with new functions quickly. Before CFSQLTool, much of the development time effort was spent writing queries and the supporting CF code. Often some minor database function was bypassed because of development time. CFSQLTool changed that situation by changing the coding problem into a design feature. It put building the data access layer code into a new category by replacing keystrokes with key clicks. It changed hand coding one procedure at a time by designing four procedures at once and then code generating the design. 

About the Author

Joseph Flanigan's computer career spans 35 years. In 1995, he switched from Perl to ColdFusion for Web applications. By late 1999, his first ideas for the Switchbox framework started and were published as Switch-box.org in August 2000. Joseph is a member of the IEEE and ACM.

joseph@switch-box.org

Terms

- **Stored procedure:** A precompiled collection of Transact-SQL statements stored under a name and processed as a unit.
- **Foreign keys:** The column or combination of columns in which values match the primary key (PK) or unique key in the same or another table.
- **Parameters:** The list of arguments being passed into a function.
- **Attributes:** The list of arguments being received by a function.
- **Pass by reference:** The variable memory location address of the value of a parameter.
- **Pass by name:** The variable name of the memory address of the value of a parameter.
- **Pass by value:** The actual value of a variable.
- **Namespace:** A set of unique names in which each name is an alphanumeric term consisting of both letters and numbers and often other symbols. This identifies a data set, statement, program, or cataloged procedure. In programs, the identity of a variable, array, function, subroutine, common block, or namelist. Besides a program environment, the notion of namespace occurs in other abstract universes like the domain name system, XML elements and entities, and dictionaries. 

Listing 1: Create User, Keys and Constraints

```
1 CREATE TABLE [dbo].[tUser] (  
2     [UserID] [int] IDENTITY (1, 1) NOT NULL ,  
3     [ContactID] [int] NOT NULL ,  
4     [StatusID] [smallint] NOT NULL ,  
5     [UserID] [smallint] NOT NULL ,  
6     [UserName] [varchar] (36) NOT NULL ,  
7     [Password] [varchar] (36) NOT NULL ,  
8     [DateCreate] [datetime] NOT NULL ,  
9     [Reminder] [varchar] (36) NULL  
10 ) ON [PRIMARY]  
11 GO  
12  
13 ALTER TABLE [dbo].[tUser] ADD  
14     CONSTRAINT [DF_tUser_DateCreate] DEFAULT (getdate())  
15 FOR [DateCreate],  
16     CONSTRAINT [PK_tUser] PRIMARY KEY CLUSTERED  
17     (  
18         [UserID]  
19     ) ON [PRIMARY]  
20 GO  
21 ALTER TABLE [dbo].[tUser] ADD  
22     CONSTRAINT [FK_tUser_tContact] FOREIGN KEY
```

```

23      (
24      [ContactID]
25      ) REFERENCES [dbo].[tContact] (
26      [ContactID]
27      ),
28      CONSTRAINT [FK_tUser_tStatus] FOREIGN KEY
29      (
30      [StatusID]
31      ) REFERENCES [dbo].[tStatus] (
32      [StatusID]
33      ),
34      CONSTRAINT [FK_tUser_tUserType] FOREIGN KEY
35      (
36      [UserTypeID]
37      ) REFERENCES [dbo].[tUserType] (
38      [UserTypeID]
39      )
40  GO

```

Transact-SQL Reference CREATE TABLE

Listing 2: Stored Procedure Insert with Scope_Identity

```

1  CREATE PROCEDURE pr_tUser_INS
2  (
3      @ContactID          [int],
4      @StatusID           [smallint],
5      @UserTypeID         [smallint],
6      @UserName           [varchar] (36),
7      @Password           [varchar] (36)
8  )
9  AS
10 BEGIN

```

```

11 BEGIN TRAN
12     INSERT INTO  dbo.tUser
13     (
14     [ContactID],
15     [StatusID],
16     [UserTypeID],
17     [UserName],
18     [Password]
19     )
20     VALUES
21     (
22     @ContactID,
23     @StatusID,
24     @UserTypeID,
25     @UserName,
26     @Password
27     )
28
29     SELECT  [UserID], [DateCreate]
30     FROM    dbo.tUser
31     WHERE   UserID = SCOPE_IDENTITY()
32
33 COMMIT TRAN
34 END

```

Transact-SQL Reference SCOPE_IDENTITY()

Listing 3: CFFunction to call tUser insert procedure

```

1  <cffunction name="pr_tUser_INS" returntype="any" display-
name="pr_tUser_INS">
2      <cfargument name="theParams" type="struct" required="yes"
>
3      <cfargument name="DSN" type="string"

```

Be the 1st to develop with
Blackstone

HostMySite.com will be the first host to offer
the innovative new version of ColdFusion.

Be the first, visit hostmysite.com/blackstone

 **HostMySite.com**

1-877-215-4678

stored procedures

```

default="#this.DSN#"
4      <cfset var RS1= "" >
5
6      <cfstoredproc procedure="pr_tUser_INS"
datasource="#Arguments.DSN#" returncode="YES">
7          <cfprocparam type="In"
8              cfsqltype="CF_SQL_INTEGER"
9              dbvarname="@ContactID"
10             value="#theParams.$ContactID#"
11             maxlength="4"
12             null="No">
13      <cfprocparam type="In"
14          cfsqltype="CF_SQL_SMALLINT"
15          dbvarname="@StatusID"
16          value="#theParams.$StatusID#"
17          maxlength="2"
18          null="No">
19      <cfprocparam type="In"
20          cfsqltype="CF_SQL_SMALLINT"
21          dbvarname="@UserID"
22          value="#theParams.$UserID#"
23          maxlength="2"
24          null="No">
25      <cfprocparam type="In"
26          cfsqltype="CF_SQL_VARCHAR"
27          dbvarname="@UserName"
28          value="#theParams.$UserName#"
29          maxlength="36"
30          null="No">
31      <cfprocparam type="In"
32          cfsqltype="CF_SQL_VARCHAR"
33          dbvarname="@Password"
34          value="#theParams.$Password#"
35          maxlength="36"
36          null="No">
37
38      <cfprocresult name="RS1">
39      </cfstoredproc>
40      <cfreturn RS1 > <!-- END Procedure: pr_tUser_INS --->
41      </cffunction> <!-- END Function: pr_tUser_INS --->

```

See CFML LRM for cffunction and cfstoredproc description

Listing 4: CFC to call tUser stored procedures

```

1
2      <cfcomponent displayname="tUser_proc">
3
4
5      <!---
6          DSN: CFDJ
7          Function Count: 4
8          Function: pr_tUser_DEL
9          Function: pr_tUser_INS
10         Function: pr_tUser_UPD
11         Function: pr_tUserByUserID_SEL
12     --->
13
14     <!---
***** ----->
15     <cffunction name="init" returntype="struct" hint="Init for the
CFC without arguments" output="no">
16         <cfreturn this>
17     </cffunction>
18
19     <!---
***** ----->
20
21     <!---

```

```

***** ----->
22     <cffunction name="SetInit" returntype="boolean" hint="SetInit
function with optional DSN argument" output="no">
23         <cfargument name="DSN" type="string" default="">
24         <cfset this.DSN = Arguments.DSN >
25         <cfreturn TRUE>
26     </cffunction>
27
28     <!---
***** ----->
29     <!--- ::
30         Function: pr_tUser_DEL
31         Gen Date: Tuesday, November 16, 2004
32         DSN: CFDJ
33         Procedure: pr_tUser_DEL
34     ::--->
35     <cffunction name="pr_tUser_DEL" returntype="any" display-
name="pr_tUser_DEL">
36
37         <cfargument name="theParams" type="struct" required="yes"
>
38         <cfargument name="DSN" type="string"
default="#this.DSN#">
39         <cfset var RS1= "" >
40
41         <!---
42         Namespaces for the stored procedure pr_tUser_DEL.
43
44         <cfscript>
45             /* Data Structure for pass-by-reference into function
pr_tUser_DEL */
46             pr_tUser_DEL = StructNew();
47
48             /* Set initial values of each key to Request.IN value */
49             pr_tUser_DEL.$UserID = Request.IN.UserID;
50         </cfscript>
51
52
53         <cfscript>
54             /* Call the stored procedure pr_tUser_DEL. Prefix with
CFC object name. */
55             CFCname.pr_tUser_DEL(pr_tUser_DEL);
56         </cfscript>
57
58         SQL Stored Procedure:
59         CREATE PROCEDURE pr_tUser_DEL
60         (
61             @UserID          [int]
62         )
63         AS
64         BEGIN
65             BEGIN TRAN
66                 DELETE
67                 FROM   dbo.tUser
68                 WHERE  [UserID] = @UserID
69
70             IF (@@error!=0)
71                 BEGIN
72                     RAISERROR 20000 'pr_tUser_DEL: Cannot Delete
data from dbo.tUser'
73                     ROLLBACK TRAN
74                     RETURN(1)
75                 END
76
77             COMMIT TRAN
78             RETURN(0)
79         END
80

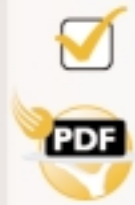
```

```

81    --->
82
83    <cftry>
84    <cfstoredproc procedure="pr_tUser_DEL"
datasource="#Arguments.DSN#" returncode="YES">
85        <cfprocparam type="IN" cfsqltype="CF_SQL_INTEGER"
dbvarname="@UserID" maxlength="4" value="#theParams.UserID#" >
86
87        <cfprocresult name="RS1">
88    </cfstoredproc>
89    <cfset Request.pr_tUser_DEL =
cfstoredproc.statusCode >
90
91    <cfcatch type="ANY">
92        <cfthrow extendedinfo="#cfcatch#" message="Database
Error with procedure pr_tUser_DEL" >
93    </cfcatch>
94    </cftry>
95
96    <cfreturn RS1 > <!--- END Procedure: pr_tUser_DEL --->
97    </cffunction> <!--- END Funtion: pr_tUser_DEL --->
98
99    <!--- ::
100        Function: pr_tUser_INS
101        Gen Date: Tuesday, November 16, 2004
102        DSN: CFDJ
103        Procedure: pr_tUser_INS
104    :!--->
105    <cffunction name="pr_tUser_INS" returntype="any" dis-
playname="pr_tUser_INS">
106
107        <cfargument name="theParams" type="struct"
required="yes" >
108        <cfargument name="DSN" type="string"
default="#this.DSN#">
109        <cfset var RS1= "" >
110
111    <!---
112    Namespaces for the stored procedure pr_tUser_INS.
113
114    <cfscript>
115        /* Data Structure for pass-by-reference into function
pr_tUser_INS */
116        pr_tUser_INS = StructNew();
117
118        /* Set initial values of each key to Request.IN value
*/
119        pr_tUser_INS.$ContactID = Request.IN.ContactID;
120        pr_tUser_INS.$StatusID = Request.IN.StatusID;
121        pr_tUser_INS.$UserID = Request.IN.UserID;
122        pr_tUser_INS.$UserName = Request.IN.UserName;
123        pr_tUser_INS.$Password = Request.IN.Password;
124    </cfscript>
125
126
127    <cfscript>
128        /* Call the stored procedure pr_tUser_INS. Prefix
with CFC object name. */
129        CFName.pr_tUser_INS(pr_tUser_INS);
130    </cfscript>
131
132    SQL Stored Procedure:
133    CREATE PROCEDURE pr_tUser_INS
134    (
135        @ContactID        [int],
136        @StatusID         [smallint],
137        @UserID           [smallint],
138        @UserName         [varchar] (36),
139        @Password         [varchar] (36)

```

What's your PDF?



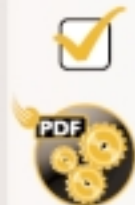
Precise Document Formatting

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.



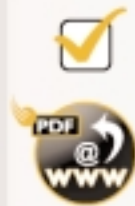
Populate Dynamic Forms

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, print, 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.

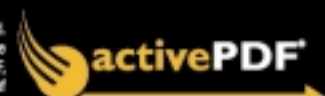


Present Data Fashionably

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 Paper Revolution" and the
activePDF logo are registered trademarks of activePDF,
Inc. All activePDF product names are trademarks of
activePDF, Inc.



stored procedures

```

140     )
141     AS
142     BEGIN
143     BEGIN TRAN
144         INSERT INTO  dbo.tUser
145         (
146             [ContactID],
147             [StatusID],
148             [UserID],
149             [UserName],
150             [Password]
151         )
152     VALUES
153     (
154         @ContactID,
155         @StatusID,
156         @UserID,
157         @UserName,
158         @Password
159     )
160
161     IF (@@error!=0)
162     BEGIN
163         RAISERROR 20000 'pr_tUser_INS: Cannot Insert
data from dbo.tUser'
164         ROLLBACK TRAN
165         RETURN(1)
166     END
167
168     SELECT [UserID], [DateCreate]
169     FROM dbo.tUser
170     WHERE UserID = SCOPE_IDENTITY()
171
172     IF (@@error!=0)
173     BEGIN
174         RAISERROR 20000 'pr_tUser_INS: Cannot Select
data from dbo.tUser'
175         ROLLBACK TRAN
176         RETURN(1)
177     END
178
179     COMMIT TRAN
180     RETURN(0)
181     END
182
183     --->
184
185     <cftry>
186     <cfstoredproc procedure="pr_tUser_INS"
datasource="#Arguments.DSN#" returncode="YES">
187         <cfprocparam type="IN" cfsqltype="CF_SQL_INTEGER" dbvar-
name="@ContactID" maxlength="4" value="#theParams.$ContactID#" >
188         <cfprocparam type="IN" cfsqltype="CF_SQL_smallint" dbvar-
name="@StatusID" maxlength="2" value="#theParams.$StatusID#" >
189         <cfprocparam type="IN" cfsqltype="CF_SQL_smallint" dbvar-
name="@UserID" maxlength="2" value="#theParams.$UserID#" >
190         <cfprocparam type="IN" cfsqltype="CF_SQL_varchar" dbvar-
name="@UserName" maxlength="36" value="#theParams.$UserName#" >
191         <cfprocparam type="IN" cfsqltype="CF_SQL_varchar" dbvar-
name="@Password" maxlength="36" value="#theParams.$Password#" >
192
193         <cfprocresult name="RS1">
194     </cfstoredproc>
195     <cfset Request.pr_tUser_INS = cfstoredproc.statusCode >
196
197     <cfcatch type="ANY">
198         <cfthrow extendedinfo="#cfcatch#" message="Database Error
with procedure pr_tUser_INS" >
199     </cfcatch>

```

```

200     </cftry>
201
202     <cfreturn RS1 > <!-- END Procedure: pr_tUser_INS --->
203     </cffunction> <!-- END Funtion: pr_tUser_INS --->
204
205     <!-- ::
206         Function: pr_tUser_UPD
207         Gen Date: Tuesday, November 16, 2004
208         DSN: CFDJ
209         Procedure: pr_tUser_UPD
210     :--->
211     <cffunction name="pr_tUser_UPD" returntype="any"
displayname="pr_tUser_UPD">
212
213         <cfargument name="theParams" type="struct" required="yes"
>
214         <cfargument name="DSN" type="string"
default="#this.DSN#">
215         <cfset var RS1= "" >
216
217     <!--
218         Namespaces for the stored procedure pr_tUser_UPD.
219
220     <cfscript>
221         /* Data Structure for pass-by-reference into function
pr_tUser_UPD */
222         pr_tUser_UPD = StructNew();
223
224         /* Set initial values of each key to Request.IN value */
225         pr_tUser_UPD.$UserID = Request.IN.UserID;
226         pr_tUser_UPD.$ContactID = Request.IN.ContactID;
227         pr_tUser_UPD.$StatusID = Request.IN.StatusID;
228         pr_tUser_UPD.$UserID = Request.IN.UserID;
229         pr_tUser_UPD.$UserName = Request.IN.UserName;
230         pr_tUser_UPD.$Password = Request.IN.Password;
231         pr_tUser_UPD.$Reminder = Request.IN.Reminder;
232     </cfscript>
233
234
235     <cfscript>
236         /* Call the stored procedure pr_tUser_UPD. Prefix with
CFC object name. */
237         CFCname.pr_tUser_UPD(pr_tUser_UPD);
238     </cfscript>
239
240     SQL Stored Procedure:
241     CREATE PROCEDURE pr_tUser_UPD
242     (
243         @UserID          [int],
244         @ContactID       [int],
245         @StatusID        [smallint],
246         @UserID          [smallint],
247         @UserName        [varchar] (36),
248         @Password        [varchar] (36),
249         @Reminder        [varchar] (36)
250     )
251     AS
252     BEGIN
253     BEGIN TRAN
254         UPDATE  dbo.tUser
255     SET
256         [ContactID]          = @ContactID,
257         [StatusID]          = @StatusID,
258         [UserID]            = @UserID,
259         [UserName]          = @UserName,
260         [Password]          = @Password,
261         [Reminder]          = @Reminder
262
263     WHERE

```

```

264      (
265      UserID = @UserID
266      )
267
268
269      IF (@@error!=0)
270      BEGIN
271      RAISERROR 20000 'pr_tUser_UPD: Cannot Update
data from dbo.tUser'
272      ROLLBACK TRAN
273      RETURN(1)
274      END
275
276      COMMIT TRAN
277      RETURN(0)
278      END
279
280      --->
281
282      <cftry>
283      <cfstoredproc procedure="pr_tUser_UPD"
datasource="#Arguments.DSN#" returncode="YES">
284      <cfprocparam type="IN" cfsqltype="CF_SQL_INTEGER" dbvar-
name="@UserID" maxlength="4" value="#theParams.$UserID#" >
285      <cfprocparam type="IN" cfsqltype="CF_SQL_INTEGER" dbvar-
name="@ContactID" maxlength="4" value="#theParams.$ContactID#" >
286      <cfprocparam type="IN" cfsqltype="CF_SQL_smallint" dbvar-
name="@StatusID" maxlength="2" value="#theParams.$StatusID#" >
287      <cfprocparam type="IN" cfsqltype="CF_SQL_smallint" dbvar-
name="@UserID" maxlength="2" value="#theParams.$UserID#" >
288      <cfprocparam type="IN" cfsqltype="CF_SQL_varchar" dbvar-
name="@UserName" maxlength="36" value="#theParams.$UserName#" >
289      <cfprocparam type="IN" cfsqltype="CF_SQL_varchar" dbvar-
name="@Password" maxlength="36" value="#theParams.$Password#" >
290      <cfprocparam type="IN" cfsqltype="CF_SQL_varchar" dbvar-
name="@Reminder" maxlength="36" value="#theParams.$Reminder#" >
291
292      <cfprocresult name="RS1">
293      </cfstoredproc>
294      <cfset Request.pr_tUser_UPD = cfstoredproc.statusCode >
295
296      <cfcatch type="ANY">
297      <cfthrow extendedinfo="#cfcatch#" message="Database Error
with procedure pr_tUser_UPD" >
298      </cfcatch>
299      </cftry>
300
301      <cfreturn RS1 > <!-- END Procedure: pr_tUser_UPD --->
302      </cffunction> <!-- END Funtion: pr_tUser_UPD --->
303
304      <!-- ::
305      Function: pr_tUserByUserID_SEL
306      Gen Date: Tuesday, November 16, 2004
307      DSN: CFDJ
308      Procedure: pr_tUserByUserID_SEL
309      ::-->
310      <cffunction name="pr_tUserByUserID_SEL" returntype="any" dis-
playname="pr_tUserByUserID_SEL">
311
312      <cfargument name="theParams" type="struct" required="yes"
>
313      <cfargument name="DSN" type="string"
default="#this.DSN#">
314      <cfset var RS1= "" >
315
316      <!--
317      Namespaces for the stored procedure pr_tUserByUserID_SEL.
318
319      <cfscript>

```

```

320      /* Data Structure for pass-by-reference into function
pr_tUserByUserID_SEL */
321      pr_tUserByUserID_SEL = StructNew();
322
323      /* Set initial values of each key to Request.IN value */
324      pr_tUserByUserID_SEL.$UserID = Request.IN.UserID;
325      </cfscript>
326
327
328      <cfscript>
329      /* Call the stored procedure pr_tUserByUserID_SEL. Prefix
with CFC object name. */
330      CFCname.pr_tUserByUserID_SEL(pr_tUserByUserID_SEL);
331      </cfscript>
332
333      SQL Stored Procedure:
334      CREATE PROCEDURE pr_tUserByUserID_SEL
335      (
336      @UserID          [int]
337      )
338      AS
339      BEGIN
340      BEGIN TRAN
341      SELECT TOP 100 PERCENT
[UserID],[ContactID],[StatusID],[UserID],[UserName],[Password],[Dat
eCreate],[Reminder]
342      FROM dbo.tUser
343      WHERE ( [UserID] = @UserID )
344
345      IF (@@error!=0)
346      BEGIN
347      RAISERROR 20000 'pr_tUserByUserID_SEL: Cannot
Select data from dbo.tUser'
348      ROLLBACK TRAN
349      RETURN(1)
350      END
351
352      COMMIT TRAN
353      RETURN(0)
354      END
355
356      --->
357
358      <cftry>
359      <cfstoredproc procedure="pr_tUserByUserID_SEL"
datasource="#Arguments.DSN#" returncode="YES">
360      <cfprocparam type="IN" cfsqltype="CF_SQL_INTEGER" dbvar-
name="@UserID" maxlength="4" value="#theParams.$UserID#" >
361
362      <cfprocresult name="RS1">
363      </cfstoredproc>
364      <cfset Request.pr_tUserByUserID_SEL =
cfstoredproc.statusCode >
365
366      <cfcatch type="ANY">
367      <cfthrow extendedinfo="#cfcatch#" message="Database Error
with procedure pr_tUserByUserID_SEL" >
368      </cfcatch>
369      </cftry>
370
371      <cfreturn RS1 > <!-- END Procedure: pr_tUserByUserID_SEL --->
372      </cffunction> <!-- END Funtion: pr_tUserByUserID_SEL --->
373      </cfcomponent> <!-- END component tUser_proc --->

```

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

ColdFusion

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.sacflug.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.jaxcfusion.org/

Florida
Winter Springs
Gainesville, FL CFUG
www.gistcfusion.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.hoosiercfusion.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.kcfusion.org

Missouri
O'Fallon
St. Louis, MO CFUG
www.stimmug.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

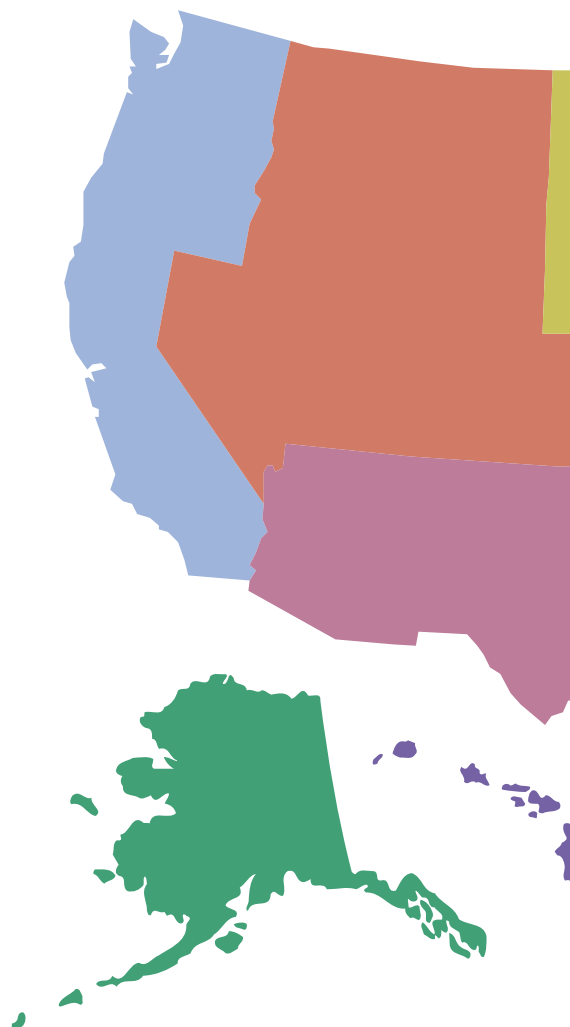
New York
Brooklyn
New York, NY CFUG
www.nycfug.org

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

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

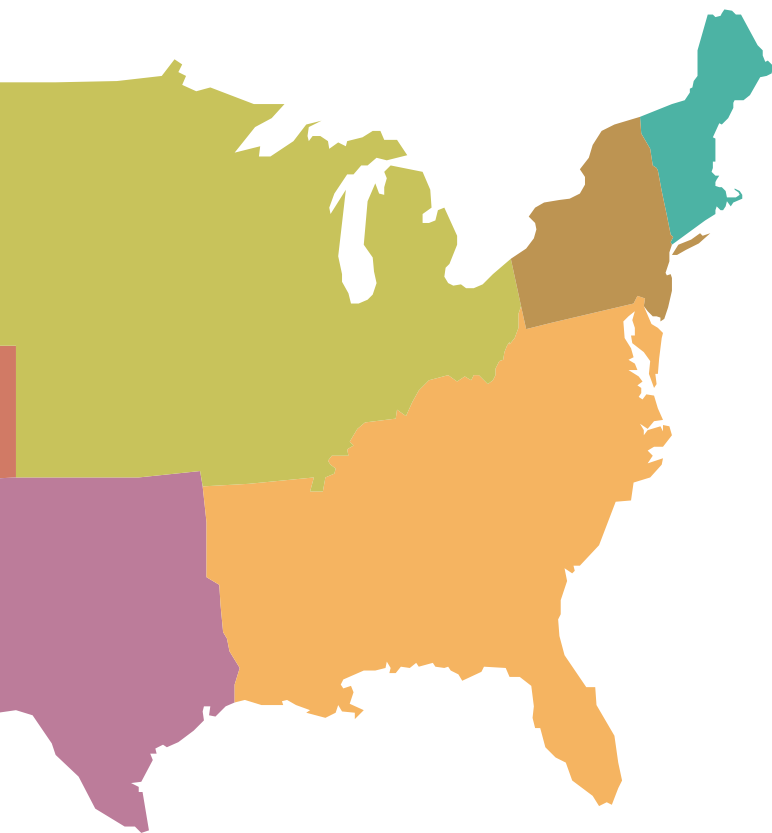
Ohio
Dayton
Greater Dayton CFUG
www.cfd Dayton.com

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/

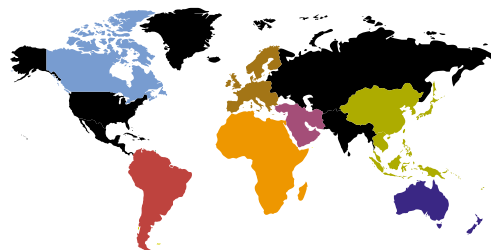
Texas
Houston
Houston Area CFUG
www.houcfug.org

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.cfusion.org.au/

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

Australia
Victoria CFUG
www.cfcentral.com.au

Australia
Western Australia CFUG
www.cfusionwa.com/

Brazil
Brasilia CFUG
www.cfusionbr.com.br

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

Brazil
Sao Paulo CFUG
www.cfusionsp.com.br

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfusiontoronto.org

Ireland
Dublin, Ireland CFUG
www.mmug-dublin.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

Spain
Spanish CFUG
www.cfusionspain.org

Switzerland
Swiss CFUG
www.swisscfug.org

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

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org



Creating a Component to Help You Collect Addresses

Squaring away address fields

Last month I introduced ColdFusion Components in this column. I wrote about the CFC file extension, the tags that make up components (cfcomponent and cffunction), how to create a component, and how to call methods on that component instance.

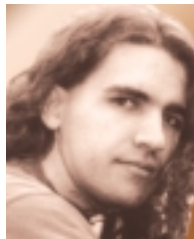
This month, I'll take you step-by-step through the process of creating a component that you can reuse in your own development. I'd be willing to bet that somewhere in your development adventures you've needed to collect an address from your users. Maybe you needed a billing address when you collected credit card information, or a shipping address when you sold a product through an e-commerce store. Perhaps you needed to collect a home address and a work address for the address book on your company's intranet. In this article, we'll create a standard component that you can use in all of these cases.

The Database Design

Before we can start writing code, we need to decide what type of data we want to store. A typical U.S.-based address will probably contain a street address, city, state, and zip. If you are dealing with international information, you may also want to store a country. Additional information might be a phone or fax number. You may want to store a name with your address information, although I usually have that information stored in a separate user table, so it's left out of this example (see Figure 1).

I added an AddressID field to the address table. This field is used as a primary key. When future tables or code reference an address they can do so using the primary key.

The street address is separated into two fields, Address1 and Address2. This might be used in cases where the user needs one line for a street and house number, and a second line for an apartment number. State data is relegated to a separate table and referenced using a foreign key, StateID. I also added some date fields including DateCreated, TimeCreated, DateModified, and TimeModified. I add these fields to most tables, and will use them for reporting purposes.



By Jeffery Houser

Creating the Component

The first thing that we need to do is to set the instance variables of our component. I will set some initialization code in the pseudoconstructor area of the component. You may remember from last month that constructor code inside a component, but not inside a cffunction tag, is often called pseudoconstructor code. This code is executed every time an instance of the component is created (see Listing 1).

Integer fields, such as IDs, are initialized to zero. Text fields are initialized to empty strings. I initialized the date and time fields to empty strings, although in some cases I have initialized them to the current day or time. The instance data is placed in the variables scope, which means it is private to the component. All the instance data is placed in a structure inside the variables scope, which I called instance. This is so you can easily dump (using the cfdump tag) all the instance variables without having to display all the extraneous component information that would be displayed had you just dumped the variables scope.

The instance data is used to represent the database fields in the address table. It also contains the state data from the state table, such as state and state abbreviation. This data will probably be used primarily for display purposes in our application; when we update the address table, we'll only need the ID field.

You'll need a way to access the data from outside the component, since it is internal to the component. You can provide this through the use of getter and setter methods. A getter method is one that will get the value of a component variable.

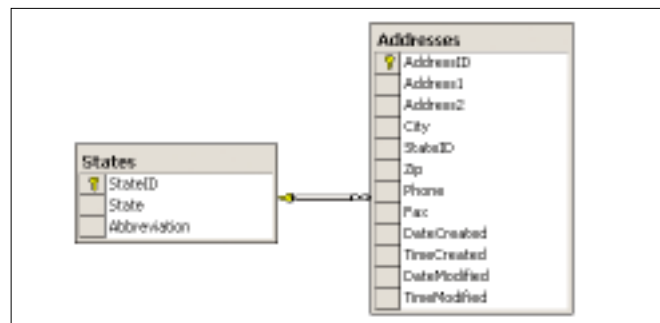


Figure 1: Address table

A setter method is one that will set the value of a component variable. Most of your getter and setter methods will be similar to each other, so I'll only give a simple example here. See Listing 2 for the get method for the address 1 variable

The name is `GetAddress1`, and the method is public. The return value is a string. The method body has a single line, which returns the value of the `address1` instance variable. The set method is shown in Listing 3.

The name of this method is `SetAddress1`. Its access is public. The return type is a Boolean value. The method will return true if the set is a success, and false otherwise. Since this simple method does not contain any additional error checking, it will always return true (I'm making the assumption that the `cfset` will not fail). You could institute more complex error checking and have the method return false on failure. As an example, you may want to do this if a zip code has too few or too many characters. A single string argument is accepted for this method. The body consists of a `cfset` and a return.

Most of the get and set methods are simple, like the `Address1` methods I just explained. One deviation is the `StateID` method. In reality, we could get by with just the `StateID` method, since that's the only field inside the address table. However, the state name and state abbreviation fields are included as a convenience to the developer. Listing 4 shows how the `SetStateID` method will retrieve this information.

This method accepts two arguments, the ID of the state and the name of the data source. It uses a query to retrieve the state data based on the ID passed into the function. Instead of just setting a single piece of data, it sets the state ID, the state name, and the state abbreviation. This method is an example of when more error checking might be beneficial. For instance, if nothing is returned from the query we could return false, or if the query failed for some reason we could return false.

Init and Commit Methods

A component's usefulness may be limited if it only has getter and setter methods. In our address component, we

are going to add two additional methods. The first is an init method. The init method can be used to load data from the database and populate the instance data. The second method we are going to add is a commit method. The commit method will either create or update a database with the instance data of the component. Listing 5 shows the init method.

It accepts two arguments, the `AddressID` and the DSN. The `AddressID` is the primary key of the address record in the database. The DSN is the data source that refers to the database. It creates the name of the query as a local function variable using the `var` keyword. Then the code queries the database to retrieve the address information. The instance variables are set based on the query data and the function returns a Boolean true value.

The commit method is designed to update the information in the database. The code can be seen in Listing 6.

The method accepts the data source name (DSN) as the only argument. It checks the `AddressID` instance variable to see if we need to update the database

Bridging the Collaboration Gap.

Collaboration

Combine your legacy collaboration applications into one easy-to-use, powerful interface.

FuseTalk for Microsoft .NET.

1-866-477-7542

Discussion Forums

www.fusetalk.com



Discussion forum solutions that make web-based collaboration risk-free and easy.

or insert a new record. If the AddressID is 0, then we are dealing with new data. If the AddressID is not zero, then we can update that record in the database. If we create the new record, the AddressID is set. An SQL Server-specific method is used to retrieve the most up-to-date ID.

Using the Component

Now that you've created the component, how do you use it? Let's suppose you are creating a user component. The user component contains information about your users including their name, access levels, a home address, and a work address. How would I use the address component to create the home and work address? First, I'd initialize them in the constructor code using something like this:

```
<cfoobject component="address" name="vari-
```

```
ables.instance.WorkAddress">
<cfoobject component="address" name="vari-
ables.instance.HomeAddress">
```


I would add get and set methods, similar to Listing 7. For the set method, the address component initialization would be handled outside of the user method, and the component would be passed into the method as a variable. The argument type is the name of the address component, address.

To manipulate the address components through the user component you can use code like this:

```
UserComponentInstance.GetWorkAddress().GetAdd-
ress1()
```

Conclusion

This article should help you with the steps of building modular code. Components allow you to create the

building blocks of an application. Over time, you'll develop a library of different components for different purposes, and then creating applications is just a matter of referencing your previously created jobs. As you start to develop a library of these components eventually your development will become quicker since you won't start every application from scratch. 

About the Author

Jeffrey 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 CF, most recently ColdFusion MX: The Complete Reference (McGraw-Hill Osborne Media).

jeff@instantcoldfusion.com

Listing 1

```
<cfscrip>
variables.instance.AddressID = 0;
variables.instance.Address1 = "";
variables.instance.Address2 = "";
variables.instance.City = "";
variables.instance.StateID = 0;
variables.instance.State = "";
variables.instance.StateAbbreviation = "";
variables.instance.Zip = "";
variables.instance.Phone = "";
variables.instance.Fax = "";
variables.instance.DateCreated = "";
variables.instance.TimeCreated = "";
variables.instance.DateModified = "";
variables.instance.TimeModified = "";
</cfscrip>
```

Listing 2

```
<cffunction name="GetAddress1" access="public" returntype="string" out-
put="No">
<cfreturn variables.instance.Address1>
</cffunction>
```

Listing 3

```
<cffunction name="SetAddress1" access="public" returntype="boolean" out-
put="No">
<cfargument name="data" required="Yes" type="string">
<cfset variables.instance.Address1 = arguments.data>
<cfreturn true>
</cffunction>
```

Listing 4

```
<cffunction name="SetStateID" access="public" returntype="boolean" out-
put="No">
<cfargument name="data" required="Yes" type="string">
<cfargument name="DSN" required="yes" type="string">
<cfset var getState = "">

<cfquery datasource="#arguments.dsn#" name="GetState">
```

```
select States.State, States.Abbreviation
from States
where States.StateID = #arguments.data#
</cfquery>
```

```
<cfset variables.instance.StateID = arguments.data>
<cfset variables.instance.State = getState.State>
<cfset variables.instance.StateAbbreviation = getState.Abbreviation>
<cfreturn true>
</cffunction>
```

Listing 5

```
<cffunction name="Init" access="public" returntype="numeric"
output="No">
<cfargument name="AddressID" required="yes" type="numeric">
<cfargument name="DSN" required="yes" type="string">
<cfset var GetAddress = "">
```

```
<cfquery datasource="#arguments.dsn#" name="GetAddress">
select Addresses.*, states.State, States.Abbreviation
from Addresses, States
where Addresses.AddressID = #arguments.AddressID# and
States.StateID = Addresses.StateID</cfquery>
```

```
<cfscrip>
variables.instance.AddressID = GetAddress.AddressID;
variables.instance.Address1 = GetAddress.Address1;
variables.instance.Address2 = GetAddress.Address2;
variables.instance.City = GetAddress.City;
variables.instance.StateID = GetAddress.StateID;
variables.instance.State = GetAddress.State;
variables.instance.StateAbbreviation = GetAddress.Abbreviation;
variables.instance.Zip = GetAddress.Zip;
variables.instance.Phone = GetAddress.Phone;
variables.instance.Fax = GetAddress.Fax;
variables.instance.DateCreated = GetAddress.DateCreated;
variables.instance.TimeCreated = GetAddress.TimeCreated;
variables.instance.DateModified = GetAddress.DateModified;
variables.instance.TimeModified = GetAddress.TimeModified;
</cfscrip>
```

```
<cfreturn true>
```

```
</cffunction>
```

Listing 6

```
<cffunction name="CommitAddress" access="public" returntype="Boolean">
<cfargument name="DSN" required="yes" type="string">
<cfset var CreateAddress = "">
<cfif variables.instance.AddressID GT 0>
<cfquery datasource="#arguments.dsn#" name="CreateAddress">
update Addresses
set Address1 = '#variables.instance.Address1#',
Address2 = '#variables.instance.Address2#',
City = '#variables.instance.City#',
StateID = '#variables.instance.StateID#',
Zip = '#variables.instance.Zip#',
Phone = '#variables.instance.Phone#',
Fax = '#variables.instance.Fax#',

DateModified = #createodbcdate(now())#,
TimeModified = #CreateODBCTime(now())#
where AddressID = #variables.instance.AddressID#
</cfquery>

<cfelse>
<cfquery datasource="#arguments.dsn#" name="CreateAddress">

SET NOCOUNT ON
Insert Into Addresses (Address1, Address2, City, StateID,
Zip, Phone, Fax,
DateCreated, TimeCreated,
DateModified, TimeModified)
values(
'#variables.instance.Address1#', '#variables.instance.Address2#',
```

```
'#variables.instance.City#',
'#variables.instance.StateID#',
'#variables.instance.Zip#', '#variables.instance.Phone#', '#vari-
ables.instance.Fax#',
#createodbcdate(now())#, #CreateODBCTime(now())#,
#createodbcdate(now())#, #CreateODBCTime(now())#
)
Select @@identity as AddressID
SET NOCOUNT OFF
</cfquery>
```

```
<cfset variables.instance.AddressID = #CreateAddress.AddressID#>
</cfif>
<cfreturn true>
```

```
</cffunction>
```

Listing 7

```
<cffunction name="GetWorkAddress" access="public" returntype="address"
output="No">
<cfreturn variables.instance.WorkAddress>
</cffunction>

<cffunction name="SetWorkAddress" access="public" returntype="boolean"
output="No">
<cfargument name="data" required="Yes" type="address">
<cfset variables.instance.WorkAddress = arguments.data>
<cfreturn true>
</cffunction>
```

Download the Code...

Go to www.coldfusionjournal.com

fast • easy • affordable •

CommonSpot™ Content Server

Efficient Content Management

- 100 % browser-based
- Content object architecture
- Template driven pages
- 50+ standard elements
- Extensible via ColdFusion
- Content reuse
- Content scheduling
- Flexible workflow
- Granular security
- CSS support
- SOB compliance
- Personalization
- Replication
- Custom metadata
- Custom authentication
- Static site generation
- Multilanguage support

With CommonSpot Content Server you get it all. CommonSpot's exceptional blend of rapid deployment, ease of use, customization and scalability make it the leading ColdFusion content management solution.

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

For the past six years, PaperThin has been a leader in the ColdFusion community, and CommonSpot has been the solution of choice for organizations of all sizes, including Architect of the Capitol, AFL-CIO, Boeing, Kaiser Family Foundation, Ohio University, PGA.com and hundreds of others. CommonSpot's sophisticated feature set and affordable pricing are an unbeatable combination.

Call us today at 800.940.3087 to schedule a live demonstration of your site running under CommonSpot, or visit www.paperthin.com to learn more.

Paper | Thin

© Copyright 2004 PaperThin, Inc. All rights reserved.

Thinking Outside the Table

PART 2

Storing search results

A two-part series looks at techniques for shifting workload away from the application server and onto the database by using “extra” database tables.

It's just an average search – three full text indexes, ten subselects on many-to-many joins, and a bit of Pythagoras – to find results within 2km using latitude and longitude. It's the sort of query that makes your database give up just thinking about it, and your customers want to run it twice every second.

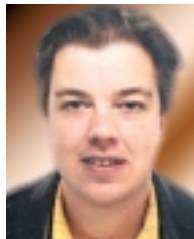
Sometimes databases will surprise you with just how quickly they can perform complex tasks. Sometimes, it's the opposite. You've normalized everything, indexed everything, cut down NULs, applied foreign keys, etc., and still it runs slowly.

The usual suspects are the tools that get used far more in Web publishing than the financial and administrative tasks on which database performance benchmarks are often based, for instance, outer joins and subselects. Both are vital tools for classifying and sorting the often semi-structured data that goes onto Web sites.

For example, imagine that you have products joined to subcategories joined to categories via a chain of joins (shown in Figure 1), and to search by category you have to first find which subcategories belong to that category and then find which products are in that subcategory.

Looking at an execution plan for that query using subselects, you can see why your database is feeling the strain.

Using joins would be a more efficient way of doing this, but sometimes when search criteria and relationships are optional, you have to either use an outer join, which is slow, or you



By Tom Peer

end up having to construct the query in ColdFusion and optionally include joined tables, which can be torturous.

If you're working on Web sites, the chances are good that you're going to end up writing queries that run slowly (hopefully not too often). Fortunately, ColdFusion has several features to help alleviate the problem; timed caches and putting queries into shared scopes are the two primary means, and the only two that most programmers will ever need. There are times, though, when these

techniques aren't enough. Timed caches can't currently use parameters and aren't advisable on queries that get supplied with many different values (product searches would probably fit this criteria), as the cache will quickly fill up.

Putting queries into a persistent scope is the most underused technique by new developers and is probably the most over-used technique by experienced ones. It's ideal for storing queries that rarely change, but putting search queries into the session scope is not the answer for storing the results of complex queries.

In early versions of my database front-end system, Article Manager, I attempted to relieve load on the database for complex queries by storing the value list of primary keys from search results in the session scope and then running a subselect for subsequent results pages. It worked fine for small results sets, but the implications for large result sets are obvious and not good. I soon changed it to the more scalable solution outlined later in this article, but not before I'd realized it had a lot of advantages besides cutting down load.

By storing valuelists in the session scope, it was easy to achieve some of the features of desktop database packages – such as “omit records,” whereby you exclude records from your results in order to refine your search. The main reason you'd want to do this is so you can use another common feature, “save results,” which allows you to save your results and then revisit them at a later date. These were very easy to achieve using list operations, but the system was limited to light use and small results sets.

I needed a more scalable system and one that did a better job reducing the database load, but I wanted to preserve the useful functions of “omit” and “save results.” The solution was to store

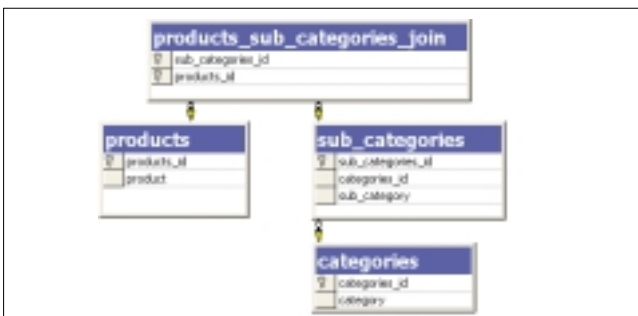


Figure 1: Typical product relations



Figure 2: Execution plan for a query by category ID

the results of the search not in the session scope, but in the database itself. By adding an additional table for search results and storing the primary keys of results sets in it, subsequent searches can be performed using an inner join and a single criterion.

The only thing that needed to be stored in the session scope was the ID number of the search, which could then be used to query the stored results. Say, for instance, you have a table of articles, and you want to perform a search on it. Instead of returning the results directly, you insert the results into a results table first, and then query the database; e.g., to perform the search:

```
INSERT INTO search_table (am_record_id,
am_saved_searches_id)
SELECT articles_id AS am_record_id,
#search_id#
FROM articles
WHERE [search clause here]
```

To get the results themselves, you then join to this table:

```
SELECT headline, description, pubdate
FROM articles A
```

```
INNER JOIN search_table S
ON A.articles_id = S.am_record_id
WHERE S.am_saved_searches_id =
#search_id#
```

This technique is not without its critics. It places an overhead on any given search, with insertions and indexing of the search table, and the join required to get the results carries an overhead when compared to the simplicity of a single search on one table, albeit a small one. Figure 4 shows the execution plan for the following query:

```
SELECT headline, description, pubdate
FROM articles A
where pubdate > {d '2004-11-01'}
```

Databases are built to ensure that queries such as this run quickly, and provide caching mechanisms to ensure that repeated calls run even quicker. If all your queries look like this, then saving results in the database is not for you. If, however, your queries look like my queries sometimes do (large and

unwieldy), and you like the idea of result sets and saved results, then this technique has many advantages.

First, your queries don't have to get much more complicated than this before you'll see a performance gain from storing results. If you're returning small numbers of results compared to the total number of records, then the overhead of adding to the search results table becomes less important. Searches using the LIKE operator can also be slow, as can OUTER joins, and if you have big, complex queries that aren't completely parameterized, then compiling them will take time, and storing results can alleviate load.

But it's not until you start to use some of the techniques discussed earlier that

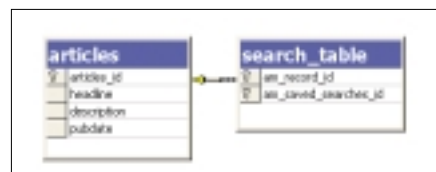


Figure 3: Joining the search table to the main table

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.ACTIVEPDF.COM		31
CFDYNAMICS	WWW.CFDYNAMICS.COM	866-233-9626	4
COLDFUSION DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/COLDFUSION/	888-303-5282	43
EKTRON	www.ektron.com/cfdj		21
EV1 Servers	WWW.EV1SERVERS.NET	800-504-SURF	6
FUSETALK	WWW.FUSETALK.COM	866-477-7542	37
HAL HELMS, INC	WWW.HALHELMS.COM		41
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM/BLACKSTONE	877-215-4678	29
INTERAKT ONLINE	WWW.INTERAKTONLINE.COM		19
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800-379-7729	COVER IV
MACROMEDIA	WWW.MACROMEDIA.COM/GO/VOLVO		2-3
MACROMEDIA WEB PUBLISHING SYSTEM	WWW.WEBPUBLISHINGSYSTEM.COM		51
PAPERTHIN	WWW.PAPERTHIN.COM	800-940-3087	39
SEAPINE SOFTWARE	WWW.SEAPINE.COM	888-683-6456	15
SERVICESIDE	WWW.SERVICESIDE.NET	888-682-2544	17
WEB SERVICES EDGE EAST	WWW.SYS-CON.COM/EDGE	201-802-3066	47

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 disclaimer 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.

"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.



Figure 4: Execution plan for a search on a single table with a single criterion

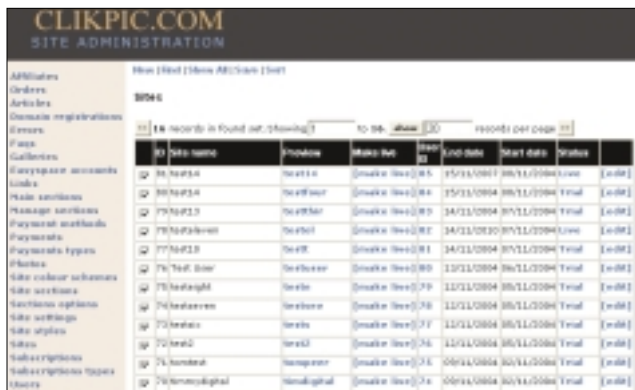


Figure 5: Article Manager in action, showing a found set created by ticking selected records after a search, and then omitting unchecked records. The results can then be saved for future use.



Figure 6: Search options for Article Manager

this technique comes into its own. Omitting records from a search set is simply a matter of deleting records from the join table. Saving search sets is as easy as saving the ID of the search in question. The search results can be reloaded at any time without having to rerun the query.

Another neat feature is the ability to search within the found set or add to the found set. Figure 5 shows the search options for Article Manager when a found set already exists. This is easily achieved as the results of the found set are in the search table and can be used in a join or a subquery.


If you're in a situation where you don't have any write access to the tables, then obviously the technique can't be used. If you're in a position where you have SELECT and STORED PROCEDURE access only, then you can still use the technique in SQL Server by making use of a new feature in SQL Server 2000, the ability of a user-defined function to return a table. First you write a function that takes a varchar list of IDs and splits them to return a table of separate IDs. You can then use a select query to generate your results and a stored procedure, which takes a varchar parameter to store them. The stored procedure would look something like this:

```
INSERT INTO search_table (am_record_id, am_saved_searches_id)
SELECT ID, @searchID from SplitParameterWithCommas(@IDLIST)
```

where @IDLIST would be a varchar, something like "1,3,5,3,7,1,4,6,7,3."

See Listing 1 for an example of returning a table from a UDF

Conclusion

In summary, this isn't a technique for everybody. The downsides are obvious – the extra overhead on the initial search and the need to have write access to the tables. You need to have good reasons to adopt it, but there are plenty of them. If your site's performance is suffering under the load of many-to-many joins, text searches, grouping, filtering, and sorting, then it may be the answer. If you like the idea of saved search sets, then it's definitely for you. 

About the Author

Tom Peer is a Web developer specializing in online journals and magazines (www.digitalmethod.co.uk). He was formerly manager of the New Scientist online, one of the top science Web sites, and now runs an agency providing design, development, and hosting services to a number of other UK publishers.

tom.peer@digitalmethod.co.uk

Listing 1: SplitParameterWithCommas

```
--##SUMMARY Return a table with one row for each of the values
--##SUMMARY supplied in the comma separated list (NB int values only)
--##DETAILS Useful for inserting into temporary tables or for doing
--##DETAILS stored results
--##DETAILS E.g. select * from
SplitParameterWithCommas('1,3,5,3,7,1,4,6,7,3')
```

```
CREATE FUNCTION SplitParameterWithCommas (@paramList varchar(3000))
```

```
RETURNS @params TABLE
```

```
(
    ID int
)
```

```
AS
BEGIN
```

```
declare @start_char int, @paramVal varchar(100), @charTemp char(1)
select @start_char = 1
select @paramVal = ''
while (@start_char <= len(@paramList))
begin
    select @charTemp = SUBSTRING ( @paramList , @start_char , 1 )
    if (@charTemp = ',')
    begin
        insert into @params values (@paramVal)
        select @paramVal = ''
    end
    else
    begin
        select @paramVal = @paramVal + @charTemp
    end
    select @start_char = @start_char + 1
end
```

```
insert into @params values (@paramVal)
```

```
RETURN
```

```
END
```

Download the Code...

Go to www.coldfusionjournal.com

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



Designer Coffee and the Decorator Pattern

Avoiding the problem
of class explosion



While standing in line at my local coffee shop the other day, I was thinking about how helpful a thorough knowledge of design patterns is for a developer. In case you're unclear about what design patterns are, think of them as time-tested solutions to very specific problems.

I was thinking of design patterns as similar to woodworking joints. Each joint is a time-tested solution to a very specific problem. For example, what joint would a woodworker use where the joint will be subjected to being pulled apart – something that occurs with drawers? The best design pattern for this situation is a dovetail joint – one that gets tighter as tension is applied. If you don't know about the dovetail joint or don't know how to cut one, your drawer is likely to fail over time.

Programming design patterns offer similar help to programmers. Because most books on design patterns don't have ColdFusion examples, learning the repertoire of design patterns is especially difficult for us ColdFusion programmers.

All of this was rumbling through my head when I realized I had been in line for over 10 minutes. Now, I'm as understand-



By Hal Helms

ing as the next guy. I understand that the patrons ahead of me are eager to get their beverage just right. But after 13 minutes of waiting for such esoteric orders as "Half-caf extra hot low-foam cappuccino with an espresso shot and hazelnut syrup," well, a man can stand only so much.

That's how I got the idea for my new venture: a fully automated coffee shop! Of course, the name is important and so I sought professional advice from a branding firm. I spoke with their VP for Really Cool Names, J. Billingsly Farnsworth III, and

explained my idea.

"I like it!" exclaimed J. "It's bold! It's innovative! And I have just the name for it!" (Marketing people, I found out, speak almost exclusively in exclamation marks.)

"What did you come up with?" I asked.

"Ready? Here it is: StarBoard Coffee!"

"StarBoard?" I asked. "Isn't that a little close to...you know?"

"Ha! We'll leverage their brand equity to achieve a market penetration that will place us in a position to enter the constant-growth economy!"

"Umm...I see," I lied.

"And I have your tagline! You ready for this? Here it is: 'We Do Coffee Right'"

"Err...isn't that awfully close to another company's tagline?" I asked nervously.

"Yeah, but they mean 'right' as in 'correct' and we mean 'right' as in the direction! Get it? Starboard: Right. It's brilliant!"

I've run it by our permanent in-house focus group using a galvanometer test and they concur! Now, I've put together an advertising plan that'll we'll narrowcast based on lifestyle segmentation and psychographics..."

He said a lot more, but I gathered that with his marketing and my idea, I was well on my way to joining the pantheon of superstar CEOs. Only one thing remained: actually creating the AutoBarista.

The real secret to success would be creating a machine that would provide the kind of flexibility that today's coffee consumers demand while making future changes and upgrades simple. The hardware part was easy enough, but getting the software right would be a challenge.



Figure 1: The AutoBarista's stunning interface

I started with the prototype. This would be the menu customers would encounter at any one of the thousands of stores I had planned (see Figure 1).

Here is the code that produces the menu:

```
<h2>Please select your StarBoard beverage!</h2>

<form action="Order.cfm" method="post">
<h3>Beverage type</h3>
  <p>
    <input type="Radio" name="beverage" value="coffee" checked />
    Coffee <br />
    <input type="Radio" name="beverage" value="tea" /> Tea
  </p>

<h3>Standard Options</h3>
  <p>
```

```
    <input type="Checkbox" name="options" value="Cream" /> Cream <br />
    <input type="Checkbox" name="options" value="SoyMilk" /> Soy
    milk<br />
    <input type="Checkbox" name="options" value="EspressoShot" />
    Espresso shot<br />
  </p>

<h3>Flavored Syrups</h3>
  <p><input type="Checkbox" name="syrops" value="Almond" /> Almond<br
  />
    <input type="Checkbox" name="syrops" value="Vanilla" />
    Vanilla<br />
    <input type="Checkbox" name="syrops" value="Hazelnut" />
    Hazelnut<br />
    <input type="Checkbox" name="syrops" value="NewCar" /> New
    car <br /></p>
<br />
<input type="Submit" value="Place order" />
</form>
```

Having worked on many projects, I knew that the most important thing I could do to help ensure my new creation's success was to provide it with ease of maintenance. After all, 70–90% of an application's cost throughout its entire life cycle is spent on maintenance.

Now, when I think of ease of maintenance, I think of object orientation. This is one of OO's great strengths, after all. So, I began thinking about the various CFCs I would need for my AutoBarista. Each class would need methods for returning the cost and description of each coffee drink. I began listing some of the classes I would need:

- Coffee
- CoffeeWithCream
- CoffeeWithSoyMilk
- CoffeeWithCreamAndShotOfEspresso
- CoffeeWithSoyMilkAndShotOfEspresso
- CoffeewithCreamAndFlavorSyruP

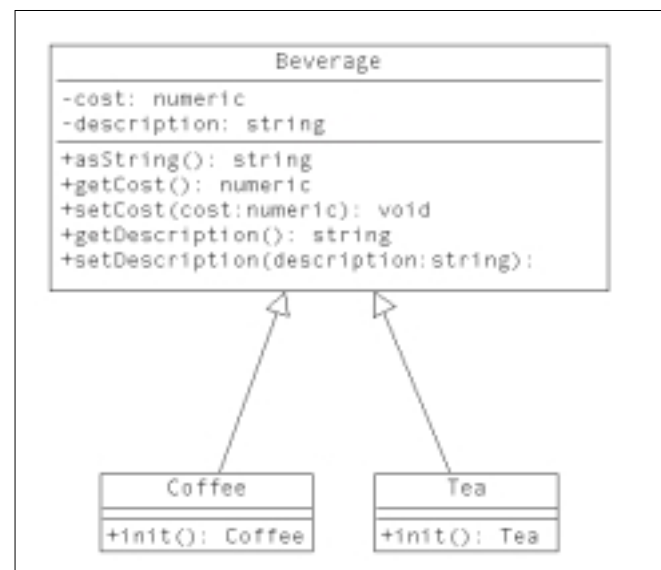


Figure 2: Beverage superclass

- CoffeeWithCreamAndFlavorSyrupAndShotOfEspresso
- CoffeeWithSoyMilkAndShotOfEspressoAndFlavorSyrup

I looked over my partial list and realized something was very wrong. If for each time I had a new option for a coffee drink I had to create classes to reflect all the possible combinations of coffee with options, I'd have an unmaintainable monster on my hands. And then, I'd have to repeat it all for tea and any other beverages I might offer. I needed a better idea.

"Better ideas" are exactly what design patterns provide. They deal with recurring problems that often have no obvious, or simple solution. Take my problem, for example: class explosion. One particular design pattern, the Decorator pattern, provides time-tested, expert guidance on how to deal with the problem of base objects with multiple options while avoiding the problem of class explosion.

To understand how decorators work, let's first look at how I might model coffee and tea (without options) using CFCs. Since there is a great deal of similarity between coffee and tea, I've chosen to abstract common functionality into a superclass, Beverage (see Figure 2).

If you're new to the UML class diagrams I'm using, here's a quick primer. Each box contains three sections. The top section is the name of the class. The middle section (empty for Coffee and Tea) describes any properties (data) associated with the class. The bottom section specifies the methods associated with the class. The arrows point from the subclass to the superclass.

Here is the code for these three classes.

```
<cfcomponent displayname="Beverage">
  <cfset variables.cost = 0 />
  <cfset variables.description = "" />

  <cffunction name="getCost" access="public" returntype="numeric"
output="false">
    <cfreturn variables.cost />
  </cffunction>
  <cffunction name="setCost" access="public" returntype="void" out-
put="false">
    <cfargument name="cost" type="numeric" required="true" />
    <cfset variables.cost = arguments.cost />
  </cffunction>

  <cffunction name="getDescription" access="private"
returntype="string" output="false">
    <cfreturn variables.description />
  </cffunction>
  <cffunction name="setDescription" access="private"
returntype="void" output="false">
    <cfargument name="description" type="string" required="true" />
    <cfset variables.description = arguments.description />
  </cffunction>

  <cffunction name="asString" access="public" returntype="string"
output="false">
    <cfreturn getDescription() />
  </cffunction>
```

```
</cfcomponent>
Code for Beverage.cfc

<cfcomponent displayname="Coffee" extends="Beverage">

  <cffunction name="init" access="public" returntype="Coffee" out-
put="false">
    <cfset setCost(getCost() + 2.55) />
    <cfset setDescription("Coffee " & getDescription()) />
    <cfreturn this />
  </cffunction>
</cfcomponent>
Code for Coffee.cfc
```

```
<cfcomponent displayname="Tea" extends="Beverage">

  <cffunction name="init" access="public" returntype="Tea"
output="false">
    <cfset setCost(getCost() + 2.70) />
    <cfset setDescription("Tea " & getDescription()) />
    <cfreturn this />
  </cffunction>
</cfcomponent>
Code for Tea.cfc
```

Most of the functionality is located in the Beverage class. Coffee and Tea have a pseudo-constructor, init, that initializes the objects created from these classes.

The problem of adding what could work into many options for each different beverage is a daunting one. The first "solution" I arrived at (class explosion) is no solution at all. But the Decorator pattern offers a solution that is flexible and manageable.

A Decorator is a clever thing. It nests or wraps the base object within itself. The Decorator is the same data type as the object it decorates. In my case, all Decorators will extend the Beverage class, making each decorator a Beverage. The Decorator can then be used for wherever the base object is called. And because the Decorator has access to the base object, it can call any nonprivate methods on it that it needs to.

To see how this works, let's consider the case of a customer

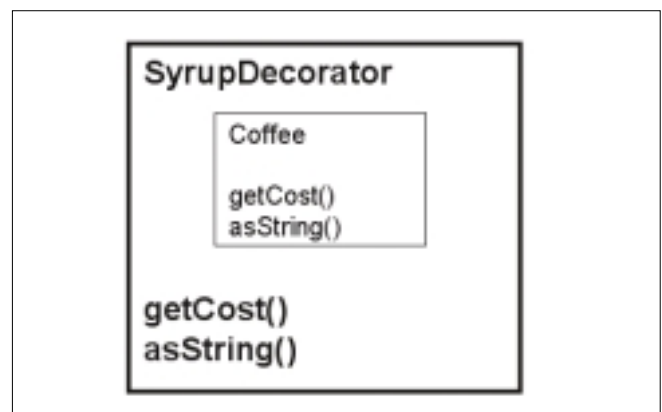


Figure 3: SyrupDecorator



web services **EDGE**
conference & expo

Web Services Edge
2005 East

International Web Services Conference & Expo

Hynes Convention Center, Boston, MA
February 15-17, 2005

The Largest
i-Technology
Event of
the Year!



**Guaranteed
Minimum
Attendance
3,000
Delegates**



Tuesday, 2/15:
Conference & Expo

Wednesday, 2/16:
Conference & Expo

Thursday, 2/17:
Conference & Expo

Join us in delivering the latest, freshest, and most proven Web services solutions... at the *Fifth Annual Web Services Edge 2005 East - International Conference & Expo* as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics including:

- Transitioning Successfully to SOA
- Federated Web Services
- ebXML
- Orchestration
- Discovery
- The Business Case for SOA
- Interop & Standards
- Web Services Management
- Messaging Buses and SOA
- SOBAs (Service-Oriented Business Apps)
- Enterprise Service Buses
- Delivering ROI with SOA
- Java Web Services
- XML Web Services
- Security
- Professional Open Source
- Systems Integration
- Sarbanes-Oxley
- Grid Computing
- Business Process Management
- Web Services Choreography

3-Day Conference & Education Program features:

- Daily keynotes from companies building successful and secure Web services
- Daily keynote panels from each technology track
- Over 60 sessions and seminars to choose from
- Daily training programs that will cover Web Service Security, J2EE, and ASP.NET
- FREE full-day tutorials on .NET, J2EE, MX, and WebSphere
- Opening night reception

Interested in Exhibiting, Sponsoring or Partnering?

Becoming a Web Services Edge Exhibitor, Sponsor or Partner offers you a unique opportunity to present your organization's message to a targeted audience of Web services professionals. Make your plans now to reach the most qualified software developers, engineers, system architects, analysts, consultants, group leaders, and C-level management responsible for Web services, initiatives, deployment, development and management at the region's best-known IT business address - The Hynes Convention Center in Boston.

For exhibit and sponsorship information please contact Jim Hanchrow at 201.802.3066, or e-mail at jimh@sys-con.com.

Sponsored by:



All brand and product names mentioned above are trade names, service marks or trademarks of their respective companies.

Contact for Conference Information: Jim Hanchrow, 201-802-3066, jimh@sys-con.com



www.sys-con.com/edge

who wants to add a shot of vanilla syrup to his or her coffee (see Figure 3).

In ObjectLand, I would create the base coffee object with this code:

```
<cfset beverage = CreateObject('component', 'Coffee').init() />
```

Then, I can decorate the beverage with a SyrupDecorator:

```
<cfset beverage = CreateObject('component', 'SyrupDecorator').init(beverage, 'vanilla') />
```

Since both the original beverage and the decorated beverage are of type Beverage, any method that's expecting a Beverage object will accept either a base beverage or a decorated beverage: both have the same data type and both will respond to the "getCost" and "asString" methods.

We'll see shortly one particularly important method that will accept a beverage. First, here is the code for the superclass, Decorator.cfc, and one of the Decorator subclasses, SyrupDecorator.cfc:

```
<cfcomponent displayname="Decorator" extends="Beverage">
    <cfset variables.baseObject = "" />

    <cffunction name="getBaseObject" access="private"
    returnType="Beverage" output="false">
        <cfreturn variables.baseObject />
    </cffunction>

    <cffunction name="setBaseObject" access="private" returnType="void"
    output="false">
        <cfargument name="baseObject" type="Beverage" required="true" />
        <cfset variables.baseObject = arguments.baseObject />
    </cffunction>
</cfcomponent>
Code for Decorator.cfc
```

```
<cfcomponent displayname="SyrupDecorator" extends="Decorator">

    <cffunction name="init" access="public" returnType="SyrupDecorator"
    output="false">
        <cfargument name="baseObject" type="Beverage"
        required="true" />
        <cfargument name="flavor" type="string" required="true" />
        <cfset setBaseObject(arguments.baseObject) />
        <cfset setDescription( getBaseObject().asString() & " with "
        & arguments.flavor & " syrup") />
        <cfset setCost(getBaseObject().getCost() + .55) />
        <cfreturn this />
    </cffunction>
</cfcomponent>
```

The init method accepts two arguments, a baseObject (of type Beverage) and a string describing the flavor of the syrup to be added to the customer's coffee order. This init method calls the "setBaseObject" method of its parent, Decorator, thereby allowing the decorator to hold the beverage passed into it as an instance variable.

Fine, you say. But how does this help us? Let's take a look at another class that I certainly hope will be getting a lot of use, the CashRegister.

```
<cfcomponent displayname="CashRegister">

    <cffunction name="init" access="public" returnType="CashRegister"
    output="false">
        <cfreturn this />
    </cffunction>

    <cffunction name="order" access="public" returnType="void" output="true">
        <cfargument name="beverage" type="Beverage" required="true" />

        You have selected our fine #arguments.beverage.asString().
        That will be #DollarFormat(arguments.beverage.getCost())# <br />
    </cffunction>
</cfcomponent>
```

Notice that there is a single "order" method that describes the beverage ordered and shows the customer the amount owed by calling the "getCost" method of the beverage passed into the CashRegister.

Now, when a customer orders a simple coffee (no options), an object of type Coffee will be created and passed to the CashRegister's "order" method. Since a coffee is a Beverage, it has a "getCost" method and, in the Coffee CFC, the init method specifies that the base price for a cup of coffee is a mere \$2.55.

But when a customer wants a cup of coffee with flavored syrup added to it, I create the Coffee object and then pass it in to the SyrupDecorator's init method (where the Coffee object is registered as the SyrupDecorator's base object). Notice that in SyrupDecorator (and in all the Decorator subclasses), the

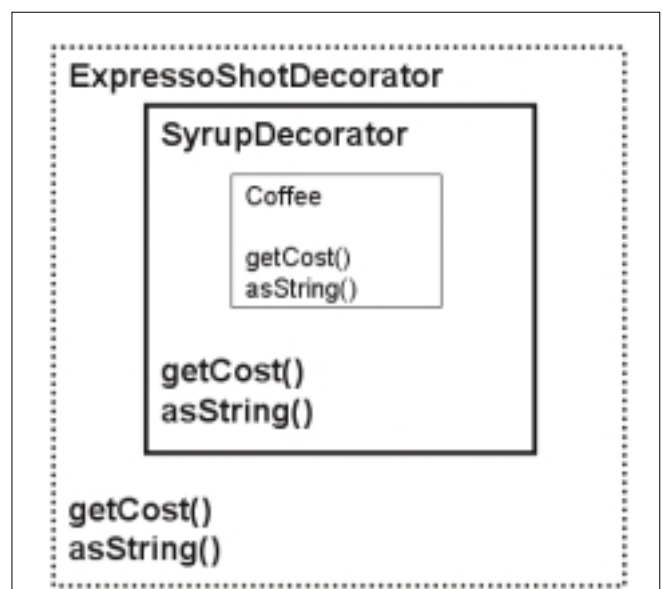


Figure 4: EspressoShotDecorator

“getCost” method has been changed from the initial version in Beverage. Each Decorator subclass says that its cost is whatever its base object’s cost is plus any add-on for the option. In the case of flavored syrups, that add-on is \$0.55.

Now that I have the SyrupDecorator object (with its base Beverage object), I can pass this to the CashRegister’s “order” method. This is an example of the benefits of loose coupling between components and designing to interfaces rather than to implementations.

All well and good, you say, but what if the customer wants a coffee with vanilla syrup and a shot of espresso? Just decorate the decorator. In other words, since all decorators have an init method that accepts a base object of type Beverage, create a SyrupDecorator that decorates a Coffee object and then decorate the SyrupDecorator with an EspressoShotDecorator (see Figure 4).

The code for this would look like this:

```
<cfset beverage = CreateObject('component', 'Coffee').init() />
<cfset beverage = CreateObject('component', 'SyrupDecorator').init(beverage, 'vanilla') />
<cfset beverage = CreateObject('component', 'EspressoShotDecorator').init(beverage) />
Order.cfm, the page that the user selection form posts to, handles this with a few short lines of code:
<!-- If needed, create cash register -->
<cfif NOT IsDefined('application.cashRegister')>
    <cfset application.cashRegister = CreateObject('component', 'CashRegister').init() />
```

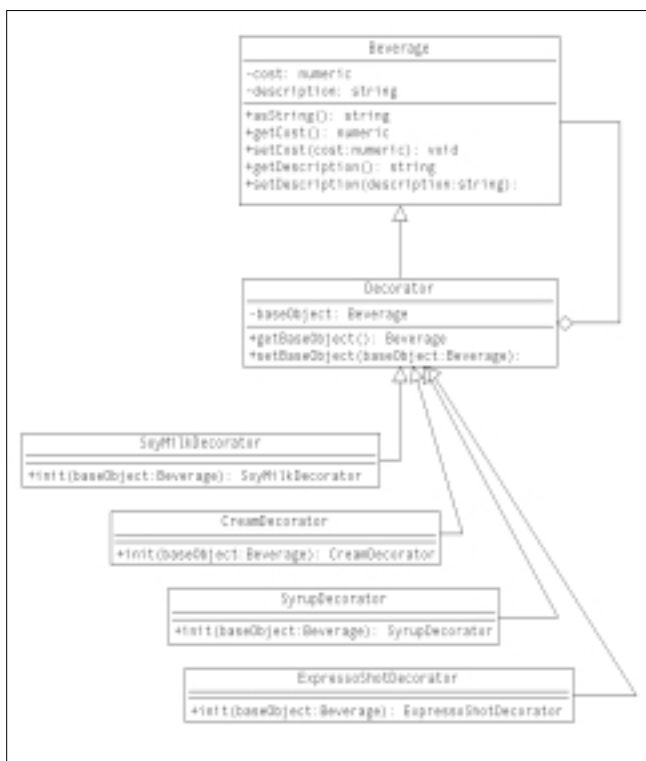


Figure 5: UML for the Decorator CFCs

```
</cfif>

<!-- Create base beverage -->
<cfset beverage = CreateObject('component', '#form.beverage#').init() />

<cfparam name="form.options" default="" />
<cfparam name="form.syrups" default="" />

<!-- If any non-syrup options were selected, create decorators for these -->
<cfloop list="#form.options#" index="option">
    <cfset beverage = CreateObject('component', '#option#Decorator').init(beverage) />
</cfloop>

<!-- If any syrup options were selected, create decorators for these -->
<cfloop list="#form.syrups#" index="syrup">
    <cfset beverage = CreateObject('component', 'SyrupDecorator').init(beverage, syrup) />
</cfloop>

<!-- Pass the beverage (decorated or not) to cash register -->
<cfset application.cashRegister.order(beverage) />
```

Of course, you can keep on decorating decorators until you have built up your base object with all the options required. Here’s the UML for the Decorator CFCs I added to my original design (see Figure 5).

Notice the new type of connector between Beverage and Decorator. This line with a diamond (rather than an arrow) indicates that the Decorator class will have a property of type Beverage. This is the baseObject that it decorates. The “init” method of each of the Decorator subclasses specifies that it accepts a baseObject of type Beverage. Once a baseObject is provided in the “init” method, each Decorator will have access to the baseObject throughout its life.

Here’s the code for each of the other Decorators:

```
<cfcomponent displayName="CreamDecorator" extends="Decorator">

    <cffunction name="init" access="public" returntype="CreamDecorator" output="false">
        <cfargument name="baseObject" type="Beverage" required="true" />
        <cfset setBaseObject(arguments.baseObject) />
        <cfset setCost(getBaseObject().getCost()) />
        <cfset setDescription( getBaseObject().asString() & " with cream ") />
        <cfreturn this />
    </cffunction>
</cfcomponent>
Code for CreamDecorator.cfc

<cfcomponent displayName="EspressoShotDecorator" extends="Decorator">

    <cffunction name="init" access="public">
```


Once you're in it...



...reprint it!

- ColdFusion Developer's Journal
- Java Developer's Journal
- Web Services Journal
- Wireless Business & Technology
- MX Developer's Journal
- PowerBuilder Developer's Journal
- .NET Developer's Journal
- LinuxWorld Magazine
- WebSphere Journal
- WLDJ

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

REprints



foundations

```
returntype="EspressoShotDecorator"
output="false">
    <cfargument name="baseObject"
type="Beverage" required="true" />
    <cfset
setBaseObject(arguments.baseObject) />
    <cfset
setCost(getBaseObject().getCost() + .55) />
    <cfset
setDescription(getBaseObject().asString() &
" with shot of espresso ") />
    <cfreturn this />
</cffunction>
</cfcomponent>
Code for EspressoShotDecorator.cfc
```

```
<cfcomponent displayname="SoyMilkDecorator"
extends="Decorator">

    <cffunction name="init" access="public"
returntype="SoyMilkDecorator" output="false">
    <cfargument name="baseObject"
type="Beverage" required="true" />
    <cfset
setBaseObject(arguments.baseObject) />
    <cfset setDescription(
getBaseObject().asString() & " with soy milk
") />
    <cfset
setCost(getBaseObject().getCost() + .35) />
    <cfreturn this />
</cffunction>
</cfcomponent>
Code for SoyMilkDecorator.cfc
```

Unless you've seen the Decorator pattern previously, the UML diagram probably didn't help you too much, so let's break it down further. Note that the Decorator class extends Beverage. This means that by the magic of polymorphism, a Decorator *is* a Beverage and can be substituted for a Beverage whenever a Beverage is called for.

When might a Beverage object be called for? What about a CashRegister object with an "order" method as shown in the code for Order.cfm? The purpose of this method will be to describe what the customer has ordered and to provide the price for all this goodness. Here is the code for CashRegister.cfc:

```
<cfcomponent displayname="CashRegister">


    <cffunction name="init" access="public"
returntype="CashRegister" output="false">
```

```
<cfreturn this />
</cffunction>
```

```
<cffunction name="order" access="public"
returntype="void" output="true">
    <cfargument name="beverage"
type="Beverage" required="true" />
    You have selected our fine #argu-
ments.beverage.asString()#. That will be
#DollarFormat(arguments.beverage.getCost())#
<br />
</cffunction>
</cfcomponent>
```

The CashRegister asks each Beverage sent into it for the Beverage's description (using the "asString" method) and its cost (using the "getCost" method). Because it accepts a generic "Beverage," there's no need to have the many different methods that would be needed to reflect all the combinations of base beverages and options.

I said that one of the strengths of object-oriented programming is that it eases the burden of maintaining code. Here's what I mean: if we later decide to add an option – perhaps, a shot of Irish Cream – all we need to do is create a new IrishCreamDecorator CFC and add the new option to our menu page. No other code will be affected. Similarly, if I no longer decide to offer an option, I can remove the CFC and remove the display code from Menu.cfm. No other code is affected.

Decorators are a design pattern meant to solve a very specific problem, one in which a base object may have many separate options. Learning how to use the Decorator pattern will give you the ability to use just the right tool in just the right place. 

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.helms@teamallaire.com



When was the last time your Intranet was updated?

Announcing the Macromedia® Web Publishing System. You build and manage the site. Users keep content fresh.

With Macromedia® Studio MX 2004, Contribute™ 3 and Contribute Publishing Services, you can affordably build, manage, and publish enterprise sites. And unlike the average content management system, this works. You determine who can edit and who can publish. Then users simply point and click to update any page. In no time, dated content will be a thing of the past. Learn more at www.webpublishingsystem.com

macromedia
**WEB PUBLISHING
SYSTEM**



be the pilot!

FREE SETUP on Shared
Hosting Accounts With
ColdFusion MX Support
Use Promo Code CFDJ2004



WE DARE YOU TO TAKE A FREE TEST FLIGHT!

Managing technology that runs your business is a matter of trust and control. INTERMEDIA.NET gives you both.

TRUST. Since 1995 we have been providing outstanding hosting service and technology to our clients. Don't take our word for it... take theirs.

"The support and service that you offer are nothing short of golden. The high quality of your system and service for CF customers is something one could only ever dream of." – Claude Raiola, Director, AustralianAccommodation.com Pty. Ltd.

CONTROL. We give you instant control over your site, server and account configuration changes. No more submitting requests and waiting for someone else to take action. You are in control to pilot your business through its daily needs.

BE THE PILOT. Take a free test flight and see what our HostPilot™ Control Panel offers you beyond all others. Check out our SLA guarantees. To see more testimonials and to find out about our competitive advantages, visit our Web site at www.Intermedia.NET.

Managed Hosting • Shared Hosting • Microsoft Exchange Hosting

Call us at: 1.800.379.7729 • Visit us at: WWW.INTERMEDIA.NET

