CHOOSING THE *RIGHT* DATABASE

U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

March 2001 Volume: 3 Issue: 3

Sept. 23–26, 2001 Santa Clara, CA
JAVAEDGE conference&expo
28

Oct. 22–25, 2001 New York, NY
XMLEDGE conference&expo
45

Editorial
Introducing Macromedia ColdFusion
Robert Diamond page 5

Foundations
Exploring the Development Process
Hal Helms page 12

How To
Company Calendar
Kevin Schmidt page 16

Product Reviews
Holistix Web Manager 2.0
Jerry Bradenbaugh page 40
Fireworks 4 Studio
Jerry Bradenbaugh page 52

Q&A
Ask the Training Staff
Bruce Van Horn page 42

RETAILERS PLEASE DISPLAY
UNTIL MAY 31, 2001
$8.99US $9.99CAN

SYS-CON MEDIA

by Charles Arehart page 6

<BF> on <CF>: Database Support Is Going to Get a Whole Lot Better 22
CF5 is about to debut with exciting new features *Ben Forta*

CFDJ Feature: Some Thoughts on the Design of CF Data Input Applications 24
Implementing application functionality *Kailasnath Awati and Mario Techera*

ColdFusion & Java: Five Ways to Call Client- and Server-Side Java from CF 32
You don't have to wait for Pharaoh or switch to JRun *Andrew Cripps*

CFDJ Feature: Comparing Local and Client/Server Databases 36
Choosing a database platform for a ColdFusion Web application *Ted Blue*

Practical CF: Upsize from MSAccess to SQL Server Part 1 of 2 44
Don't be afraid! We'll show you how, step-by-step *Eron Cohen and Michael Smith*

Tips & Techniques: Directory: Structures and You 50
Choosing the best directory structure for your application *John Wilker*

ColdFusion Developer's Journal

EDITORIAL ADVISORY BOARD
STEVEN D. DRUCKER, JIM ESTEN, BEN FORTA,
STEVE NELSON, RICHARD SCHULZE, PAUL UNDERWOOD

EDITOR-IN-CHIEF ROBERT DIAMOND
ART DIRECTOR JIM MORGAN
EXECUTIVE EDITOR M'LOU PINKHAM
EDITOR NANCY VALENTINE
MANAGING EDITOR CHERYL VAN SISE
ASSOCIATE EDITOR JAMIE MATUSOW
PRODUCT REVIEW EDITOR TOM TAULLI
TIPS & TECHNIQUES EDITOR MATT NEWBERRY

WRITERS IN THIS ISSUE
CHARLES AREHART, KAILASNATH AWATI, TED BLUE,
JERRY BRADENBAUGH, ERON COHEN, ANDREW CRIPPS,
ROBERT DIAMOND, BEN FORTA, HAL HELMS, KEVIN SCHMIDT,
MICHAEL SMITH, MARIO TECHERA, BRUCE VAN HORN, JOHN WILKER

SUBSCRIPTIONS
FOR SUBSCRIPTION REQUESTS PLEASE CALL
1 800 513 7111
OR GO TO: **WWW.SYS-CON.COM**

DOMESTIC $89.99/YR. (12 ISSUES)
CANADA/MEXICO $99/YR
OVERSEAS $129/YR
BACK ISSUES $10 U.S. $15 ALL OTHER

PRESIDENT AND CEO FUAT A. KIRCAALI
VP, PRODUCTION JIM MORGAN
SENIOR VP, SALES & MARKETING CARMEN GONZALEZ
VP, SALES & MARKETING MILES SILVERMAN
ADVERTISING ACCOUNT EXECUTIVE RONALD J. PERRETTI
ASSISTANT CONTROLLER JUDITH CALNAN
CREDIT & COLLECTIONS CYNTHIA OBIDZINSKI
ACCOUNTS PAYABLE JOAN LAROSE
ADVERTISING ACCOUNT MANAGER ROBYN FORMA
ADVERTISING ACCOUNT MANAGER MEGAN RING
ASSOCIATE SALES MANAGER CARRIE GEBERT
ASSOCIATE SALES MANAGER CHRISTINE RUSSELL
SALES ASSISTANT ALISA CATALANO
VICE PRESIDENT, CIRCULATION AGNES VANEK
CIRCULATION MANAGER CHERIE JOHNSON
VP, EVENTS CATHY WALTERS
ART DIRECTOR ALEX BOTERO
ASSISTANT ART DIRECTOR CATHRYN BURAK
ASSISTANT ART DIRECTOR LOUIS F. CUFFARI
GRAPHIC DESIGNER ABRAHAM ADDO
GRAPHIC DESIGN INTERN AARATHI VENKATARAMAN
WEB DESIGNER STEPHEN KILMURRAY
WEB DESIGNER GINA ALAYYAN
WEB DESIGN INTERN PURVA DAVE
SYS-CON EVENTS MANAGER ANTHONY D. SPITZER

EDITORIAL OFFICES
SYS-CON MEDIA, INC. 135 CHESTNUT RIDGE RD.,
MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9600

COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
IS PUBLISHED MONTHLY (12 TIMES A YEAR)
FOR $89.99 BY SYS-CON PUBLICATIONS, INC.,
135 CHESTNUT RIDGE RD.,MONTVALE, NJ 07645

POSTMASTER
SEND ADDRESS CHANGES TO:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
135 CHESTNUT RIDGE RD.,MONTVALE, NJ 07645

© COPYRIGHT
COPYRIGHT © 2001 BY SYS-CON MEDIA
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.

FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR.
SYS-CON PUBLICATIONS, INC., RESERVES THE RIGHT TO REVISE,
REPUBLISH, AND AUTHORIZE ITS READERS TO USE
THE ARTICLES SUBMITTED FOR PUBLICATION.

WORLDWIDE DISTRIBUTION
BY **CURTIS CIRCULATION COMPANY** 730 RIVER ROAD,
NEW MILFORD, NJ 07646-3048 PHONE: 201 634-7400

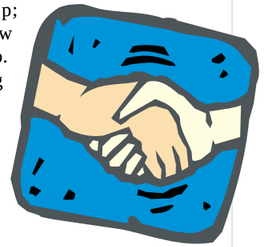
DISTRIBUTED IN USA
BY **INTERNATIONAL PERIODICAL DISTRIBUTORS**
674 VIA DE LA VALLE, SUITE 204, SOLANA BEACH, CA 92075
619 481-5928

ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES
ARE TRADE NAMES, SERVICE MARKS, OR TRADEMARKS
OF THEIR RESPECTIVE COMPANIES.

Introducing Macromedia ColdFusion

BY **ROBERT DIAMOND**

Some of the most interesting news to hit the ColdFusion world in recent memory is the January 16th announcement that Macromedia and Allaire are to merge into a new, larger-than-life Macromedia. The goal of the combined company is to take the best of what each company brings to the table and create one big company that allows its customers to create and develop a Web site – both the design and the application (and everything in between) – from start to finish. They see themselves as a one-stop Web development shop; looking at the combined product list you can see why. The new Macromedia isn't going to stop with just developing for the Web. They're jumping on the wireless bandwagon as well and looking for initiatives to unite development on all of today's wireless devices, from PDAs to cell phones. They want to "deliver everything everywhere," which is certainly a noble goal that, if achieved, will make a CF developer's life just a little bit easier.



When does it all start to come together? The merger is scheduled to be completed during the second quarter of 2001 in a $360-million transaction. The new company will have its headquarters in San Francisco with offices in Boston. In case you're wondering about the new management team – Kevin Lynch will remain Macromedia's president of products, and Jeremy Allaire will join Macromedia as its chief technology officer.

Okay, that's the nitty-gritty of the deal, but the real question in everyone's mind is, "What's happening to ColdFusion?!?" (At least that was the first question that came to my mind.) The answer is that for the time being, at least (and who can really talk much further than that?), the new Macromedia is committed to continuing and expanding the current roadmap including CF 5 and the J2EE-rich Neo, both still slated for release in 2001.

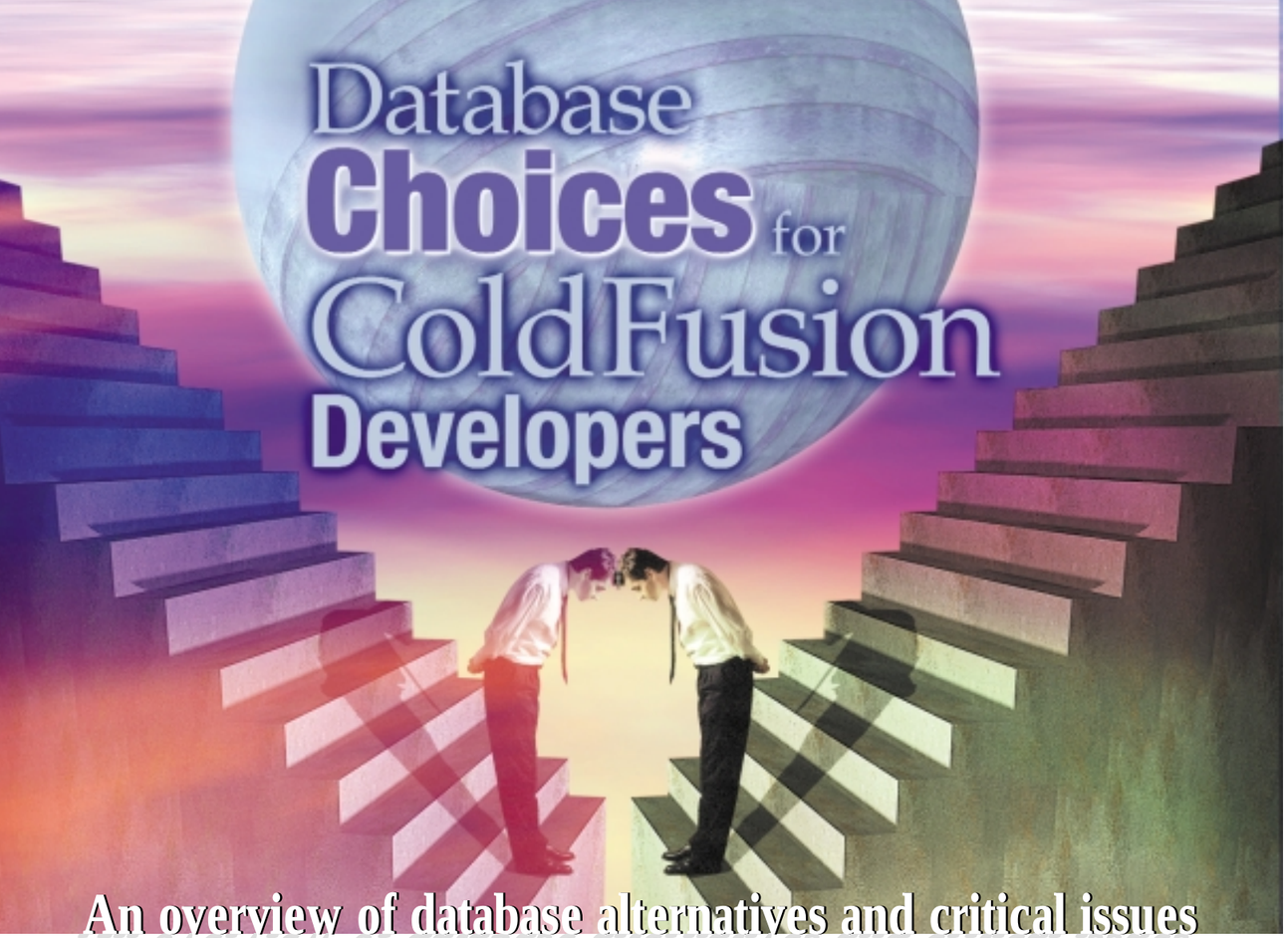
Also with the two companies joined together, we can look forward to increased integration between their various products and services, details of which are still coming out. The first round of Flash components that plug right into CF were demonstrated way back (way back in Internet-time at least) at the Allaire Conference in November. One wonders how long this merger was in the works "behind the scenes"? As evidenced by the cheering crowd at the announcement, making it easier for developers to visually spruce up their products and add functionality with Flash is a good thing. With over 96% market penetration, Flash support on the client end isn't a major concern. And for those few people without it – it's a download so simple that almost anyone can do it. (I know, take nothing for granted, but this one really is easy.)

I can only speak for myself – but if the limit of your graphics ability involves fumbling around in Adobe Photoshop, and you're begging graphic designers for help on your projects, I wouldn't mind saying "Look what I made!" without having to open up a graphics program. That'll show those designers!

So join me and raise a glass in congratulations to the new Macromedia, and look to ***CFDJ*** in the coming months for continuing coverage on what it will mean to you.

ABOUT THE AUTHOR
Robert Diamond is editor-in-chief of **ColdFusion Developer's Journal** *as well as* **SYS-CON**'s *newest magazine,* **Wireless Business & Technology**. *Named one of the "Top thirty magazine industry executives under the age of 30" in Folio magazine's November 2000 issue, Robert is currently a senior in the School of Information Studies at Syracuse University.*

ROBERT@SYS-CON.COM



An overview of database alternatives and critical issues

CFDJ FEATURE

BY **CHARLES AREHART**

What database do you use for your ColdFusion applications? Are you fully satisfied with your choice? Did you even have a choice? Would you be interested in knowing what other alternatives you may have? In this article I present some of the discussions that inevitably arise when considering database choices available to ColdFusion developers.

Not a Simple Matter

When this special database issue was planned, the editors asked if I'd write an article on this subject. I was at once intrigued and concerned. Any discussion of databases is likely to touch off hot buttons. There are so many issues to be considered, including:

- Cost/availability
- Performance/scalability
- Security/administration

Some think the case for a given database platform in these matters would be clear cut, whereas others may argue against those assertions. Then there are the subtler debates, such as whether to choose a commercial DBMS (database management system) or go the open-source route. There are still more points to consider. In any case, a discussion on this subject is a perilous one.

Still, I'll share some thoughts from my perspective as a developer, instructor, and consultant, as well as my observations of such discussions in various forums. I don't claim this article to be a definitive resource on the subject by any stretch. Instead, I hope it stimulates debate and brings some people to a better understanding of the issues and alternatives.

Much of the discussion refers to Microsoft alternatives. As most CF developers are familiar with them, I can make some references to put the discussion in context. At the end, I'll discuss other choices as well, including different platforms and vendors.

The Big Picture Issues

Regardless of the preferences you may have, the constraints you face, or the choice you ultimately make, you'll have certain big picture issues to consider when selecting a DBMS. Probably the biggest factor for most people is cost.

Cost

Cost is a complex variable in the equation of determining DBMS choice. While the up-front price (or lack thereof) to purchase a DBMS may be the most obvious aspect of cost, there are actually multiple facets, including:

- ***Licensing:*** Purchasing the product, including the cost for developer connections and connectivity from Internet-connected clients
- ***Platform requirements:*** Implementing the physical infrastructure needed to support the DBMS, which may include server hardware, networking support, and more
- ***Connectivity:*** Purchasing drivers to connect ColdFusion to the DBMS, if needed
- ***Administration:*** Installing and administering the DBMS, including ongoing care and feeding of backup and recovery, performance tuning, database administration, and much more
- ***Development:*** Creating applications against the chosen DBMS, including cost of modifications or migration if moving from one DBMS to another
- ***Training:*** Training developers and administrators, and in some cases end users, who may be provided query and reporting tools by the DBMS

As you can see, there's much more to the "cost" of a DBMS alternative than the simple up-front purchase price. This point doesn't necessarily favor one DBMS over another. Some may use it to argue that it points up the "false economy" of choosing a "free" open-source DBMS (if the other costs exceed comparable totals for another DBMS), but it could also be argued that a "high-priced" DBMS may be favorable (because the sum of those costs is less in the long run than a free or inexpensive alternative).

Even then, there's so much more than cost to consider when choosing a DBMS.

Availability

By availability I mean the choice of DBMSs already installed and available to you (in-house if you're hosting your Web application yourself, or on your Web service provider if your site is hosted).

Of course, this again presupposes that the "simplest" choice is the best (let's use what we've got), when in fact a careful analysis may show that other alternatives are indeed preferable. If you're not the one who decides what software is installed, though, you're left to choose among what you have.

You may even be specifically precluded from installing an alternative, even a "free" one, because of organizational restrictions on installing new software.

In all these cases, the simple fact is that for some people availability is more important than true cost. As such, the question becomes what choices are available to you? Don't be so quick to presume you know. In an organization of more than a few people, it's possible there are DBMSs installed that you may not know about. Be sure to thoroughly investigate what's available.

Further, as will be discussed later, you may unknowingly have alternatives available at no "cost" (if that's an issue) that have not yet been installed. (Specifically, I'm referring to MSDE, which is available to users of Microsoft Access 2000 and other MS development tools.) You may find you can greatly improve many facets of your DBMS platform by taking advantage of such "undiscovered" alternatives.

Performance

If you've developed a database application of any size, you've certainly learned the importance of database performance. By many estimates, 90% of the cause of poor performance in these applications, especially Web applications such as those enabled by ColdFusion, can be attributed to poor database performance.

The question is: What's the cause of the performance problem? Is it the inability of the DBMS to effectively process the database requests issued against it? The poor quality of the program performing the database requests? Bad database design?

There's no denying some DBMS platforms make much better choices for high-performance, demanding applications. If you need to serve hundreds, thousands, or indeed millions of users a day, your database better be up to the task. Fundamental architectural capabilities are built into large-scale DBMSs to make them more suitable for such high-performance applications.

It's also true that sometimes the very infrastructure required of higher-end DBMSs is a key to their greater capability (this is the "platform requirements" cost mentioned previously). As such, even a lower-cost (lower-end) DBMS may run well with the right combination of platform and programming skills.

Regardless, it's often the case that your current DBMS could handle the job better in your current platform. As pointed out in other ***CFDJ*** articles (see Resources at the end of this article), it's quite possible to stretch a good performance out of even MS Access. Many regard Access as absolutely unsuitable for production use (arguments are made in some of those articles on that very point), but there are situations in which it has proven to work satisfactorily for some applications.

The real question is: What's "production use"? What are the performance requirements of the application? For low-volume requirements, it's possible that low-end DBMS solutions may indeed be capable of performing well.

A more important question, though, is what are the expected performance requirements over the long term? This is a question of scalability: the ability of the database to handle a growing volume of use.

Scalability

Even if you find that a given DBMS choice (or database and application design) is adequately meeting your needs, as your site or application grows in popularity (by good marketing, word of mouth, growing acceptance of it as a technology among your user base, etc.), it may not in the long run.

As the volume increases, is your database capable of handling that higher volume?

A more insidious facet of this issue comes into play when moving a project from the development to the deployment stage. Many organizations or individuals have faced public humiliation when an application that "worked" all through the development cycle suddenly crumbled to its knees when exposed to access on a larger scale.

And "larger scale" is also relative. Some applications fall to pieces when someone other than the developer, even just one other user, uses the application. More realistically, challenges often come when several people (or perhaps not until hundreds or thousands of them) use the application at once. There may be issues of concurrency (problems when several users try to access the same data at once) or integrity (the application may not have been designed to handle all the possible inputs that a broad base of users would provide).

Higher-end DBMSs have better built-in support for larger volumes and these matters of concurrency (row-level versus page-level locking strategies, for instance) and integrity ("declarative referential integrity" and "triggers," to name two).

Then there's the matter of the DBMS's ability to store a given volume of data. Even on a site with relatively sparse traffic (and no concurrency problems), the amount of data generated by the number of visitors trickling through may still eventually overwhelm the DBMS's ability to store that data.

Security

Related (in a way) to the matter of increasing traffic/volume is ensuring that intended users only access data they're permitted to see. There are actually two parts to that:

- ***Authentication:*** Who are they?
- ***Authorization or permissions:*** What are they allowed to do?

This is an area in which many ColdFusion developers must admit a weakness in their implementation. I've seen far too many applications that have virtually no security on the database. Not that the application doesn't implement authentication and authorization: they may indeed require a login to use the application.

The question is: How is the database secured? In too many instances, the database (especially in the case of Access databases) is sitting on the server with no real security. Anyone who can access the file (whether a hacker or another developer on the same server when directories are not adequately protected) could grab the database, open it, and have full access to the data. The sidebar provides information on how to protect your database from public access.

There are two issues involved with fundamental authentication. Is the database protected so it's not open to anyone who can get hold of it? If it's secured with a password, at least it's a step in the right direction.

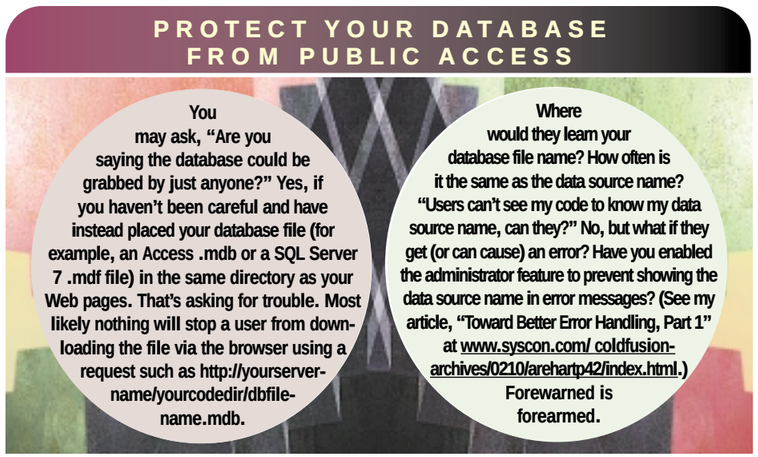
If you want to have user IDs for each user in a real multiuser application, the question is: Where are the user IDs (and passwords) stored? Are they stored in the DBMS (as in SQL Server's native logins)? In a separate file (as in Access's "workgroup information" file)? Or in the operating system authentication repository (such as SQL Server's ability to leverage NT user IDs).

Further, you want a way to associate users with their respective permissions. That's the authorization component. A more complete DBMS security implementation would allow the assignment of a broad range of rights (from the types of SQL statements a user can execute, to changes that can be made to the database, and lots more). Better still would be a means to associate user IDs with groups, so you can assign permissions to groups and place people into or remove them from groups.

Fortunately, even Access can support this kind of security structure (see the Microsoft documentation for more information). Of course, using this sort of security raises the question of whether and how to integrate an application's login security with the database's login security, which is beyond the scope of this article.

Administration

The issue of database administration is often given short shrift by ColdFusion developers. They figure once the database is created and working, the job is done. There are indeed those aforementioned issues of managing the performance, watching for file growth, and ensuring adequate security. Each of these is more effectively managed if the database provides management tools.



On the one hand, an effective interface for implementing such features is valuable. On the other, a means to report on (and act upon) management information is also vital in larger applications. If you're not actively watching the database, you better hope your DBMS (or a tool you can set up) is doing it for you and can notify you (or take corrective action itself). SQL Server 7, for instance, added substantial improvements in the self-diagnosis and repair of a database. It's always been replete with an interface for managing databases (Enterprise Manager), which has only gotten better in Release 7.

Other administration factors include database design and modification, database backup/restore, recovery in the event of a crash, and support for the administration of remote databases, to name a few.

Some Other Issues

We really can't cover every issue in the consideration of DBMS alternatives, though those above are the key factors. Other important ones include:

- ***Portability:*** Support of open standards and compliance with SQL standards
- ***Integrity:*** Includes support for referential integrity, transactions, and triggers
- ***Stored procedures:*** A means to store and easily reuse a library of precompiled, parameter-driven queries
- ***Replication:*** Keeps multiple, perhaps geographically segregated, databases synchronized
- ***XML support:*** The ability to present query results in XML format, or to accept XML to drive database queries and updates

Typically, these features are found only in higher-end databases, but they're also offered – or can be made to mimic them – at the lower end (see my article, "Stored Procedures in Access? Yes, Indeed!" [***CFDJ***, Vol.1, issue 5]).

Things also change with new releases. For instance, Access 2000 has dramatically improved its ability to support SQL that's compliant with SQL Server (which conforms closely to the ANSI-92 standard). From the SQL DDL for defining and altering database objects to the terms used to describe different data types, such as Varchar(20), in addition to Text(20), you can now create applications against Access that will more easily migrate to SQL Server (or any other ANSI-92–compliant DBMS).

Some DBMS Alternatives

I'll now briefly look at some of the major DBMS alternatives. This is also a sticky proposition – besides the matter of different vendors, there's the matter of platforms. If you work in a Linux environ-

ment, for instance, you don't have quite the same choices as a site hosted on a Microsoft platform. You may even be interested in creating a ColdFusion application against a mainframe DBMS (ODBC drivers from mainframe DBMS vendors may enable this).

Naturally, this article can't evaluate all the available DBMS choices. And frankly, if you have a unique platform (by which I'll infer a non-Microsoft platform or your preference to use open source products), you probably have a clear picture of your options. I'll leave it to you to investigate those choices.

However, I think it's fair to assert that most CF developers who can select commercially available DBMSs will seriously consider Microsoft alternatives because of their ubiquity. Still, these people may not realize all the Microsoft options available, or get the chance to consider them carefully. There are three major Microsoft alternatives: Access, MSDE (which may be entirely new to some readers), and SQL Server.

Microsoft Access

Microsoft Access, in both Access 97 and Access 2000 flavors, is a natural choice for many people since it's relatively inexpensive (or is included in some tools and suites). It's a low-cost alternative (on all those cost factors mentioned). It has its detractors, in some cases deserved, as discussed; however, it has its place (see Resources section). For more information visit www.microsoft.com/office/access/default.htm.

Microsoft SQL Server

Microsoft SQL Server, available in SQL Server 7 (circa 1999) and SQL Server 2000 versions, is regarded as the "expensive," enterprise alternative. While it does have higher costs on nearly all counts, the price brings substantial benefit, for some of the reasons cited earlier. You can learn more about it at www.microsoft.com/sql/.

CF developers should note that there's a SQL Server Desktop edition, available in some suites, that's great for single-user development situations. And SQL Server 7 allows the product to run in Windows 95 and 98 platforms (though those aren't good choices for a production platform).

There's also a simple-to-use Upsizing Wizard available in Access that supports upgrading a database to SQL Server (or MSDE). See "Using the Microsoft Access 2000 Upsizing Tools with SQL Server 7.0 or MSDE" at http://msdn.microsoft.com/library/techart/upsize00.htm.

MSDE

A relative newcomer to the debate over which Microsoft platform to choose is Microsoft Data Engine, or MSDE. It's basically SQL Server without an administration interface (it doesn't include Enterprise Manager). Other than that, it looks, smells, and tastes like SQL Server. Seriously, it's SQL Server, just an unusual packaging of it.

> "MSDE is tuned for best optimum performance at five concurrent users or less

It seems to have been created as a bridge that enables Access developers to easily move to SQL Server. Believe it or not, you don't have to buy it; in fact, you can't. It's bundled inside other tools, and here's a surprise: it's bundled with Access 2000 and Office 2000 Developer edition. Install it from the respective CDs of those products.

There's no single resource I can recommend, though a couple of useful ones are worthwhile reading if you're new to MSDE:

- ***Microsoft SQL Server:*** "Microsoft Access 2000 Data Engine Options" – http://msdn.microsoft.com/library/backgrnd/html/acmsdeop.htm
- ***Microsoft Data Engine (MSDE) for Microsoft Visual Studio 6.0:*** "An Alternative to Jet for Building Desktop and Shared Solutions" – http://msdn.microsoft.com/library/backgrnd/html/msdeforvs.htm. (Don't let the Visual Studio reference throw you. It's perfectly applicable to CF developers and has no references to Visual Studio as a tool.)

There's no better way to migrate slowly into SQL Server. The cost barrier is eliminated (the licensing cost), and you're actually free to distribute the DBMS with your application (see the license and the second article above for specific details). The documentation suggests there's a five-person concurrency limit, but I also found the following in that article:

MSDE is tuned for best optimum performance at five concurrent users or less. MSDE can support more than five concurrent users both from a technical and licensing standpoint, but Microsoft strongly suggests that the number of concurrent users be five or less in order to enjoy the performance levels available with MSDE.

I've also heard that MSDE was not licensed for use with Internet applications, but as I haven't seen the specifics of that, I leave it to you to investigate further.

Finally, though Enterprise Manager is not provided with the tool, Access 2000 has been redesigned to support design, administration, and management of MSDE databases via the concept of "projects." See the product for more information on that capability. The bottom line is that if for some reason you haven't yet moved to SQL Server, MSDE provides a great alternative for getting started.

Other DBMS Alternatives

My apologies to those who feel I'm leaving out their preferences. Remember, the premise of this article was to outline the issues and features to consider in your DBMS choices, not to be a comprehensive review.

I've run out of space and, of course, there are a number of other DBMS alternatives to consider, including Oracle, Informix, DB2, and Sybase on the high end and MySQL, SQLDB, and others in the middle to low end. There are still others, and again your mainframe database may provide an ODBC driver to enable access via tools such as CF (as does CCA's Model 204, where I got my start in databases nearly 20 years ago).

I hope this overview of database alternatives and critical issues has helped in your selection of databases as a CF developer.

Resources

- Thompson, R. (2000). "DB Conversion and CF – A User's Tale." *ColdFusion Developer's Journal*. May. www.sys-con.com/coldfusion/archives/0205/thompson/index.html
- Forta, B. (1999). "Take Your Database Out of Retirement." *ColdFusion Developer's Journal*. May/June. www.sys-con.com/coldfusion/archives/0103/forta/index.html
- Van Horn, B. (2000). "In Defense of MS Access." *ColdFusion Developer's Journal*. April. www.sys-con.com/coldfusion/archives/0204/vanhorn/
- Forta, B. (2000). "Access Denied." *ColdFusion Developer's Journal*. June. www.sys-con.com/coldfusion/archives/0206/forta/index.html

About the Author

Charles Arehart is a certified Allaire trainer and CTO of SysteManage, an Allaire partner. He contributes to several CF resources and is a frequent speaker at user groups throughout the country.

carehart@systemanage.com

Exploring the Development Process

12 steps to a successful project

BY HAL HELMS

Recently I was asked by another developer to describe my development process. Since I enjoy hearing how other developers approach issues, I thought you might like to hear how one developer approaches the entire development process.

My process has 12 steps divided into three sections: *discovery*, *design*, and *implementation*.

Discovery

Step 1. *Sales meeting*

A prospective client has contacted me and wants to meet to discuss a possible project. My goal at this meeting is to determine what, in broad terms, the prospect wants. I need to find out a few things:

- What kinds of technologies will be involved?
- How large of a project is this?
- What cost expectations does the prospect have?
- When does the prospect need/want this (and are these two the same)?

I explain the development process I use and see if the prospect feels this will work for them. I also identify who will need to be involved in the next step (if we go further).

After the meeting is over I try to determine whether there seems to be a good fit between us. If it isn't a good match, painful experience has taught me to pass on the job.

Charge for this step: None.

Step 2. *Initial needs meeting*

At the end of Step 1 I've explained that I need key decision makers on this project. If I can't get them (they're too busy, etc.) that pretty much tells me all I need to know about the chances for success. I can't afford a failure (even if the client can) and I'll pass on the job.

But let's assume a rosier scenario – all decision makers are present. My goal is to understand, on a much deeper level, what they want. Are there any technological concerns? I certainly want to know if the new application needs to interface with existing systems.

How about political issues? These are harder to unearth, but they're like land mines so I spend time gently probing for areas of resistance. I tread lightly here. Who is the target user audience for the app? How will they access it? What browsers must be supported? Fast or slow connection? Any prohibitions on cookies, JavaScript, Java?

I want to identify any other people needed for the next step and then set a time. I write a lot of very simple use cases in the form of, "User wants to do so and so"; "System should return a list of such and such" – all single, simple declarative sentences.

Charge: Hourly.

Step 3. *Wireframe meeting*

Between the last meeting and this one I've created a *wireframe*. I use this term to indicate a skeletal miniapplication that shows the user what each page does on a page-by-page basis and what possible locations the page can go to after it has completed its task.

There should be a corresponding wireframe page for each page the user will see in the real application. It explains what the page does and what links to other pages should appear. This is done completely in text. I have a little app I created for this that generates a wireframe from a single "ini" file – very simple, but also very helpful. (You can download this from www.halhelms.com.)

I find the wireframe produces much better results than writing formal specs:

- They don't get read by the people who ultimately will vote yea or nay on paying.
- They're necessarily too vague to have any directive power.

Think of having the Mona Lisa described to you – then using this "specification" to make a reproduction of the painting. A picture is worth a thousand words and a clickable app is worth a thousand pages of wordy specifications.

Charge: Hourly.

Step 4. *Sales meeting*

We go over the final wireframe application to ensure that we're communicating well and understand each other. I present a proposal on what it will take to complete the design phase. If we agree, we go on to the next step.

Charge: None.

Design

Step 5. *Look and feel*

Someone has to come up with graphical comps. That may be me or the client may have staff that can do this. I'm looking typically for a few "views" – the main page and any unique detail pages. In practice, this usually means about three views. If I'm doing them, there are usually at least three separate sets of views, each with a different look and feel. This is big concept stuff. Clients love this – choosing colors, rounded versus drop-shadow buttons – stuff like that. Once a decision is made, we move to the next step.

Charge: Hourly or fixed price.

Step 6. *Prototype and specs*

The prototype integrates the chosen look and feel with the func-

tionality discovered by the wireframe work. This is exactly – *exactly* – what the real app will look like. There's no (or minimal) coding going on here. It's all HTML work with Photoshop graphics. That's important since having coders do this work is overly expensive. Also, coders often are not the best people to help the client work through this crucial phase.

This is a very iterative process. I've found that rushing through this phase to begin coding is a great mistake. Clients usually don't know exactly what they want or need until they've seen it. That's why developers share the universal experience of delivering an application they've built according to client specifications, then having the client provide more requirements.

My answer to this problem is a prototype phase that consists of an initial "cut" of the pages the client will see followed by an iterative series of client reviews, suggestions, and prototype changes. This is, by far, the most time-consuming part of the development process. I've created something called "DevNotes" that lets clients and developers communicate, and then saves this exchange in a database. It looks something like a minithreaded discussion application and goes on the bottom of each prototype page.

Each day clients can check on the progress and make notes. "I think this should be moved up here"; "this text is hard to read on my monitor" – whatever. My goal is to create a prototype the client loves. As I said, this is an iterative process, one that can take quite some time, depending on what the client wants and needs.

Of all the steps I find this to be the most crucial to a successful project. Projects usually don't fail because we couldn't figure out how to implement an efficient sorting algorithm; they fail because we don't give the clients what they want. And most of the time we're set up for failure. The only time the client can really say "yes" or "no" is when they see the real app. But if you've already written the code, it's not a great time to be told, "Not exactly...." This is on the Web so the client can work on it any time; it doesn't require an in-person meeting.

> "Any 'Well, it really should have...' must be met with 'Where does that appear on the test plan?'

DevNotes go both ways: clients tell us, "We need to have a user poll." I ask, "Will this poll change over time, or is this a permanent poll?" The prototype also has things that will never be seen by the real app. For instance, in a checkout process we need to be able to see what will happen if the credit card is approved or rejected. For example, after the "BillingInfo" page, a page will appear that says "Click here to simulate approved credit card" or "Click here to simulate denied credit card." That's true for any conditional logic the user needs to be aware of.

We go over and over and over with the prototype. We're done when the client says we're done. Hopefully, they listen to us if we tell them, "We're not done yet." This gives the client the needed ability to manage the risk of the project. How complete should this version of the app be? Is it more important to have all the functionality at once or to get a limited model into production?

These are decisions I want the client to make before I deliver the application to them. If I don't make sure the questions are answered at this point, they'll be answered later – when I don't want to hear the answer.

I encourage clients to take their time because once we reach prototype freeze – the point when both client and developer agree the application has been fully determined and coding is ready to begin – no more changes are possible.

Well, that's not strictly true – sometimes events occur that require changes even after prototype freeze occurs. I've found that the number and extent of the changes are far less than what I get without going through the prototype. By this point the client has agreed to the development process and understands that after prototype freeze, I get to determine (1) what changes can be allowed and (2) what price these will be. (Obviously the client can say "no" to changes.)

Clients often tend to rush through this phase. Because I want the job to be successful, I keep steering them back to the prototype, "Is it exactly what you want?" Until they can say yes with confidence, we're not done.

When we're done with the prototype, the DevNotes get turned into a set of notes that accompany it. The deliverable at the end of this design phase is the prototype and the specifications, which the client can then use to continue with the next step.

I think of development in terms of building a house – probably because this is something I did at one time. In this analogy, the prototype is the plan for a house and the DevNotes are the specifications that accompany it. This simple scheme allows builders (developers) and clients to work out the details of a project before building (coding) starts.

Charge: This depends. Safest way is hourly against a retainer. I've done fixed prices with varying success. I'd usually only offer a fixed price with a client I know well. Since most of this work doesn't involve coding, it frees up coders to do real coding and usually costs the client much less than having coders go through iterations with real code.

Implementation

Step 7. *Sales meeting*

Based on the prototype and specs, I present a fixed price bid on the job. Any agreements reached here can be easily overturned if the next step isn't satisfactory to both sides.

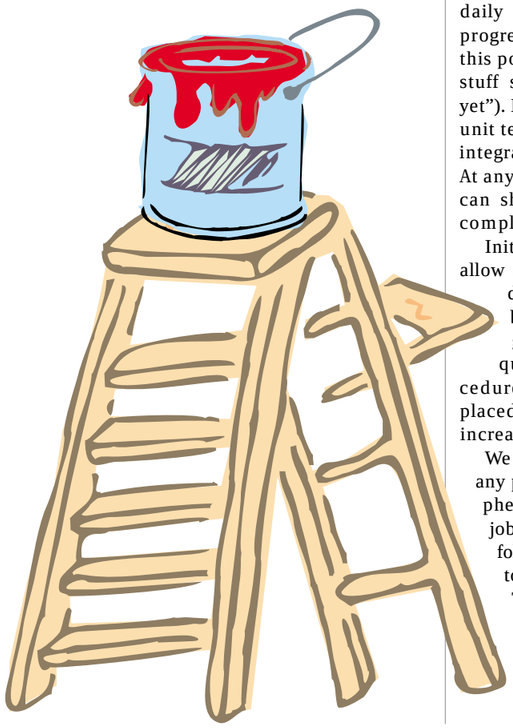
Charge: None.

Step 8. *Completion agreement*

When are we done? How will we know we're done? The most obvious answer is, "When the prototype works." Other things will probably have to be determined. "The loading portion of the program must be shown to handle 5,000 transactions per hour," for example. The important part is that we agree on what constitutes "doneness" before we begin.

It's important to agree on the staging environment in which acceptance testing is to be done. Unless the staging environment exactly mirrors the real one, you're almost assured of having unforeseen issues pop up at deployment. Sometimes an exact mirror isn't possible for a number of very legitimate reasons. In one large job I headed up, the client had almost a million dollars in hardware. It seemed a bit much to ask them to spend that again for a staging environment!

What I must have is the client's agreement that the environment they give us (or that we agree on) is the one the completion tests will be run against. Our agreement is to produce an application that does what we say it will do in the staging environment. When we're done, we print out the prototype, page by page, and both the client and I sign it, including any notes and completion test specifications. We each get a copy.



If we can't reach an agreement on this, either party can back out with no penalty. I want a really boring final deployment so I try to make sure there are no misunderstandings here. This should be a very simple part if the prototype has been done correctly.

Charge: None.

Step 9. *Architectural work*

This includes architecting the job and testing out various ideas. The client doesn't see any of this as it's all internal work. Since I use an extended Fusebox methodology, I begin identifying fuseactions and the circuits these will run in. Finally the corresponding fuses are determined, Fusedocs are written, and the prototype HTML code is included to produce fusestubs. Database work begins at this point.

Charge: Included in implementation fixed price.

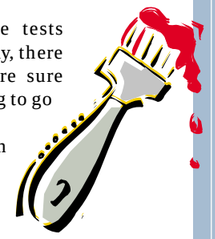
Step 10. *Coding*

The fusestubs are put up in a site that's accessible to the client. We do daily builds so all can see the progress. Nothing should break at this point (although initially a lot of stuff says, "I'm not implemented yet"). During this phase we're doing unit testing with test harnesses and integrated testing with daily builds. At any point we have something we can ship (although it may not be complete).

Initially, query sims are used to allow coding to proceed since the database has probably not been finalized. These are gradually replaced with real queries and calls to stored procedures. These daily builds are placed on the staging server, which increases the project visibility.

We can both see where we are at any point, avoiding the well-known phenomenon of the first 90% of a job being done in three months, followed by another three months to finish the remaining "10%." The problem, of course, is we never were 90% done. Daily builds help eliminate this self-deception.

Charge: Included in implementation fixed price.



Step 11. *Internal beta test*

We run through all the tests identified in Step 8. Hopefully, there are no surprises. Once we're sure we're ready, we call a meeting to go through the next step.

Charge: Included in implementation fixed price.

Step 12. *Completion test meeting*

Now we meet with the client. We take out the printed prototype pages and the test plan. The client runs the tests. Everything is judged against these docs. Any "Well, it really should have..." must be met with "Where does that appear on the test plan?" By this point, if I've done my job, clients understand and like the process. We've become partners rather than adversaries and they see me, rightly, as their representative making sure the project is successfully completed.

Once these tests are satisfactorily run, we're done and we bill the client. What about deployment to the production environment? Unless we've been able to stage the application on the production machinery in the production environment, some issues are likely to arise. Usually these are easily resolved. We normally specify in the contract a certain amount of time for deployment.

Sometimes, though, the client needs extended help due to unusual circumstances. It may be that the final configuration of hardware and software isn't working correctly, preventing us from successfully deploying our application. At that time the client will likely want us to help wade through the finger pointing of the vendors involved and achieve a solution. Since it can't be determined how long this will take, I approach this on an hourly basis.

Charge: Included in implementation fixed price.

Well, that's it. I'm not suggesting this is sacrosanct – only that it's the result of many postmortems and painful lessons. Hopefully, you'll find it a good starting point in fixing your own development process.

ABOUT THE AUTHOR

Hal Helms (www.halhelms.com) is a Team Allaire member who provides both on-site and remote training in ColdFusion and Fusebox.

HAL.HELMS@TEAMALLAIRE.COM

Point. Click. Company Calendar

An easy-to-assemble online events calendar

BY KEVIN SCHMIDT

Ever wonder how to keep your colleagues up to speed on what's happening within your organization? How best to inform them of company picnics, community events, and ColdFusion clinics?

Wouldn't it be nice to direct them to a Web site where a single resource provides all this info and more?

With ColdFusion it's easy to assemble and maintain your own online events calendar. Just think, no more paper calendars push-pinned to your cubicle wall. Now you can access up-to-date information with a simple point and click. An online events calendar can be used for many applications – tracking employee vacations, company functions, training seminars, and community events are just a few.

Why a Dynamic Calendar?

A dynamic, database-driven calendar offers many advantages over its static HTML counterpart. The first is flexibility. A dynamic calendar gives the user options. It's easy to browse over a previous month's events or jump ahead to see what's down the road. And each event can have its own page separate from the calendar, allowing users to access detailed information while the administrator maintains a single template rather than a host of static pages. This brings us to the second advantage: ease of maintenance. Users are allowed to post and maintain their own events. This saves the calendar's administrator countless time. Users can make simple changes and add events at their own leisure. But enough of the why, let's examine the how.

Event Information

The first step in developing the calendar is to decide what information you want to include. Most obvious would be the date, time, title, location, description, and a unique ID for each event. These are the most basic pieces of information, but a host of other variables can be factored in, too. Who's hosting the event? Is registration required? Is attendance free? Who does one contact for more information? These elements extend the calendar's capabilities even further, but for the purpose of this article I'll stick with only the most basic and necessary information.

I constructed an Access database consisting of these fields: EventID, EventDate, EventTime, EventTitle, EventLocation, and EventDescription. Once the database was constructed, I populated it with some dummy events so I could test the code as I went along.

The Calendar

My first goal was to generate a table that looked like a monthly calendar and had the days displayed correctly. I built the first row to display the days of the week starting with Monday. I decided to give users the option to display different months, so I needed to check whether I'm displaying the default, the current month, or a different month. Users can choose the month they want to display from a select list. The code below determines the date to be displayed:

```
<CFIF IsDefined("form.showdate")>
<CFSET DATETODISPLAY = #url.showdate#>

<CFELSE>
<CFSET DATETODISPLAY =
#DateFormat(Now(), "mm/dd/yyyy")#>
</CFIF>
```

Now that we know the date the user wants, we can set up the calendar for that month. To determine what day the first of the month falls on, we must first set up a variable that contains the date for the first of the month (i.e., mm/01/yyyy). This can be accomplished using the following code:

```
 <CFSET FIRSTDAY=" #DateFormat(DATE-
TODISPLAY,'mm')#/01/#DateFormat(DATE-
TODISPLAY, 'yyyy')# ">
```

This takes the date that was passed and uses the month and year to set up the date. Assuming that the month was September and the year was 2000, we'd end up with 09/01/2000. Once we have this date, it's necessary to determine what day of the week it falls on. This can be accomplished using the following code:

```
 <CFIF #DateFormat(FIRSTDAY, 'DDD')#
EQ "MON">
<CFSET LOOPSTART = 0>
</CFIF>
```

This code checks to see if the FIRSTDAY variable falls on a Monday. By using the "DDD" mask of the DateFormat function, the day of the week is returned as a three-letter abbreviation. Depending on what day is returned, the variable, LOOPSTART, is set. This value is used in the next step, when we generate the first week of the calendar. Another variable, DATECOUNT, is set to 1 once the LOOPSTART variable is determined. The DATECOUNT variable is used later to fill in the value of the date.

As long as the LOOPSTART variable isn't 0, it's necessary to generate some blank cells before starting the calendar. For example, if the first of the month is on a Wednesday, the cells for Monday and Tuesday need to be left blank.

The following loop generates the necessary blank cells:

```
<CFIF LOOPSTART NEQ 0>
<CFLOOP FROM="1" TO="#LOOPSTART#"
INDEX="firstweek">
 <TD ALIGN="center" WIDTH="75"
HEIGHT="75"> </TD>
  </CFLOOP>
  </CFIF>
```

For each iteration of the loop, a 75 x 75 table data cell is generated, with a nonbreaking space inserted into the cell. Once the blank cells are generated, another loop will generate and fill in the rest of the week with the appropriate dates:

```
<CFSET WEEK1 = 7 - LOOPSTART>
  <CFLOOP FROM="1" TO="#week1#"
INDEX="Loop2">
  <CFOUTPUT>
<TD ALIGN="RIGHT" VALIGN="TOP"
WIDTH="75" HEIGHT="75"<CFIF
(Month(DATETODISPLAY)) EQ
(#Month(Now())#) AND
(#DateCount#EQ(#Day(Now())#))> BGCOL-
OR="NAVY" <CFELSE>BGCOLOR="##FFFFFF"
</CFIF>>
<FONT FACE="Arial, Helvetica" size=1>
 </CFOUTPUT>

<CFOUTPUT>#DateCount#</CFOUTPUT>
 <CFSET DateCount = DateCount + 1>
 </FONT>
</TD>
  </CFLOOP>
```

The first thing to be determined is the number of cells left to be generated in the first row. By subtracting the LOOPSTART variable from 7, we're left with the number of cells to be created. The loop uses this value for its TO attribute. The loop then generates the remaining cells in the row and populates them with the DATECOUNT variable set earlier. Finally, the loop increments the DATECOUNT variable by 1. There's a conditional statement in the <TD> tag that can be used if desired, but it isn't necessary. The <CFIF> statement checks to see if the cell it's generating is for today's date. If it is, the <CFIF> statement sets the background of that cell to whatever color is specified (in this case, it's NAVY). This gives someone looking at the calendar a visual reference of the current date.

The next step is to generate the rest of the calendar. Another loop is used to accomplish this:

```
<CFSET WeekIndex = 0>
<CFSET LOOPTO = DaysInMonth(DATE-
TODISPLAY) - 1>
<CFLOOP FROM="#week1#" TO="#loopto#"
INDEX="Loop3">
<CFIF WeekIndex EQ 0><TR></CFIF>
<CFOUTPUT>
<TD ALIGN="RIGHT" VALIGN="TOP"
WIDTH="75" HEIGHT="75"
<CFIF (Month(DATETODISPLAY)) EQ
(#Month(Now())#) AND (#DateCount# EQ
(#Day(Now())#))> BGCOLOR="NAVY"
<CFELSE>BGCOLOR="##FFFFFF" </CFIF>>
<FONT FACE="Arial, Helvetica"
SIZE="1">
  </CFOUTPUT>
  <CFOUTPUT>#DateCount#</CFOUTPUT>
  </TD>
<CFSET WeekIndex = WeekIndex +
1><CFSET DateCount = DateCount + 1>
<CFIF WeekIndex EQ 7></TR><CFSET
WeekIndex = 0></CFIF>
  </CFLOOP>
```

The loop will run until the LOOPTO value has gone through all the days in the month. The value of WEEK1 contains the number of days used by the previous loop. The loop will then run through the remaining days in the month.

The variable WEEKINDEX keeps track of the day of the week the loop is currently on. If WEEKINDEX is 0, the loop includes a <TR> statement to start a new row. The default value of WEEKINDEX is 0, as this loop will always be starting a new row the first time it's run. The loop is similar to the previous one in that it generates a <TD> statement in each iteration and populates it with the DATECOUNT variable. It also increments the DATECOUNT and WEEKINDEX variables through each iteration. After the loop increments WEEKINDEX, it checks to see if its value is 7. If so, it ends the row and resets WEEKINDEX to 0. This once again signals the loop that it's time to start a new row.

After the loop has gone through all the days in the month, it's necessary to check and see if empty cells are needed at the end. If the month ends on a Wednesday, for example, the remaining days are filled with empty cells similar to the first week of the month. The following code accomplishes this task:

```
<CFIF WeekIndex NEQ 0>
<CFLOOP FROM="#WeekIndex#" TO="6"
INDEX="Loop4">
<TD ALIGN="center" WIDTH="75"
HEIGHT="75"> </TD>
</CFLOOP>
 </CFIF>
```

The <CFIF> statement checks to see if WEEKINDEX is 0. If not, it means that the previous loop finished in the middle of the week and thus the week needs to be finished. The loop uses the WEEKINDEX value to control the loop and determine how many cells to generate. The loop, like the one controlling the first week, generates 75 x 75 <TD> statements and populates them with nonbreaking spaces.

This takes care of generating the calendar. There is, however, one major piece missing: event integration.

Bring on the Events

Integrating the events into the calendar involves extending the functions of the loops that generate the calendar. The first step is selecting the desired events from the database. To streamline the calendar, it helps to select only the events that occur in the month you're displaying. To accomplish this, two

dates are needed: the first of the month and the last of the month. The first of the month is already available in the variable FIRSTDAY. The number of days in the month can be calculated in the same fashion as the LOOPTO variable from above:

```
 <CFSET DATETOADD = DaysInMonth
(DATETODISPLAY) - 1>
```

Thus, to get the last day of the month, simply add the DATETOADD value to FIRSTDAY. The code below adds DATETOADD to FIRSTDAY to generate the variable LASTDAY.

```
<CFSET ODBCLASTDAY =
#CreateODBCDate(#DateAdd('d', DATE-
TOADD, FIRSTDAY)#)#>
<CFSET ODBCFIRSTDAY =
#CreateODBCDate(FIRSTDAY)#>
```

Once we have both of these variables, we can query the database using the between clause and select only the events for the month we want. The <CFQUERY> statement below shows how:

```
 <CFQUERY DATASOURCE="yourdsn"
NAME="EventLisitng">
  SELECT *
  FROM Events
  WHERE EventDate
  BETWEEN #ODBCFIRSTDAY# AND
#ODBCLASTDAY#
 </CFQUERY>
```

Now that we've selected the events yet to occur as well as those already concluded, we can display them in the calendar. To do this, the code that generated the calendar must be modified. Now when the loops are generating the days, they also check for an event occurring on that day. If there is one, it must be displayed. To illustrate this, I'll use the loop for the first week. Let's assume the week started on a Tuesday; our LOOPSTART value would therefore be 1, and the loop would run six times. The changed code is in the shaded box:

```
<CFSET WEEK1 = 7 - LOOPSTART>
<CFLOOP FROM="1" TO="#week1#"
INDEX="Loop2">
<CFOUTPUT>
<TD ALIGN="RIGHT" VALIGN="TOP"
WIDTH="75" HEIGHT="75"<CFIF
(Month(DATETODISPLAY)) EQ
(#Month(Now())#) AND
(#DateCount#EQ(#Day(Now())#))> BGCOLOR=
"NAVY" <CFELSE>BGCOLOR="##FFFFFF"
</CFIF>>
<FONT FACE="Arial, Helvetica" size=1>
</CFOUTPUT>

<CFOUTPUT>#DateCount#</CFOUTPUT>

<CFOUTPUT QUERY="Events">
<CFIF ((DatePart("m", EventDate) EQ
DatePart("m", DATETODISPLAY)) AND
(DatePart("yyyy", EventDate) EQ
DatePart("yyyy", DATETODISPLAY)) AND
(DatePart("d", EventDate) EQ
DateCount))>
<BR>
<A
HREF="eventdetail.cfm?EventID=#Event-
ID#">EventTitle</A>
<BR><BR>
</CFIF>
</CFOUTPUT>

<CFSET DateCount = DateCount + 1>
</FONT>
</TD>
  </CFLOOP>
```

The new code added to the loop will go through the query data and check for any events falling on the date the loop is currently processing. When it finds such an event, the loop outputs a link with the Events ID as a URL variable and displays the title of the event. The same code must be inserted into the second loop as well. When users click on the event title, they're taken to the eventdetail.cfm template and shown the specifics about the selected event. The eventdetail.cfm should consist of a query to the database to retrieve the selected event and a <CFOUTPUT> to show the specifics of the event. The query and the output tags are shown here:

```
 <CFQUERY NAME="EventDetail" DATA-
SOURCE="yourdsn">
 SELECT *
 FROM Events
 WHERE EventID = #URL.EventID#
 </CFQUERY>

 <CFOUTPUT QUERY="EventDetail">
 #EventTitle#
 #EventTime#
 #EventDescription#
 </CFOUTPUT>
```

You can format the event information however you choose. And now, since we've got a working calendar, let's shift our focus to maintaining it.

Calendar Maintenance

One of the main reasons to use an online events calendar is its ease of maintenance. The calendars are generated dynamically, so there are no monthly updates to change the days, and users can enter their own events. There's plenty of flexibility when it comes to event entering, too. Events can be sent to an editor for approval before appearing on the calendar, users can update their events after they're submitted, and so forth. For the purpose of this article I'll demonstrate with a simple template that allows basic submission, with none of the extended features mentioned above. The first step in setting up the submission template is to build an HTML form with fields for each piece of information in your database.

A typical form would look like this:

```
 <FORM NAME="addevent" ACTION="add-
updateevent.cfm?addtoDB=yes"
METHOD="post">
 <INPUT TYPE="text" SIZE="25"
NAME="EventDate">
<INPUT TYPE="text" SIZE="25"
NAME="EventTime">
<INPUT TYPE="text" SIZE="25"
NAME="EventTitle">
<INPUT TYPE="text" SIZE="25"
NAME="EventLocation">
<INPUT TYPE="text" SIZE="25"
NAME="EventDescription">

<INPUT TYPE="Submit" VALUE="Add
Event">
</FORM>
```

When you build the form, there's one major issue that bears consideration: error checking. You don't want users omitting vital information or mixing up the event date with the event title. There are a couple of ways to handle this. If you use <CFFORM> to build your form, ColdFusion offers built-in error checking. Based on the different

parameters you specify, ColdFusion generates JavaScript to handle the problem. Another option is to write your own JavaScript, which is the method I prefer. I feel it offers more control and allows for cleaner, less complex JavaScript than ColdFusion generates. The final method is to use a server-side script to validate form data (which is somewhat beyond the scope of this article).

Once you've ensured that there are no omissions or mistakes in your form data, it's time to insert it into the database. To do so, use the following code:

```
<CFINSERT DATASOURCE="yourdsn" TABLE-
NAME="Events">
```

The <CFINSERT> tag here takes the values of the form fields from above and inserts them into the database. The field names in the form must match the field names in the database for this to work. If a field is passed that isn't in the database, an ODBC error will be generated. If you have fields you don't want submitted, you can use the FORMFIELDS attribute for <CFINSERT> and specify which fields you want submitted.

The user's event has now been submitted. But because mistakes can be made or an event time may change, users may want to update this information. For this example I included a link on the eventdetail.cfm template to the template used for updating events. It's the same one used to add events, but a different URL variable is passed to the template when an update needs to be made. The link from the eventdetail.cfm template looks like this:

```
<A HREF="addupdateevent.cfm?up-
date=yes&EventID=#EventID#">
```

This URL passes two variables to the addupdateevent.cfm template. By using <CFIF> statements to determine what variables were passed to the template, only the code needed to perform each operation is displayed. The first variable update lets the template know that we're updating an event. The second variable tells the template the EventID so it can retrieve the correct information. The first thing to do when updating an event is to grab the information for that event from the database. The following query will accomplish this:

```
<CFQUERY DATASOURCE="yourdsn"
NAME="update">
 SELECT *
 FROM Events
 WHERE EventID = #URL.EventID#
 </CFQUERY>
```

Now that we have our selected record, we can build the form needed for updating. This form will be nearly identical to the one used for adding an event, but with a few differences. The form can be enclosed in <CFOUTPUT> tags that specify the update query, such as <CFOUTPUT QUERY="update">. The other method is to specify the query name in the variable itself, such as #update.EventDate#. It's much easier to specify the query in the output tag in this case. The next difference is in the <FORM> tag. The URL variable must be changed to let the template know that this is an update, not the addition of a new event. It should look this:

```
<FORM NAME="update" ACTION="addupda-
teevent.cfm?updateDB=yes"
METHOD="post">
```

There are also a couple of changes required in the form fields. The first is to set up the fields to display the values returned from the database. The attribute VALUE should be added and given the ColdFusion variable corresponding to it. For example:

```
<INPUT TYPE="text" NAME="EventTitle"
SIZE="25" VALUE="#EventTitle#">
```

In addition, the EventID must be included in the form. This is passed to the template when the update is done so the template knows which record it's updating. It can be included in the form of a hidden field:

```
<INPUT TYPE="hidden" NAME="EventID"
VALUE="#EventID#">
```

> One of the main reasons to use an online events calendar is its ease of maintenance

Now the user can update the information and resubmit it, though the usual error-checking should be done to catch any mistakes or omissions. Once it's passed validation, the template will update the database. This is accomplished using a tag similar to <CFINSERT>, namely, the <CFUPDATE> tag. The syntax for the two is nearly identical. In this case:

```
<CFUPDATE DATASOURCE="yourdsn"
TABLE="Events">
```

After this is run, the event information will have been updated and will appear to anyone viewing the calendar.

Summary

An online events calendar can serve many purposes. The calendar discussed here is very basic; it illustrates simple event scheduling and maintenance, and handling the display of those events. Many pieces can be customized and extended to fit specific needs, though, so the next time your company gives you a schedule of training classes or company events, suggest they consider an online calendar instead. The source code for this application is available on the Web at coldfusionjournal.com.

ABOUT THE AUTHOR

Kevin Schmidt is an Allaire certified ColdFusion developer and is Web technology manager for PWb, an integrated marketing and information design firm in Ann Arbor, Michigan. He spends most of his time developing Web applications for clients using ColdFusion and Oracle.

SCHMIDT@PWB.COM

Database Support Is Going to Get a Whole Lot Better

CF5 is about to debut with exciting new features

BY BEN FORTA

At the Developer's Conference a few months back I was fortunate enough to introduce and first demonstrate the next major version of ColdFusion – CF5. However, time constraints prevented me from going into as much detail as I would have liked. As this issue of ***CFDJ*** is all about databases, I thought I'd take the opportunity to tell you about two very important and exciting database-related features that will make their debut in CF5.

Of course, I must point out that at the time of this writing CF5 Beta 1 has yet to be released, so some of the details discussed here could change between now and the time the product ships. Having said that, here goes….

Query-of-Queries

"Query-of-queries" (another name being tossed around is "CFSQL") is a mechanism by which ColdFusion developers can execute SQL queries against previous queries. What does this actually mean? Here are a few examples of how you'd use this feature:

- E-commerce sites often load entire product catalogs into queries within the APPLICATION scope to improve the speed of browsing catalogs and to reduce database access. This technique works well when the entire result set is needed, but accessing a specific subset requires complex looping and comparisons (often requiring going back to the database anyway). In CF5 developers will be able to execute SQL queries against that APPLICATION scope query, returning a new query result without having to hit the databases again.
- Full text searches (using the Verity engine) usually require at least two queries – one against Verity and another against the databases (using the data returned from Verity). This process needs to be repeated for each full text search; the results of one search cannot be used for a subsequent search (unless the searches were identical). Using CF5, developers would be able to perform queries against both Verity and the database and then JOIN the results allowing more powerful searches and results reuse.
- <CFPOP> is used to retrieve e-mail from a POP mailbox. <CFLDAP> is used to retrieve information from directory services. Both tags return results as queries, and those queries can be JOINed to access e-mail, along with any known information about senders and recipients.

As you can see, one of the most intriguing aspects of this new feature is that it can be used for any ColdFusion queries, not just database queries.

Here's what the code might look like for that first example. First the complete catalog is retrieved:

```
<!--- Read catalog --->
<CFIF NOT IsDefined("APPLICATION.catalog")>
 <!--- Lock the scope --->
 <CFLOCK SCOPE="APPLICATION"
TYPE="EXCLUSIVE" TIMEOUT="30">
  <!--- And execute the query --->
  <CFQUERY DATASOURCE="dsn"
NAME="APPLICATION.catalog">
   SELECT prod_id, prod_desc,
prod_price, prod_image
   FROM products
   ORDER BY prod_desc
  </CFQUERY>
 </CFLOCK>
</CFIF>
```

To perform a query against that APPLICATION scope query you could do the following:

```
<!--- Read-only lock in case being
updated --->
<CFLOCK SCOPE="APPLICATION"
TYPE="READONLY" TIMEOUT="30">
 <!--- Execute the query --->
 <CFQUERY DBTYPE="query" NAME="search">
  SELECT *
  FROM APPLICATION.catalog
  WHERE prod_desc LIKE
'%#FORM.search#%'
 </CFQUERY>
</CFLOCK>
```

In this second query the DBTYPE is specified as "query", telling ColdFusion that you're going to query a query. The SQL code then does a SELECT against the existing query using the query name as the table name. And that's all there is to it.

There are a few limitations in the SQL support – but all the basic language features (sorting, grouping, filtering, aggregate functions, aliases, and some JOIN types) are supported.

Why would you want to use this feature? It could improve performance (although not always; after all ColdFusion is not a full DBMS nor will it become one), it definitely can help alleviate server load, and it can also allow the types of data access and manipulation that aren't possible even with straight database queries.

DSNless Connections

Another exciting and much requested feature (and one not announced at the Developer's Conference) is the ability to use databases without having data sources defined. Until now ColdFusion has supported database access via data sources (ODBC, native database drivers, or OLE-DB). Database access was impossible unless that data source had been defined. As of CF5 this changes. It's now achieved by passing complete database connection strings along with the database query. Here are a few examples of when you'd want to use this feature:

- Developers working on servers that they don't have administrative access to will be able to access databases without having to wait for someone to create data sources for them.
- ISPs or IT departments can allow developers to access databases as needed without having to define data sources.
- Temporary access to databases (perhaps for an import or export operation) can be accomplished easily and efficiently.
- If you work with multiple databases within a single database you'd no longer need a data source defined for each. Instead you'd be able to use a single data source and provide the databases to be used as part of the connection string.
- Many databases support the passing of additional connection data that can't be specified in the data source configuration. This information can now be passed as part of the connection string.

Using DNSless connections requires that the DBTYPE be specified as "__DYNAMIC__" (with two underscores before and after the text). Here's an example using Microsoft SQL Server:

```
<!--- Connect to database using a
connect string --->
<CFQUERY NAME="data_import" DATASOURCE=
"__dynamic__" CONNECTSTRING="DRIVER={SQL
SERVER};SERVER=(local);uid=#SESSION.
user#;PWD=#SESSION.password#;DATA-
BASE= data_import">
 SELECT * FROM import
</CFQUERY>
```

DNSless connections are supported by all of the CFML database access tags:

- <CFGRIDUPDATE>
- <CFINSERT>
- <CFQUERY>
- <CFSTOREDPROC>
- <CFUPDATE>

And finally, queries against DNSless connections can be used like any other queries. They can be looped over, cached, have additional queries executed against them, and more.

Other Features

CF5 boasts several other important features that, while not directly database related, are worth mentioning:

- <CFGRAPH> is a new tag that allows the dynamic generation of business graphs and charts (pie charts, bar charts, etc.). <CFGRAPH> can take a query as an attribute so as to generate database-driven queries.
- The new ColdFusion Administrator features sophisticated log file analysis that can make looking for database- or SQL-related problems much easier.
- The ColdFusion Administrator features extensive integrated context-sensitive help, which should make managing database settings easier.

Summary

ColdFusion has always boasted excellent database support. After all, once upon a time that's all ColdFusion did (the language was even called DBML – Database Markup Language). As ColdFusion and the language have matured, database access and database features have improved. ColdFusion 5 continues this trend with features that will make our development even easier.

ABOUT THE AUTHOR

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit *and its sequel,* Advanced ColdFusion 4.0 Development, *as well as* Allaire Spectra E-Business Construction Kit *and* Sams Teach Yourself SQL in 10 Minutes. *He recently released* WAP Development with WML and WMLScript.

BEN@FORTA.COM

Some Thoughts on the Design of ColdFusion Data Input Applications

BY KAILASNATH AWATI AND MARIO TECHERA

ColdFusion is a powerful addition to the toolbox of Internet application developers. However, it's only one of several used in constructing a complete application. Others include relational databases with their procedural extensions, client- and server-side Java and JavaScript, HTML, and related technologies such as CSS.

Web developers need to have a good technical mastery of each of these tools, much of which can be learned from books and courses. What's harder to learn is the art of integrating these technologies – the hows and wheres of implementing application functionality. In this article we present some of our thoughts on this issue, with particular reference to ColdFusion-based data input applications.

Our firm has used ColdFusion to develop several financial data input applications that enable users to input income statements, capital expenditure reporting and budgeting, and sales reports over the Web. These applications model complicated business processes using input forms with large numbers of input elements, complex data validation schemes, and considerable back-end processing on an enterprise-strength RDBMS (DB/2, Oracle, SQL Server, etc.). Our discussion here focuses on a few recurring design issues that cropped up when we were developing these "high-volume" data input applications. The topics we cover are conveniently classified as follows:

1. Data input and validation
2. Data processing
3. Data security

Basic Landscape: Three-Tier Architecture

The starting point is the basic three-tier architecture commonly found in Internet business applications – the usual browser-Web server-database paradigm. What makes this architecture a challenge is that each tier is actually composed of several building blocks (e.g., client/server-side Java and JavaScript) that you may or may not want to integrate in your application (see Figure 1).

For concreteness we assume that the client is equipped with an up-to-date mainstream browser (IE 5.5), the middle tier includes a top-line Web server along with the ColdFusion server, and the back end has Oracle 8*i* (DB/2, Sybase, or Informix will meet the requirements just as well). Although simple on the surface (see Figure 2), this basic architecture is versatile and scalable. Keep in mind that this three-tier architecture considerably complicates the decision process regarding where to place elements of the application by multiplying the choices offered to the developer (for example, data validation can be done on the client, the ColdFusion server, or the database server).

Data Input and Validation Issues

To CFFORM or Not to CFFORM

One of the more common questions facing a ColdFusion developer is whether to use ColdFusion's enhanced form (CFFORM) or the standard HTML form. At first sight, the choice is obviously CFFORM, since it offers the following benefits:

1. A range of novel form elements including CFGRID, CFSLIDER, CFTEXTINPUT, and CFTREE. These offer the developer new functionality not available in standard HTML.
2. Enhancements to standard HTML elements such as INPUT (CFINPUT) and SELECT (CFSELECT). For the developer the most significant enhancement is built-in client- and server-side validation.

However, as we'll argue below, for high volume data input applications it's much better to use simple HTML form elements.

1. The novel form elements mentioned above are Java controls. A good network connection then becomes a prerequisite because some of the applets may require considerable time to download. Keep in mind that in many countries fast and stable Internet access is not easily available, even to corporate users. Be sure to take such users into account before deciding to go with these elements. Java applets are just fine, but if you can achieve exactly the same effect with simple HTML (albeit dynamically generated), then you should definitely do so. An example would be to replace the use of CFTREE to represent hierarchies with nested HTML lists. Response times are much better with the HTML lists, and the results can be formatted (using stylesheets) to look very elegant. We've used this in several projects in which a

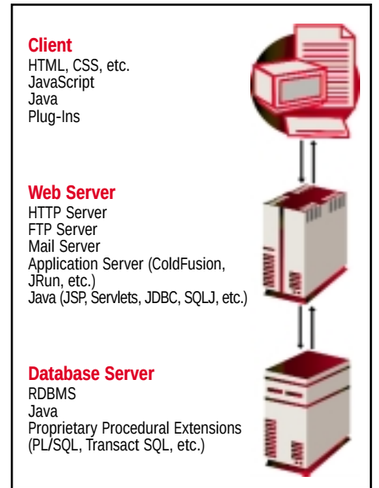


FIGURE 1: General three-tier architecture

Business presentation logic and preliminary data validation should be carried out on the client whereas all business logic should reside close to the data

www.ColdFusionJournal.com

number of users from Asia and South America were experiencing horrendous download times with the CFTREE control.
2. When client-side validation offered by CFINPUT and CFSELECT is used, ColdFusion generates separate JavaScript validation routines for each input element. If your form contains a large number of input elements, this limitation can quickly become prohibitive because of the copious volumes of code generated. For this reason we prefer to custom write our client validation code and avoid the use of CFFORM elements altogether.

Layout and Presentation

With the advent of cascading stylesheets (CSS), ColdFusion developers now have a good level of control on input form layout and presentation. We urge you to explore some of the things stylesheets make possible. For example, Listing 1 shows how a hyperlink can masquerade as a button using CSS (why you might want to do this is another matter!). Note that the demo has been tested only on Microsoft Internet Explorer 4.0+, which brings us to a word of caution: browsers vary considerably in support of CSS, so use the more arcane features only after checking that they're supported by your target browsers.

Another issue, often overlooked, is the level of control that CSS offers in the look and feel of form elements. It allows you to control font, width, size, color, and border. Here's a customized input box for numerical data input that illustrates the use of some of these CSS attributes:

```
<input type = "Text"
       size="15"
              maxlength="12"
       style="font-size: xx-small; font-
family: Arial; text-align: right;">
```

Note that CSS gives you an easy way to right-align the number instead of living with the default left alignment. Further, restrictions on data length can be enforced without a single line of JavaScript through use of the (non-CSS) "maxlength" attribute.

A frequently encountered situation for financial data input applications is that you have a complex grid of form elements that need to be entered. A possible business context is when yearly budgeting of a chart of accounts is done on a monthly basis: 12 months running on the top of the page along with say 50 accounts running down the left side of the page make for 600 input form fields. How you implement this in ColdFusion depends on how your database tables are organized. Are the months represented as separate columns or as a single column? If it's the former, a simple CFQUERY (ordered by account) followed by a CFOUTPUT will do the job. However, in the latter, your query will have to order the data by account and month, then use the "group" attribute of CFOUTPUT to ensure the form elements are placed correctly.

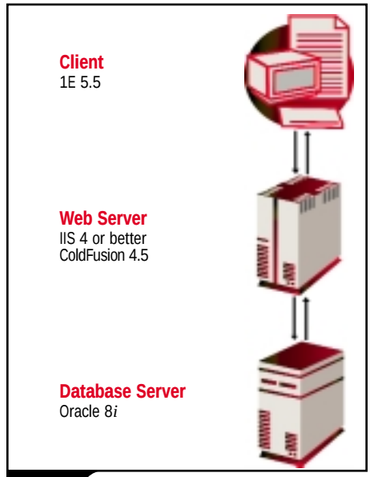


FIGURE 2: Three-tier architecture assumed in text

Staying with the above example, the end users now ask you to enable the up/down keys on their keyboards to allow for rapid navigation within the grid. The catch here is that the form elements don't have any simple way of being accessed with a two-dimensional index. It's only one form and each element is numbered sequentially. The two-dimensional grid is an illusion of the HTML layout. The only way to do this is to use JavaScript and implement a counting scheme after catching the "onkeydown" event. The number of the form element is coded in the element name. The code is presented in Listing 2 (Microsoft Internet Explorer 4+ only). In the more general case when the number of rows varies depending on data coming from the database, the ColdFusion server can be made to generate the form names dynamically so you can easily implement the counting scheme. This nontrivial piece of presentation logic is handled strictly by the Web browser.

Data Validation (Client Side)

There are several languages available for client-side validation (JavaScript and VBScript, for example). However, as JavaScript (or, more properly, ECMAScript) is an accepted standard, we'll stick with it in this discussion.

Despite standardization efforts, there are considerable differences in implementation for different browsers. Further, with new browser versions coming up for release periodically, there's a virtual zoo of JavaScript implementations out there. Our only advice is to take the lowest common denominator wherever possible, and to use browser detection codes if you need to use vendor-specific features. Check out www.javascripts.com for browser-detection code.

The nicest feature of client-side JavaScript validation is that it prevents user aggravation by avoiding unnecessary trips to the server. However, the downside is that clients can disable these features, so you shouldn't count on them being there. Consequently, you should always implement a second line of defense – data validation on the server (see below). Although client-side validation can be used to perform complex validations, it's best suited to simple checks, for example, on data types or formats. This is, in fact, what the JavaScript generated by CFFORM elements does. However, for reasons mentioned above, it's best to avoid using these.

It's difficult to give any general rules regarding the implementation of client-side validation routines for high-volume data input applications, as each case is unique. Further, it's usually best to see such routines in action, so we'll say no more about this here and refer our readers to www.orafusion.com/js.htm for a look at some JavaScript demos taken from our projects.

Data Validation (Server Side)

Server-side validation offers the most flexible and surefire method of checking user inputs. These checks can be carried out on the ColdFusion server using CFML constructs or, better yet, on the database via data type definitions, check constraints, and database triggers. Any of these three methods can be used to return database exceptions that can be handled using ColdFusion's error handling tags (CFTRY and CFCATCH). We believe that all server-side validation should be done on the database instead of the ColdFusion server. ColdFusion should serve merely as a conduit for transporting information (data, exceptions, messages) between the database and the client, or for carrying out presentation-related procedural manipulations. The ColdFusion server is definitely not the place to do any complex manipulation of data (validation included).

There are essentially three methods of data validation on the database server:

1. Data Type Validation

Data type validation is one of the most basic server-side validations available. ANSI has gone through considerable pain to define some basic SQL data types that are supported by all major enterprise RDBMSs. Whenever a table is created, each column must be assigned a unique data type. The database server will then throw an error whenever a column data type is violated. Data types by themselves tell only part of the story. The other part is told by the data type comparison semantics that define how the RDBMS allows you to compare data, even across data types in some cases. Finally, but also important in the context of data types, are user-defined data types. Because most RDBMSs now also offer object extensions, it's possible to define your own data type using the CREATE-TYPE statement. Using such basic validation you can guarantee that, for example, a product code is numerical, not alphanumerical. This guarantee is inherent to your data structure and cannot be violated, regardless of the application accessing your database. Very powerful indeed!

2. Check Constraints

Check constraints are an extremely effective and largely underutilized method of data validation. These constraints are built into table definitions and are consequently very efficient. They allow the developer to perform relatively complicated checks right where the data resides. Here are a few simple examples.

If you're storing the month for a given record in numerical form in a table, you can use the constraint:

```
CHECK (MONTH IN
[1,2,3,4,5,6,7,8,9,10,11,12]).
```

Similarly if you want to guarantee that the year is within a given range, say 1980 to 2010, use the check constraint:

```
CHECK (YEAR BETWEEN 1980 AND 2010).
```

Clearly these checks can and should also be performed on the client via JavaScript, because they're simple (quickly implemented) and can save two sets of round-trips, thus improving performance (browser-Web server and Web server-database server). The advantage of having the check in your database is that you're guaranteed that *all* applications accessing the data must conform to the business rule, not just your particular ColdFusion application.

3. Triggers

A trigger is a block of procedural code executed when a certain event that modifies data in a table is performed (INSERT, UPDATE, or DELETE). For example, if you're entering sales data for a given product in a table, you might have a trigger to first check that the product being referenced is still active. If the product is not active, you may want to raise an exception and let the users know they've somehow managed to select the wrong product. This again is done right where the data resides. On the other hand, if you filter out the active products at the level of your ColdFusion server, you're prone to data integrity violations if someone accesses the data through other tools or applications.

Until very recently these blocks of code had to be written in a procedural extension to SQL. These extensions are RDBMS specific and proprietary, examples being PL/SQL or Transact SQL. Some exciting new developments have been taking place in the database industry (supported by corresponding standards) that should open the eyes of corporate Web developers. An important one is that most major database companies (Microsoft being the exception!) allow Java code to be executed directly within the database. A trigger is only one example of where this can be done. This means you can apply any Java programming skills to developing RDBMS code blocks that implement business process logic without having to use proprietary RDBMS languages. If that's not enough, a new interface called SQLJ has been accepted as part of the Java standard (ANSI & ISO); this allows you to basically write inline SQL statements in your Java procedures, making Java and SQL appear as if they were originally conceived together. Not bad!

You can reference several introductory articles on Java and databases in ***Java Developer's Journal***. We'll also be talking a bit more about these technologies in an upcoming series of articles on Oracle and ColdFusion.

Data Processing

Once your data has been entered in the HTML form you'll want to process it. A typical example from one of our applications is that users input data (monetary values) in their local country currency and the application then converts these values into several standard corporate currencies for display and comparison purposes. There are many ways of performing such tasks, but our experience has been that, by far, the best is to place such processing of data directly on the back-end tier using stored procedures. The implementation of Java in the mainstream RDBMSs now opens up the possibility of writing code that's independent of the specific database being used. This could make Java (with SQLJ) the language of choice for implementing stored procedures.

Database procedures are generally accessed via the <CFSTOREDPROC> tag. If the procedure runs quickly (say a few seconds), we usually call the procedure directly, wait for the output, and then return any feedback directly to the user. When the procedure requires more than just a couple of seconds we usually invoke an intermediate procedure, its only function being to toss the main procedure on a job queue within the database. Details of job queue implementation depend on the specific DBMS used; we'll talk about how this is done in Oracle in one of the articles in our upcoming series. After the queued procedure runs to completion, the end user is sent feedback via an e-mail message automatically generated using <CFMAIL> and the ColdFusion scheduler. This mail usually contains the duration of the calculations, time of completion, and a summarized view of the results (for example, total income and total expenses when the user was entering income statement information).

Data Security

Finally we'd like to discuss some efficient ways of ensuring data security, a major issue in the corporate world, especially when an application is to be placed on the Internet. As we'll see, this is best done in the database.

First, each user should gain entry to the application via a database login and password. Further, the database should be located on a separate machine (normally behind the corporate firewall) from the ColdFusion/Web server. This ensures a degree of security beyond that gained by access through operating system usernames/passwords.

Second, once users are validated against the database, we need to ensure that they have controlled access to the data (i.e., they should be able to see only what they're allowed to see). Consider a simple scenario with product managers and products of a large consumer goods corporation. Each product is assigned to only one product manager who enters the yearly sales figure for the product through a ColdFusion application. The corporation, however, has a policy of strict confidentiality and asks that project managers can see and modify only their assigned products. The tables to model this process might look like Tables 1 and 2.

CREATE TABLE PRODUCT_MANAGER (	
PM_ID	NUMBER,
PM_NAME	VARCHAR (300),
USER_LOGIN	VARCHAR (10))

TABLE 1 Product manager

CREATE TABLE YEARLY_PRODUCT_SALES (	
PRODUCT_ID	NUMBER,
PRODUCT_NAME	VARCHAR (100),
YEAR	NUMBER,
TOTAL SALES	NUMBER (16, 2),
PM_ID	NUMBER)

TABLE 2 Yearly product sales

What's needed is an infrastructure that will give you row-level control on the YEARLY_PRODUCT_SALES table. You can implement the requested security policy on the ColdFusion server, but it's very cumbersome and if the product manager accesses the database through another tool such as Excel, all data will be visible.

The solution to this problem is to simply place a view on top of the table (Oracle syntax) YEARLY_PRODUCT_SALES (see Table 3).

```
CREATE VIEW V_ YEARLY_PRODUCT_SALES (
        PRODUCT_ID,
        PRODUCT_NAME,
        YEAR,
        TOTAL_SALES,
        PM_ID
      AS
      SELECT
        T2.PRODUCT_ID,
        T2.PRODUCT_NAME,
        T2.YEAR,
        T2.TOTAL_SALES,
        T2.PM_ID)
      FROM
        YEARLY_PRODUCT_SALES T1,
        PRODUCT_MANAGER T2
      WHERE
        T1.PM_ID = T2.PM_ID AND
        T2.USER_LOGIN = USER
```

TABLE 3 **Yearly product sales with view**

The simple join guarantees that the product managers have access to only the assigned products. Your ColdFusion application should never access the table directly, only the view. In fact, your users need not have any knowledge about the underlying data table. This mechanism is effective and easy to implement. Once created in the database you never need worry about it again when developing your application, as long as you access all data through the view.

Here's a peek at the future: beginning with version 8*i*, Oracle introduced the concept of CONTEXTS, which allow you to do the same thing as we displayed above but within a much more general framework. This new feature also allows you to leverage your Java knowledge in the database. Without going into the details here we definitely suggest you look at this functionality. It's another of the topics we intend to explore in our upcoming Oracle series.

Conclusion

Application development is certainly a challenge when working with high-pressure deadlines in a corporate environment. In fact, that's why a lot of developers use ColdFusion. It's important to keep in mind, though, that in a three-tier environment you also have other tools at your disposal, some of which may be more appropriate for the tasks at hand. A good case can be made for putting business-process logic in the database along with the data instead of the Web/ColdFusion server. Modern RDBMSs are extremely powerful, versatile, and probably the most scalable piece of enterprise software available aside from operating systems. We believe this can be summarized into a simple, yet powerful, generalization: *business presentation logic and preliminary data validation should be carried out as much as possible on the client whereas all business process logic should reside as closely as possible to the data itself.*

This is a tug-of-war that every Internet application must confront with the ColdFusion server sitting right in the middle. This perspective has allowed us to successfully develop complex systems for large numbers of concurrent users in relatively short periods of time. The resulting applications have also remained flexible without falling victim to "Scope Creep Fatigue" (we're sure you've heard some of your users say "Just one more little change…") – the point beyond which applications turn into maintenance nightmares.

About the Authors

Kailasnath Awati is senior consultant at Williams & Partner Management Consulting where he manages the U.S. operations. He is an experienced Internet/database application developer and architect as well as an Allaire certified ColdFusion developer. His personal Web site is www.orafusion.com.

Mario Techera specializes in Internet application design and development with relational databases. He is one of the founding partners of Williams & Partner Management Consulting and works out of Munich, Germany.

kawati@wpmc.com

mtechera@wpmc.com

Listing 1

```
<!---CSS Toolbar--->

<html>
<body>
   <table border="0" cellpadding="3" cellspacing="9"
align="center">
      <tr>
         <td style="background-color: navy; padding: 2px;
width: 100px; text-align: center;border-style: outset;">
            <a href="default.htm"
               style="font-family: verdana; font-size:
11pt;
                  font-weight: bold;color: white; text-dec-
oration: none; width: 100px;"
               onmouseover="this.style.color='powderblue';"
               onmouseout="this.style.color='white';">
               Home
            </a>
         </td>
         <td style="background-color: navy; padding: 3px;
width: 100px; text-align: center;border-style: outset;">
            <a href="info.htm"
               style="font-family: verdana; font-size:11pt;
                  font-weight: bold;color: white; text-dec-
oration: none;"
               onmouseover="this.style.color='powderblue';"
               onmouseout="this.style.color='white';">
               Information
            </a>
         </td>
         <td style="background-color: navy; padding: 3px;
width: 100px; text-align: center;border-style: outset;">
            <a href="downloads.htm"
               style="font-family: verdana; font-size:
11pt;
                  font-weight: bold;color: white; text-dec-
oration: none;"
               onmouseover="this.style.color='powderblue';"
               onmouseout="this.style.color='white';">
               Downloads
            </a>
         </td>
      </tr>
   </table>
</body>
</html>
```

Listing 2

```
<!---Example of arrow up and down functionality using
JavaScript (Internet Explorer only!)--->
<html>
<script>
   function moveIt(elt,ncols){
      //Move the cursor

      var form=elt.form
      var key=event.keyCode;
      var index
      index=indx(elt);

      if (key==38){
         if(index>ncols)
            form.elements[index-ncols-1].select();
      }
      else if (key==40){
      if(index<form.elements.length-ncols+1)
            form.elements[index+ncols-1].select();
      }
      else if (key==37){
         if(index>1)
            form.elements[index-2].select();
      }
      else if (key==39){
         if(index<form.elements.length)
            form.elements[index].select();
      }

   }

   function indx(elt){
      //get element number from element name
      var s=elt.name;
      return parseInt(s.split('_')[1])
   }
</script>
<body>
<form name="demo">
   <table>
      <tr>
         <td>
            <input type="Text"
                name="element_1"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
         <td>
            <input type="Text"
                name="element_2"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
        <td>
            <input type="Text"
                name="element_3"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
      </tr>
      <tr>
         <td>
            <input type="Text"
                name="element_4"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
         <td>
            <input type="Text"
                name="element_5"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
        <td>
            <input type="Text"
                name="element_6"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
      </tr>
      <tr>
         <td>
            <input type="Text"
                name="element_7"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
         <td>
            <input type="Text"
                name="element_8"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
 <td>
            <input type="Text"
                name="element_9"
               align="RIGHT"
               onkeydown="moveIt(this,3)"
               size="6">
         </td>
      </tr>
   </table>
</form>
</body>
</html>
```

CODE LISTING

The code listing for this article can also be located at www.ColdFusionJournal.com

Five Ways to Call Client- and Server-Side Java from ColdFusion

You can do it now

BY ANDREW CRIPPS

In 1995 Allaire launched the ColdFusion Web Application Server. Since that launch, Java has become an important language for the Web, and Allaire has moved right along to combine the power of Java with ColdFusion.

On June 15, 1999, Allaire purchased JRun, giving them a powerful Java server that currently supports Java servlets and Enterprise JavaBeans.

Allaire has a technology road map that will allow your CFML pages to execute directly on the next generation application server technology – a Java engine code named *Pharaoh*.

Right now you can develop applications in Java and run them with your JRun engine, and develop ColdFusion applications in CFML and run them on the ColdFusion Web Application Server. But how do you use ColdFusion and Java together?

Jeremy Allaire, in his **IMHO**, "Allaire ColdFusion, Java and JRun" (***CFDJ***, Vol. 1, issue 4), said that "ColdFusion and Java are perfect cousins." In this article I want to show how you can use Java with your ColdFusion applications right now. The good news is that you don't have to wait for Pharaoh or switch to JRun.

Java on the Server and the Client

When Java first came out, many people thought it was going to be primarily useful on the client. Browser manufacturers started including Java Virtual Machines (JVM) in their products, and Web sites started putting Java applets in Web pages. The browser downloads the applet, and the applet runs on the client machine. As Java has matured, it's come to be seen more as a server-side language than a client-side language. On the server, software systems can be built from a combination of Java servlets, Java classes and/or applications, and Enterprise JavaBeans. You can make use of both client- and server-side Java with ColdFusion.

There are five methods for calling Java (both server- and client-side) from ColdFusion:

1. Calling Java servlets using CFSERVLET
2. Calling Java servlets using CFHTTP
3. Calling and creating Java components with CFOBJECT
4. Calling Java applets with CFAPPLET
5. Calling Java classes with CFX tags

Calling Servlets with CFSERVLET

You can call Java servlets from your ColdFusion pages using the CFSERVLET tag. This tag executes a servlet on the JRun engine. Note that in order to make use of CFSERVLET, the machine on which the servlet exists needs to have JRun installed. You don't need JRun on the same server as Cold-Fusion. You can, of course, make remote calls to other machines (both at your site and over the Internet) as long as the remote machine is using JRun as its servlet engine (see Figure 1).

Let's say you want to call a servlet-based Web site that's using JRun. You can call any of the servlets on that machine with CFSERVLET. All you have to do is specify the JRUNPROXY attribute to be the IP address of the server running JRun.

Depending on your Web site, you might have JRun and Cold-Fusion on the same server or on the same network.

Sometimes you need to specify parameters for the servlet you're calling. To do this, specify CFSERVLETPARAM tags within the CFSERVLET call.

For example:

```
<CFSERVLET CODE="ServletName"
 JRUNPROXY="233.125.25.3:8083">
 <CFSERVLETPARAM name="param1"
value="35">
</CFSERVLET>
```

Note that when your ColdFusion page executes the CFSERVLET tag, it will wait for a response from the servlet you have called before continuing. The servlet you've called is executing on the JVM in the JRun engine.

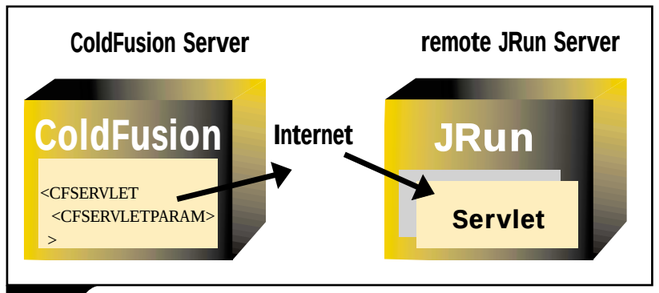


FIGURE 1: Components needed to make a CFSERVLET call

Calling Servlets with CFHTTP

Sometimes you may need to call Java servlets hosted on a Web server that's not using JRun as its servlet engine. You can do this using CFHTTP. Simply include the CFHTTP tag in your CFML page and make the URL you call point to the servlet you want to execute. If the servlet you're calling requires form input parameters, you'll need to specify the METHOD="POST" at-tribute in CFHTTP and include CHTTPPARAM tags for the form variables that your servlet requires (see Listing 1).

Calling Java Components with CFOBJECT

CFOBJECT provides a flexible way to call Java components. You can create and use Java objects and Enterprise JavaBeans using CFOBJECT. (You need ColdFusion 4.5 or later to call Java components this way.)

Setup in ColdFusion Administrator

To use CFOBJECT (and Java CFXs) you need to set up the Java section of the ColdFusion Administrator. Example settings are as follows:

- ***Load JVM when starting ColdFusion:*** If you want to load the JVM in the ColdFusion process every time ColdFusion starts, check this box. If you want to load only the JVM when a Java request is made, leave it unchecked.
- ***Java Virtual Machine path:*** This is the path to your JVM. For example, on Windows, it might be: "c:\jdk1.2.2\jre\bin\classic\jvm.dll".
- ***Classpath:*** Your classpath should contain the paths of any Java classes you want to use with CFOBJECT. For example:

```
"c:;c:\jdk1.2.2;c:\jdk1.3\;c:\xerces\
xerces.jar; c:\weblogic\lib\weblogic-
aux.jar;c:\weblogic\classes".
```

- ***Initial heap size:*** Specifies the heap size that's allocated to the JVM when the JVM starts (typically 1K).
- ***Maximum heap size:*** Specifies the maximum size for the heap assigned to the JVM (typically 16K).
- ***System options:*** Specify any JVM command-line options here (name-value pairs separated by a semicolon).
- ***Implementation options:*** Specify any implementation-specific options here (name-value pairs separated by a semicolon).
- ***CFX JAR path:*** Specify the location of the ColdFusion cfx.jar file that contains interfaces used by Java CFXs. For example:

```
"C:\CFUSION\JAVA\CLASSES"
```

Any Java class in the classpath specified in the ColdFusion Administrator can be loaded and used by ColdFusion. Note that classes execute on the JVM embedded in the ColdFusion process.

Your ColdFusion Administrator must also enable the CFOBJECT tag in the security section before you can use either CFOBJECT or the CreateObject function.

Calling Enterprise JavaBeans

You can also use CFOBJECT to call EJBs. EJBs can be deployed in any EJB server, such as JRun or BEA's WebLogic Server.

To call EJBs, first configure the ColdFusion Administrator, as described above; second, create a JavaBean and deploy it to your EJB server; third, create a Java object that's running on the ColdFusion JVM, and then use that object to call your EJBs running on the EJB server. The results of the call are returned as a ColdFusion variable you can use in your CFML template.

You can create Java objects in two ways:

- Using the <CFOBJECTACTION="-CREATE" ..> tag,
- Using the function CreateObject-("JAVA", class)

Note that CFOBJECT supports COM, Java, and CORBA objects. Solaris installations of ColdFusion support only CORBA objects at the time of writing, while Windows NT installations of ColdFusion support all three types.

For example, suppose you have written an entity bean that looks up an employee name given a unique key. In this example, we'll assume the bean has been deployed to BEA's WebLogic Server.

We now need a ColdFusion page that will call our entity bean (see Listing 2).

Calling Java Applets with CFAPPLET

Client-Side Java

Java applets are supported by ColdFusion using the CFAPPLET tag. Applets are stored on your server. They're downloaded by the client whenever a page that contains applets is requested from the server. You can write your own applets or include existing applets in your Web pages. The advantage of registering applets in the Administrator and using CFAPPLET to invoke them is that you don't have to specify all the parameters the applet requires each time you use it. CFAPPLET will use the defaults you've registered in the Administrator unless you specifically override them.

Using Java Applets

To use a Java applet in your CFML page, register the applet with the ColdFusion Administrator applets section. Once it's registered you can embed the applet in your CFML page using CFAPPLET. CFAPPLET must be used with CFFORM.

ColdFusion Administrator Settings

The applet section of the Administrator has the following settings.

- ***AppletName:*** The name you'll use when you call the applet with CFAPPLET.
- ***Codebase:*** The URL (from the server root) to the directory containing the applet class file (e.g., /test/servlets/).
- ***Code:*** The name of the applet class file (you should include the .class extension; e.g., "myapplet.class").
- ***Archive:*** Optional. This attribute is optional and simply passed through by CFAPPLET to the applet. The actual meaning of the archive attribute varies between the HTML 4.0 specification and Netscape's implementation. It's not supported by Internet Explorer. For Netscape, you can specify an archive (.zip or .jar) that packages multiple class files into a single archive. This makes it quicker for the browser to load all the required classes.

In HTML 4.0, the archive attribute allows you to specify a space-delimited list of URLs for additional classes or resource files. This is for compati-

bility with the OBJECT element, and allows required resources to be placed in several locations.

- ***Method:*** Optional. The name of a method of the applet that returns a value. Most applets don't return values. If you're not sure, leave it blank.
- ***Height:*** Default height of the applet on the Web page in pixels.
- ***Width:*** Default width of the applet on the Web page in pixels.
- ***VSpace:*** Optional.
- ***HSpace:*** Optional.
- ***Align:*** Determines how the area defined for the applet is aligned with content on the Web page (e.g., "ABSMIDDLE").
- ***Java Not Supported Message:*** The message you want the applet to display if the browser rendering the Web page does not support applets.

The CFAPPLET Tag

The CFAPPLET tag is simple to use. You specify the following attributes:

- ***Appletsource:*** Required. The value is the name you gave to your applet when you registered it in the ColdFusion Administrator.
- ***Name:*** Required. Specifies the name of the Form variable. If you specified a value for Method in the ColdFusion Administrator when registering the applet, this name should match the method name. This Form variable will contain the result of the method called in the applet.
- ***Height, Width, VSpace, HSpace, Align, Not Supported:*** These attributes override the same settings in the ColdFusion Administrator.
- ***Applet parameters:*** Most applets require parameters, and will have default values defined so they'll run even if you don't specify any parameters. You can set up your own default values for the applet in the parameters section when you register the applet in the ColdFusion Administrator only if you want to override the defaults you have specified (or if you didn't specify any defaults and want to override the applet's own defaults).

Example applet call:

```
<CFAPPLET APPLETSOURCE="test"
name="bob">
```

Calling Java Classes with CFX Tags

CFXs – or ColdFusion Extensions – allow you to extend ColdFusion with custom code written in C++ or Java. You can write your own CFXs in Java or just call existing Java classes.

To call a Java class this way, you need to do two things in the ColdFusion Administrator:

1. In the Java section set up the classpath and other information as described in the section above on CFOBJECT.
2. Register your custom extensions in the ColdFusion Administrator in the CFX tags section. Give the CFX a name, specify that it's a Java extension, and then tell ColdFusion which Java class the CFX will call.

For example, suppose you've registered a class "HelloWorld" in the ColdFusion Administrator CFX section that outputs "Hello Andrew" when it's called (see Listing 3).

In the Java section, you've included the path to your HelloWorld.class file in the classpath option, and now you want to create a sample ColdFusion page (see Listing 4).

You should see "HelloWorld" as the output when you call this page.

> CFXs – or ColdFusion Extensions – allow you to extend ColdFusion with custom code written in C++ or Java

If you get an error like this:

```
java.lang.ClassNotFoundException:
HelloColdFusion. Java exception
occurred in call to method
```

it's likely that you haven't correctly specified the path to your Java class in the classpath setting of the Java section of ColdFusion Administrator. Another possibility is that you have not correctly specified the classname in the CFX Class attribute in the CFX section of the ColdFusion Administrator.

Note that if you're using ColdFusion Server version 4 or earlier, you need to use CFX_J. CFX_J has now been replaced with the CFX mechanism described above. But you can still download and install CFX_J if you have an older version of ColdFusion Server.

Summary

In this article we've covered five ways to call both client- and server-side Java from ColdFusion. If you need to call JRun servlets, use CFSERVLET; if you need to make a simple call to Java class, use CFXs; and if you need to call EJBs, use CFOBJECT .

ABOUT THE AUTHOR

Andrew Cripps is manager and technical lead at evolutionB (www.evolutionb.com), currently on a project using Internet and Web technologies. He has over 12 years of experience in the computing industry, with master's degrees in both computing science and philosophy.

ANDREW@CRIPPS.NET

Listing 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<html>
<head>
<title>Test CFHTTP call</title>
</head>
<body>

<cfhttp
url="http://ippc.nais.luftfartsverket.no/PIBListServ-
let" method="post">
<cfhttpparam     name="RetrievePIBList"
 value="Retrieve List"
 type="FORMFIELD">
</cfhttp>

<Cfoutput>#cfhttp.filecontent#</cfoutput>

</body>
</html>
```

Listing 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<!---    Andrew Cripps, Sept. 2000          --->
<!---    Create a weblogic environment object that
allows us to
 look up the bean ìEmployeeHomeî
--->
<CFOBJECT ACTION="Create"
         TYPE="Java"
         CLASS="weblogic/jndi/Environment"
         NAME="wlEnv">
<!--- Get the context --->
<CFSET ctx=wlEnv.getInitialContext()>
<!--- Use the lookup method of the context --->
<CFSET ejbHome=ctx.lookup("EmployeeHome")>

<!--- accepts ID as input and calls EJB component
to do lookup --->
<cfparam name="form.id" default="1">

<!--- Call the findByEmployeeID method --->
<cfset temp =
ejbHome.findByEmployeeID("#form.id#")>
<!--- Call the getEmployeeName method--->
<cfset value = temp.getEmployeeName()>
<!--- Output the results--->
<html>
<head>
<title>form post processor</title>
</head>

<body>
<h1>Calls Employee EJB</h1>
<cfoutput>
Value from EmployeeBean is <b>#value#</b>
</cfoutput>
</body>
</html>
```

Listing 3

```
/* java class to print Hello World */
import com.allaire.cfx.* ;

public class HelloWorld implements CustomTag
{
   public void processRequest( Request request,
Response response )
       throws Exception
   {
      String strName = request.getAttribute( "NAME" ) ;
       response.write( "Hello, " + strName ) ;
   }
}
```

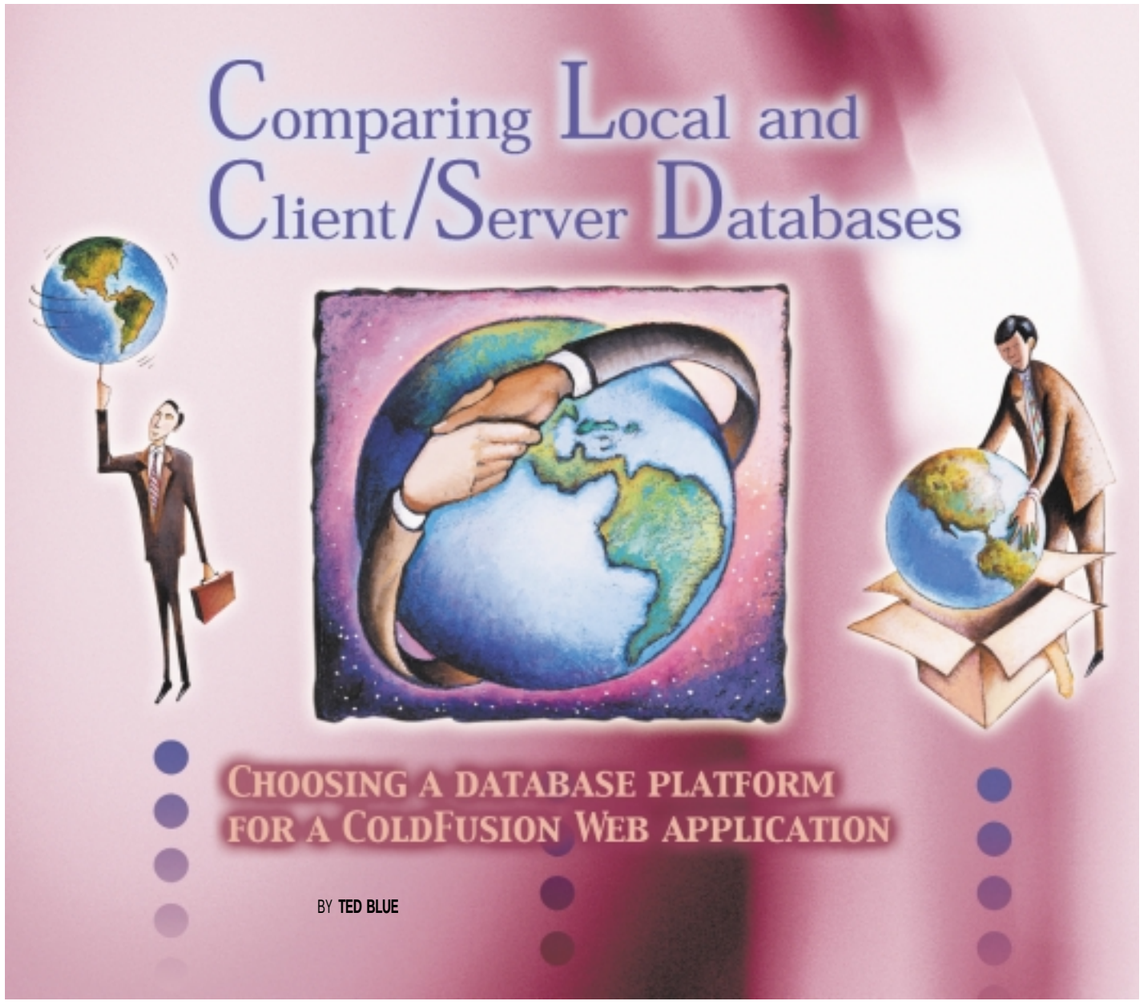
Listing 4

```
<html>
<head>
<title>Call CFX HelloWorld</title>
</head>

<body>
<h1>Calls CFX_HelloWorld</h1>
<CFX_HelloWorld NAME=ìAndrewî>
</body>
</html>
```

CODE LISTING

The code listing for this article can also be located at www.ColdFusionJournal.com



When creating a ColdFusion application, one of the first and most fundamental choices that must be made is which database platform to use. Each platform offers its own unique capabilities. Basically, there are two major categories of databases to choose from: local, such as Microsoft Access, dBASE, and FoxPro, and client/server, such as Oracle, SQL Server, and Sybase.

Don't kid yourself, this is not a trivial decision. Changing database platforms after development has begun can be difficult, and it becomes even more so the further along you get. There's a point of no return for most complex development projects where it's no longer cost-effective to change the database. At this point a major rewrite would be in order. In addition, some features your Web site might require may or may not be supported by the underlying database platform.

This article explores the fundamental differences between local and client/server databases, and the issues that often arise when choosing a database platform for a ColdFusion Web application. For discussion purposes we'll assume the ColdFusion application is for e-commerce and the site will be heavily trafficked.

Web Applications Are Client/Server

All Web applications operate in a client/server environment (see Figure 1). A Web browser (a client) sends a request to a Web server, which is relayed to the ColdFusion Application Server. The server generates the appropriate HTML code based on the user's request, which is usually based on data that's accessible to the Web application. The HTML is then relayed by the Web server back to the client (the browser).

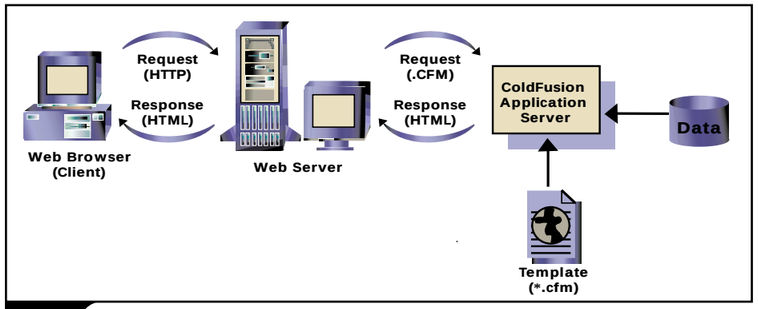


FIGURE 1: Client/server interaction in a ColdFusion application

In Figure 1 there are two separate client/server relationships: between the Web browser and the Web server where the browser is the client, and between the Web server and the ColdFusion Application Server where the Web server is the client. A third client/server relationship may also exist between the ColdFusion Application Server and the database server if the data is stored on a database management system (DBMS). We'll discuss this in more detail in a later section.

Local Databases

Let's start the discussion with local databases. Technically, a local database is any data stored in an unmanaged database format – that is, a database that doesn't have or require dedicated database management software to access its data. Examples of local database formats include Microsoft Access, dBASE, FoxPro, and Paradox; other less-used formats are ISAM, B-Tree, and Delimited Text.

Connecting to an unmanaged, local data format requires that the appropriate ODBC driver is set up and configured correctly on the Web server and any essential passwords are specified. The database files can be located elsewhere. Assuming the ODBC driver is configured correctly, no additional preparation is needed – you can access the data using appropriate ColdFusion techniques, such as the CFQUERY tag.

One of the most attractive features of using a local database format is cost. The only real cost associated with a local database format is the tool you would use to create and manage the data files. For example, you might purchase a copy of Microsoft Access for creating and managing your .MDB files. In some cases the data format has no inherent cost, as the file format is commonly used by a number of other tools. The dBASE file format, for instance, can be easily created from an Excel spreadsheet or from a variety of other software applications. In almost all cases there are no licensing fees or royalties for using these database formats.

Limitations of Local Databases

Although the configuration and setup of a local database is simple and the cost is low, certain limitations must be taken into consideration before dedicating any development to a local database format. First there's the issue of scalability. Many local database formats can't handle large numbers of records or simultaneous users accessing the data, both of which are critical to the development of a typical transaction-based e-commerce application.

For example, consider a Web site based on the traditional dBASE (.DBF) file format. Although it's possible to store a fair number of records in a dBASE table (100–500KB records isn't uncommon), the dBASE file format wasn't designed to hold huge amounts of data. There are known limits to the various versions of the dBASE file format, such as a 16MB limit on data stored in a memo field, due to the size of the data pointer. However, the problem is not so much with the amount of data stored as with the attempt to access that data using SQL through an ODBC driver. As the size of the tables increases, the performance decreases, often dramatically.

This limitation isn't specific to the dBASE format – most local database formats have performance problems when they exceed certain basic size limitations. In all cases, however, the larger the local database, the poorer the performance. The actual size limit where the performance problem occurs depends on a number of factors. For some local databases the limitation is measured in the number of records. For others, it's the total size of the database, the width of the record, or some other combination.

Performance Under Load

There are also limitations on the number of concurrent users for local databases. As the number of users increases, the performance drops dramatically. Under load, most local database formats drop performance to the point of being unusable (see Figure 2). Since a Web site depends on the time required to retrieve a Web page, which is directly affected by the time required to conduct the database operation, this is a rather critical limitation. If you combine this limitation with the fact that most Web databases gradually increase in size over time, you have a situation that will eventually bring your Web site to a crawl.

Where's the load limit for local databases? This depends on a number of factors, not the least of which is the code you write to access and manage the data. However, most experienced Web developers would probably agree that a local database starts having performance problems at somewhere between 10 and 20 simultaneous users. Not a lot if you're trying to build a Web site to handle tens of thousands of hits per day or more. Also, this problem can't be easily alleviated by throwing more hardware at the Web server (or database file server, as the case may be). The limitation is a symptom of a deeper, more fundamental problem, rooted in the way local database formats are structured.

Data Integrity and Local Databases

Perhaps the most insidious problem with local databases is that the integrity of the data can be fairly easily compromised. This can happen in a number of different ways, and can be very difficult to manage, prevent, or even detect.

Consider that most local databases use indexes to implement sort orders. These indexes are used to locate the correct record in a particular collating sequence. However, if the database is being updated (rather than a read-only database), the corresponding index key values must be updated at the same time. Since the data format is unmanaged, any problem with I/O on the Web server (or with message queuing in Windows) could result in an index that's out of sequence with the data, which means the index may refer to the wrong records. This in turn can result in the wrong records being updated, damaging the integrity of the data. Left unchecked, this can lead to data corruption. This particular problem is much more common than you might think, and is compounded as the size of the database and/or the number of simultaneous users increases.

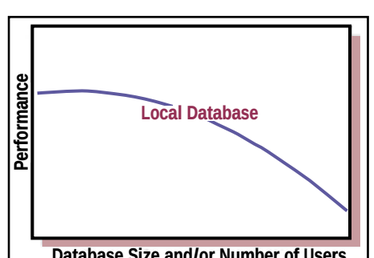


FIGURE 2: Performance in local databases

Another common problem with local database formats is that when data is written, there's nothing to prevent multiple users from accessing the same data at the same time. Some local database formats provide record-locking mechanisms to prevent simultaneous access to the same data, but this assumes the application can lock a record for the entire period of time it's being edited (e.g., pessimistic record locking). Web applications are essentially stateless, and users that are editing data don't (should not, cannot) retain locks on records. This is because there's no way for the server to detect if a user abandons the editing session, thereby leaving the record in a locked state. This allows multiple users to potentially change the same data at the same time, which in turn can corrupt the data.

Even if the absence of pessimistic locking doesn't corrupt the data, another potential problem exists with local database formats since there's no database management system to prevent concurrency problems. Consider a record with a field that represents an account balance for a particular client that's updated in the following sequence:

1. User A reads the account balance of $100.
2. User B reads the account balance of $100.
3. User A increases the account balance by $10, and updates the database with this value. The database now contains the value $110.
4. User B increases the account balance by $10, and updates the database with this value. The database now contains the value $110, but it should be $120, as it has been increased by $10 two times.

This problem exists, again, because a local database has no database management system to prevent such problems.

Client/Server Database Systems

At the opposite end of the database spectrum are managed database systems. Technically, any database with a dedicated software database management system is a managed database system. However, Web databases are specifically client/server (regardless of the database format used), due to the client/server nature of Web development. This narrows the category to the more commonly used client/server RDBMSs, including Oracle, SQL Server, Informix, Sybase, and InterBase.

All these client/server database systems have a dedicated software-based manager to handle the potential problems inherent in the local database formats, and also to provide additional programming and data management capabilities that aren't available in a local database format.

> "Throwing more hardware at the database server usually improves its performance"

Performance Under Load

Since these are managed systems, many of the problems associated with local database formats are simply non-issues with a client/server database system. For example, most client/server systems can handle huge numbers of records (billions of records is not uncommon) without a dramatic performance penalty (see Figure 3). In many systems performance can actually increase under additional user load, as the DBMS can reuse previously cached queries and service multiple client requests concurrently.

Any performance decreases due to load in a client/server database aren't nearly as dramatic as those encountered in local database formats. In addition, throwing more hardware at the database server usually improves its performance. This is far more efficient than relying on the underlying operating system to queue requests for access to a shared local data source. In most client/server systems significant performance drops don't occur until there are hundreds, or in some cases thousands, of concurrent users. When performance is affected, the performance decline is much more gradual than with local databases.

Data Integrity on Client/Server Systems

One of the strongest arguments for implementing a Web application on a client/server system is that the DBMS is specifically designed to protect the integrity of the data. Indexes are used primarily to increase performance, but not in the same way as in a local database system. Because the data is managed dynamically, the chance of index corruption is much less likely. If corruption does occur, it can be automatically detected and corrected before data corruption occurs. Other forms of data integrity problems are also automatically detected and resolved before they can adversely affect the data.

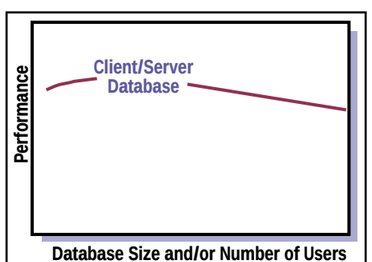


FIGURE 3: Performance in client/server databases

Transaction Control

Since a client/server database system allows many users to modify data offline, it's perfectly suited to Web application use in which users read and modify the data remotely. A client/server DBMS does this with transaction control, logging all potential changes by all users and layering on the updates in the correct order. This alone is no trivial task, as it requires a sophisticated logging and analysis system to simultaneously keep track of all reads and writes for all users.

Transaction control is implemented by logging client reads and writes to a transaction log, which the database system then uses to ensure the correct changes are applied, in the correct order, for each user.

Another advantage of using a transaction processing system is that multiple changes can be treated as if they were a single unit of work. For example, consider a simple transfer from your checking account to your savings account. Two updates must occur – reduce the checking account by a specified amount, and increase the savings account by the same amount.

If either transaction fails to complete (for whatever reason), someone will lose some cash. If the checking account was updated but the savings account update failed, you lose. If the savings account was updated but the checking account update failed, you win. Both updates must occur for the accounts to balance correctly. Transaction processing allows the database server to track changes for both updates, then commit them both once they're successfully done. If either update fails, both records are rolled back to their state just prior to the updates.

Another useful side effect of using transactions to track all changes for all clients is that if a disaster occurs at any point, such as a power failure, the database system can detect these open transactions at restart and roll them back automatically.

Other DBMS Features

There are many other reasons for choosing a client/server database system over a local database format. Here are some of the more common selling points:

- ***Security:*** Grants and revokes access and permissions to a specified object in the database
- ***Views:*** Saves predefined queries that can be used as if they were actual tables
- ***Stored procedures:*** Stores SQL programming in the database to perform common database management tasks
- ***Triggers:*** Automatically executes stored procedures when data in a specified table is modified

Cost and Implementation

The drawbacks to using a client/server database management system are really associated with the cost and complexity of the implementation. Most professional client/server database systems cost far more than their local counterparts. Oracle, for example, can cost several hundreds of dollars per user, or tens of thousands per server, depending on configuration (www.oracle.com). InterBase, on the other hand, is free for the download (www.interbase.com). Most client/server databases are somewhere in between.

Another cost associated with client/server databases is the cost of implementation and maintenance. While some systems, like Microsoft SQL Server, are relatively simple to install and maintain, others, like Oracle, can require a highly paid professional database administrator in order to operate. Again, most fall somewhere in between.

Making a Choice

Are you scared of local databases yet? Don't be. They simply weren't intended for high-volume, heavily accessed Web sites. The choice of a local database as a permanent database for your Web site might be appropriate if the following conditions are true:

- The site will provide read-only data that won't be modified remotely by clients.
- The site won't experience high levels of traffic.
- The site won't manage large amounts of data.
- The site won't implement complex transactions.

If your site doesn't fit into *all* of these categories, you should seriously consider using a client/server database system.

Choose wisely....

About the Author

Ted Blue is president of Blue Star Training (www.bluestarcorp.com), where he conducts training for Internet, software development, and database-related topics. He's the Webmaster for a number of sites, including the San Diego ColdFusion User Group site (www.sdcfug.com). He's also the founder of CERTIFIABLE Internet Services (www.certifiable.net), a Web hosting company.

tedblue@bluestarcorp.com

Holistix **Web Manager 2.0**

Advanced monitoring capabilities and flexible reporting options for Web managers

REVIEWED BY
JERRY BRADENBAUGH

There's a new product on the market for Web system monitoring: Holistix Web Manager 2.0 by Holistix. It provides comprehensive monitoring and detailed reporting for those times when bad things happen to good systems.

VITALS

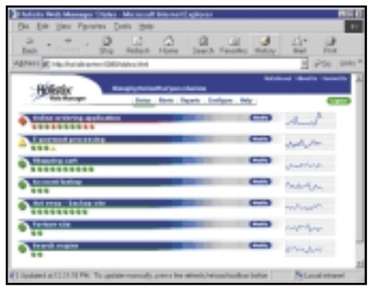
Holistix Web Manager 2.0: Holistix
Address: 1500 Dexter Avenue North Seattle, WA 98109
Phone: 206 272-5201
Web: www.holistix.net
Environment: **Windows** 2000/NT
Pricing: Starts at $1,800 per managed server CPU

Web Manager, coupled with Holistix Remote Monitor, is one of two parts of the Holistix Solution, the company's complete solution to detecting and solving Web system problems while also gauging the quality of the end-user experience.

How It Works

Holistix Web Manager works in two basic capacities: monitoring and reporting. It monitors each component of your Web system. Holistix defines a Web system as "…a combination of a specific N-tier type application model with a network infrastructure" and a Web application as the logical grouping of "…Web system components such as application servers, databases and Web servers, as well as network devices…". Each Web component is closely monitored for any signs of system failure or performance issues. Each component is carefully measured against a series of metrics, which are performance- or health-related conditions such as CPU utilization or processing speed.

Problems happen, and when they do, the information gathered during the monitoring process is leveraged during the reporting process. The report is designed to reveal problematic areas within the system, which means faster troubleshooting and ultimately lower downtimes.



The Holistix Web Manager interface

What's New

That's basically how Holistix Web Manager works. Here are the new features of version 2.0:

- At-a-glance, miniature view of Holistix console
- New profiles for ColdFusion, BEA, F5, Cisco, Linux, and others
- The Correlation Report for comparing related components or conditions
- New reporting engine powered by Macromedia's Flash
- New dedicated monitors (port monitor, Windows 2K service, and others)
- User-specific views of Web applications, reporting, and alerts
- The ability to graph multiple metrics within the same report

Monitoring

The product's new at-a-glance console displays the health of up to 200 Web applications (that's right, 200) within a single centralized view. That means you always know the latest about your Web, FTP and mail servers, databases and application servers, routers, load balancing machines, server farms, and more. No need for toggling between multiple views. This "big picture" view lends itself to easier troubleshooting and, ultimately, less downtime.

Web managers can quickly configure the product using profiles. *Profiles* are ready-to-use templates for monitoring common Web components such as ColdFusion, SQL Server, Microsoft's IIS, and Netscape's Enterprise Server. Each profile contains a set of metrics and threshold levels designed specifically for its Web component. Profiles save you the trouble of having to determine the criteria by which to measure a particular component. Holistix Web Manager 2.0 includes new profiles for ColdFusion, F5's BIG-IP Load Balancer, and BEA WebLogic Server, with more to come.

Holistix also takes monitoring one step further. While much of the application is based on reaction to the Web system being monitored, Holistix Web Manager's reporting takes a more proactive approach. It can identify weaknesses across the system, which allows Web managers the opportunity to reconsider current resource allocations to prevent further headaches.

Reporting Features

Holistix Web Manager provides full reporting on system health, availability, responsiveness, and actions and alerts. Web managers can create detailed custom reports based only on criteria they need to see. Reporting is structured in a deductive form. That is, reports can start as general overviews, then allow users to focus on narrower areas.

Among the reporting capabilities, Web Manager has the Correlation Report, which utilizes a split-screen window to compare relationships among associated alerts, monitors, and time spans. For example, Correlation Report could compare Web page loading times with Web server CPU utilization.

JERRY@CYNICPRODUCTIONS.COM

ABOUT THE AUTHOR
Jerry Bradenbaugh is a Web architect for CynicProductions.com.

Ask the **Training Staff**

A source for your CF-related questions

BY
BRUCE VAN HORN

It looks as if this month's column is all about functions. In the questions and answers below we'll look at RandRange(), ListLen(), ListAppend(), GetTickCount(), and the ever-elusive Soundex() functions.

Q: *I need to build a search engine for our employee directory that will return some results even if the user doesn't know exactly how someone's last name is spelled (I think they call this a "fuzzy" search). How can I do this?*

A: There are two (probably more) ways to do a "fuzzy" search, both of which have little to do with CF and everything to do with SQL. The first and most obvious solution is to use the LIKE operator with wild cards in your SQL (see Listing 1). This first example uses a wild card at the end of the search text. This way, if the user knows the first few letters of the person's last name, it'll return all those names that begin with those characters. Notice the use of "LIKE" instead of the "=" operator.

Another way is to search for words that sound alike, such as "Michael" and "Michelle." This can be done using the Soundex() function (see Listing 2). Please note that Soundex() is not a CF function; it's a SQL function that's supported by most database engines such as SQL Server, Oracle, and Informix (not Access!). Soundex() converts the string into its phonetic representation (way beyond the scope of this column) so that two different strings may have the same "Soundex" value.

Q: *I know the CF Server debug settings let me find out how long it took to process an entire page, but is there a way to find out how long it took to process a particular block of code within a page?*

A: Yes, that's what GetTickCount() was built to do! It's really simple (see Listing 3). Set a variable to the value of GetTickCount() at the beginning of the code you want to test, and another one equal to GetTickCount at the end of the code. Then simply compare the two values to determine how long it took (the value will be in milliseconds).

Q: *In the November issue (**CFDJ**, Vol. 2, issue 11), you showed us how to randomly display one record from a query. How would I show multiple (say four) records from a query without displaying the same record more than once?*

A: Great question! There are many ways to accomplish this, including right at the database level through SQL (this is actually the preferred method but the syntax will be specific to your RDBMS). For example, you can use the ORDER BY RAND clause with an Oracle database. However, here's one way to do it using CF (see Listing 4) that will work with any database.

First, let's retrieve all the potential records from our database. Notice the CACHEDWITHIN attribute to make sure this query doesn't execute every time for every user. Next, I create a local variable, MyList, with a value of "". This variable will ultimately hold a list of the records displayed. Since you want to display only four records, we'll do a CFLOOP that will execute until there are four items in our list of displayed records. Inside the loop we need to pick a random record using RandRange(), check to see if it's already been shown using ListFind(), and, if not, add it to the list using ListAppend(). Last, we'll display that record using array syntax on the query variable.

• • •

Please send your questions about ColdFusion (CFML, CF Server, or CF Studio) to AskCFDJ@syscon.com.

BRUCE@NETSITEDYNAMICS.COM

ABOUT THE AUTHOR
*Bruce Van Horn, a regular contributor to **CFDJ**, is president of Netsite Dynamics, LLC (an Allaire Alliance Partner), and an Allaire certified instructor.*

Listing 1: An example of SQL wild cards and the LIKE operator

```
<CFQUERY NAME="qGetEmp" DATASOURCE="Employees">
SELECT LastName, FirstName
FROM   Employees
WHERE LastName LIKE '#Form.LastName#%'
</CFQUERY>
```

Listing 2: An example of the Soundex() function

```
<CFQUERY NAME="qGetEmp" DATASOURCE="Employees">
SELECT LastName, FirstName
FROM   Employees
WHERE Soundex(LastName) = Soundex('#Form.LastName#')
</CFQUERY>
```

Listing 3: An example of the GetTickCount() function)

```
<!--- Time a block of code --->
<CFSET tickBegin = GetTickCount()>
YOUR CODE GOES HERE
<CFSET tickEnd = GetTickCount()>
<CFSET ExecuteTime = Variables.tickEnd -
Variables.tickBegin>

<!--- Report --->

<CFOUTPUT>
The execution time for the above code was:
#Variables.ExecuteTime# Milliseconds
</CFOUTPUT>
```

Listing 4: Displaying random records from a query

```
<CFQUERY NAME="qGetRandBeans" DATASOURCE="Coffee_Valley"
CACHEDWITHIN="#CreateTimeSpan(0,0,30,0)#">
 SELECT Bean_ID, Bean_Name
 FROM   Beans
</CFQUERY>
<CFSET MyList = "">
<CFLOOP CONDITION="ListLen(Variables.MyList) LT 4">
 <CFSET PickOne = RandRange(1,qGetRandBeans.RecordCount)>
 <CFIF NOT ListFind(Variables.MyList,Variables.PickOne)>
  <CFSET MyList =
ListAppend(Variables.MyList,Variables.PickOne)>
 <CFOUTPUT>
 #qGetRandBeans.Bean_Name[Variables.PickOne]#<BR>
 </CFOUTPUT>
 </CFIF>
</CFLOOP>
```

CODE LISTING

The code listing for this article can also be located at www.ColdFusionJournal.com

Your Own Magazine

Do you need to differentiate yourself from your competitors?

Do you need to get closer to your customers and top prospects?

Could your customer database stand a bit of improvement?

Could your company brand and product brands benefit from a higher profile?

Would you like to work more closely with your third-party marketing partners?

Or, would you simply like to be a magazine publisher?

SYS-CON Custom Media is a new division of SYS-CON, the world's leading publisher of Internet technology Web sites, print magazines, and journals.

SYS-CON was named America's fastest-growing, privately held publishing company by *Inc. 500* in 1999.

SYS-CON Custom Media can produce inserts, supplements, or full-scale turnkey print magazines for your company. Nothing beats your own print magazine for sheer impact on your customers' desks... and a print publication can also drive new prospects and business to your Web site. Talk to us!

We work closely with your marketing department to produce targeted, top-notch editorial and design. We can handle your distribution and database requirements, take care of all production demands, and work with your marketing partners to develop advertising revenue that can subsidize your magazine.

So contact us today!
East of the Rockies,
Robyn Forma,
robyn@sys-con.com,
Tel: 201-802-3022

West of the Rockies,
Roger Strukhoff,
roger@sys-con.com,
Tel: 925-244-9109

Upsize from MSAccess to SQL Server

Part 1 of 2

Afraid? Don't be – we'll show you how, step-by-step

BY ERON COHEN AND MICHAEL SMITH

If you're using ColdFusion with Access today, then it's likely you'll be using it with SQL Server tomorrow. Luckily, Microsoft has provided some tools that make upgrading to SQL Server a relatively easy process. You'll be surprised to learn how simple it is to upgrade and how easy it is to work with.

This article covers the basics of using these tools and helps you plan an easy database upgrade. You'll learn when and how to upgrade, and how to work with the Microsoft Access 2000 "front end" to SQL Server to make your transition as painless as possible.

Knowing When to Upgrade

There are lots of reasons to upgrade from Microsoft Access to SQL Server. The most important is a simple one: many of us are discovering that MS Access really isn't a stable platform when working with ColdFusion. I've found that sites become more stable and easier to work with as a result of the upgrade to SQL Server. And they can run faster too! Access isn't really meant to be a client/server database, which is, in effect, the way you use it when you connect it to the Web with ColdFusion. Furthermore, Allaire and Microsoft point out that the database should not be used this way.

This may come as a surprise to ColdFusion sites that have used Access successfully for years. But those sites probably have limited amounts of traffic or don't do many online updates.

So how do you know when to upgrade? First and most important, if you find that your Web site/Web server is unstable, very often the trouble has to do with using Microsoft Access. Before pointing your finger at Access, however, be sure to check a few things. For instance, be sure you're using CFLOCKs wherever necessary…especially around reading/writing session/application variables and any unthreaded CFX/COM objects.

Next, check that you've got the latest Microsoft Data Access Components (MDAC) from Microsoft's Web site: www.microsoft.com/data. Follow Microsoft's directions if you do upgrade, as installing the MDAC can be tricky. Make sure you stop every service in your service's control panel or you'll end up with a zombie installation of the MDAC (e.g., from the land of the living dead)! If you're already using the latest MDAC and still experiencing inexplicable server crashes, check your ColdFusion server logs. See if you have any Windows NT 232 errors, which are often the result of problems caused by using MS Access. If you have lots of these, you'll know it's time to move to a better database.

What SQL Server Improves over Access

There are also "positive" reasons to upgrade. First, if you plan to have multiple servers sharing the same database, you should really upgrade. Although it's possible to share an MS Access database using UNCs, it's strongly discouraged for performance and security reasons. Furthermore, with proper configuration, multiple servers in different physical locations can share the same SQL Server database. For instance, if your company has a ColdFusion server in Los Angeles and another one in Tokyo, both of them could use the same SQL Server as a data source.

If you're a developer working remotely on a site, you'll definitely find SQL Server the platform of choice. Instead of having to download the database every time you want to change or add a new field, you can access the table structures remotely and change whatever you like…and not even disrupt the Web site while you're doing it. In other words, no more errors because the table is "exclusively locked" by a developer making a change to a table or even just looking at it in design view. Furthermore, SQL Server uses multiple processors and makes better use of your server's RAM to improve performance of your database queries. Let's not forget about all the great features SQL Server has that MS Access doesn't even begin to address. For instance, SQL Server has the ability to use stored procedures and full text searching – both of which can dramatically improve the performance and functionality of your Web site. There are all sorts of other improvements and enhancements, too.

SQL Server is a more complete implementation of the SQL standard. From an application development point of view, you can now handle multiple schemas and complex security requirements. You have additional data integrity features, including triggers. You can choose among various concurrency modes to enhance performance. And there are additional tools that work with SQL Server. If you cut your programming teeth on C, there's an Embedded C interface where you'll feel comfortable programming reports.

The Tools of the Upgrade

Of course, before you can begin, you'll have to set up SQL Server somewhere. Setting it up is beyond the scope of this article. Installing the software isn't all that difficult, but

setting it up properly and securely is quite intricate. If you're using an ISP or Web-hosting provider, they'll set it up for you and give you an IP address and a username and password to get to it. If it will be set up on a machine in your office, hopefully you already have a network administrator or someone experienced with SQL Server to help you get it going. Be sure and keep up with all of the service packs that come out for SQL Server. (Microsoft is currently shipping Service Pack 2.)

AUTHORS' NOTE

The Microsoft Data Engine (MSDE) basically *is* SQL Server. It contains all the underpinnings of SQL Server and uses an identical format. It's a great way to start out a database you'd like to eventually run on SQL Server. It comes free with MS Office/Access 2000.

Assuming you already have SQL Server set up someplace, you'll need some sort of "client" software to administer it. On the most basic level, you could use CFQUERY through ColdFusion to issue commands to SQL Server. This is definitely a viable – if not the easiest – way to communicate with your database.

Enterprise Manager and Other SQL Server Tools

The second way to administer with the database is to install the SQL Server client software on the machine you'll be using to develop your Web sites. This includes the SQL Server Enterprise Manager, Import and Export Data Wizard, and Query Analyzer. To install these, just take the SQL Server CD and put it in. The automenu will pop up and offer to install SQL Server for you (see Figure 1). When you choose to install it by clicking on "Install SQL Server 7.0 Components," you'll be given the option to install only the client software. Be sure and choose "local install" when prompted for that information. Also note that you need to reboot the computer after the install is complete, even though you won't be prompted to do so.

The tools will come in handy whether or not you choose to adopt this software as your main method of setting up your databases. For instance, the Query Analyzer is a real help in figuring out how to optimize your queries and where to add indexes to tune the performance of the database.

A common variant of this second method is to have your ISP install these tools remotely and use PC-anywhere to run them on a colocated machine.

Using MS Access Projects to Manage SQL Server

I find the third method the easiest and most convenient. You can use the Microsoft Access 2000 interface to access and manipulate SQL Server databases. Although the interface isn't exactly the same as when you're using it with an actual Microsoft Access database, it's similar enough that you'll be able to get your bearings quickly and get to work.

To get started, you need to create a new "project" in Access. To do so, choose FILE > NEW. You'll most likely want to choose Project (Existing Database) to gain access to a SQL Server database that's already in service (see Figure 2). It might surprise you the first time you do this that you'll be immediately prompted to save an .ADP file. This file is the "project" file. It will be a small file that simply contains pointers and preferences for accessing the SQL Server database. Just point it to a convenient directory and continue from there.

Next you'll need to fill out the "Data Link Properties" (see Figure 3). For #1, you'll typically enter either an IP address or a machine name to get to your SQL Server. This information will be provided by whoever set up your database; #2 will also depend on the way the database was set up. Once #1 and #2 are set up correctly, you'll be able to pull down the dialog box in #3 and select the database you want to work with. To verify that you've typed the correct information, you can click on the TEST CONNECTION button to see if it fails and to help you ascertain why. Reasons for the failure could be incorrect settings, bad passwords, or the way the security on SQL Server is set up. Another possibility is that Access is trying to use the wrong protocol to communicate with the database. To determine this, open up the "Client Network Utility," which may be found in your SQL Server or MSDE program group. Be sure the correct protocol is selected as the default. For instance, if you're supposed to be using TCP/IP, don't set the default network library to Named Pipes or some other protocol.

FIGURE 1: SQL_Server_Splash

After you press the OK button, something similar to the old, familiar MS Access 2000 interface will open and present you with tables from your SQL Server database. A few prominent differences should be mentioned. First, you'll find that your QUERIES tab is replaced with "VIEWS." SQL Server's Views function along the same lines as Queries in MS Access. Next you'll notice that the Database diagrams option is built right into the menu. You'll find the diagram builder is a definite improvement over the one that you'd normally use with MS Access, but it's still lacking. Finally, you'll see a new item "Stored Procedures." If the idea of Stored Procedures is new to you, you'll be glad to know something about them. They can speed up your queries tremendously by keeping precompiled versions of them waiting to run on SQL Server. This way, you need only supply the values for a variable or two and POW! there's your data.

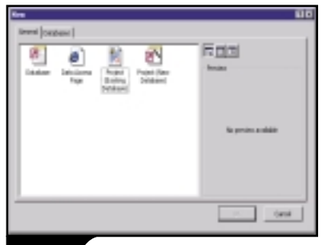
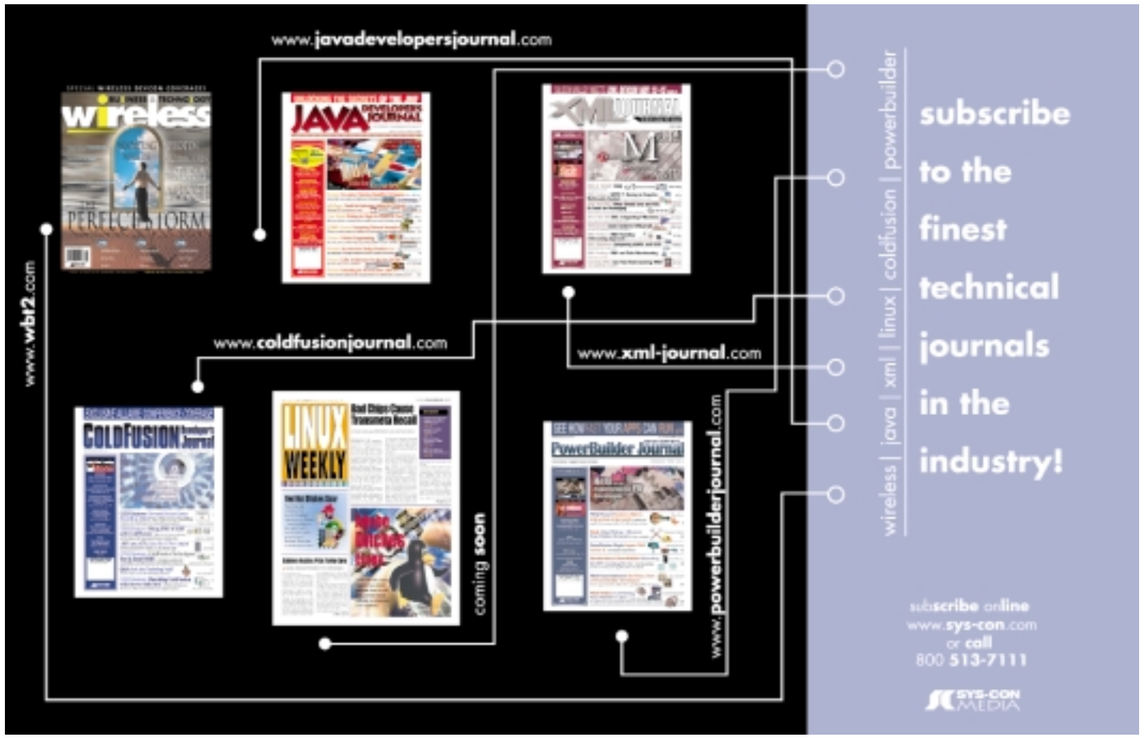


FIGURE 2: MS_Access_ADP



ABOUT THE AUTHORS

Eron Cohen is a ColdFusion programmer, MDCFUG speaker, and author.

Michael Smith is president of TeraTech (www.teratech.com/), an 11-year-old Rockville, Maryland–based consulting company, that specializes in ColdFusion, Database, and Visual Basic development. Michael runs the MDCFUG and recently organized the two-day, Washington, DC – based CFUN-2k conference that attracted more than 750 participants.

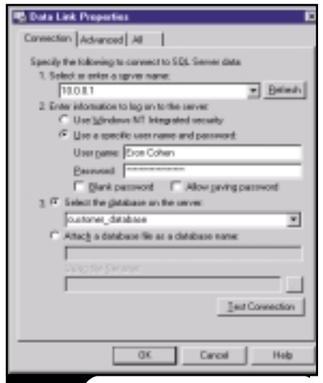


FIGURE 3: MS_Access_ADP_Settings

Using the New MSDE ("SQL Server Lite")

As a developer using SQL Server, you'll want to be aware of the "free" Microsoft Data Engine (MSDE). It's a new product that came out as a free add-on to Office 2000. You can think of the MSDE as "SQL Server 7.0 Lite," because it works like SQL Server but is licensed for less demanding use. Installing it will put a mini-SQL Server on your workstation.

Once you have it installed, you'll have the option of creating your databases using MSDE instead of the native Jet Engine that MS Access would normally default to. From a ColdFusion developer's point of view, choosing MSDE instead of the Jet Engine will provide you with an easy upgrade path to SQL Server. In other words, when you create a database using the MSDE, it's actually already in the native SQL Server format. Moving your database design to a full version of SQL Server is a relatively simple matter, and there's no danger of any compatibility problems in either the data or the SQL statements in your ColdFusion project.

MSDE is managed via MS Access through an Access project (ADP file). It does not come with its own version of SQL Server Enterprise manager, or the other SQL tools, although if you happen to have a copy of Enterprise Manager on your machine, you can use it with MSDE.

The place to find a copy of the MSDE installer is on your Microsoft Office 2000 or Microsoft Access 2000 CD. It may be installed on Windows 95 or 98 or Windows NT. To get started, run SQLSERVERSTP.EXE in the \SQL\INSTALL\ SETUP directory. An install wizard will guide you through the process.

Once you've successfully installed MSDE, you'll find an extra icon in your system tray. It's a computer with a green arrow (if you see a red square instead, it means that for some reason the service hasn't started. You may need to reboot or simply right click on the icon and choose "MSSQLserver – Start"). To use the MSDE you'll need to set up an Access Project as described above.

Summary of Part I

We've covered reasons to upsize your Access database to SQL Server, the tools you'll use to do it, and the "free" MSDE "SQL Server Lite." In Part 2 we'll explain in detail how to copy data to SQL Server and how to troubleshoot your upsizing project.

ERON.COHEN@YAHOO.COM

MICHAEL@TERATECH.COM

SYS-CON INTERACTIVE
The World's Leading ColdFusion Resource
BPA INTERNATIONAL

BUYER'S GUIDE
READER SERVICE
FEATURE STORIES
EDITORIAL COLUMNS
DEVELOPER STORE
COLDFUSION JOBS
AUTHOR GUIDELINES
EDITORIAL BOARD
CUSTOMER SERVICE

Search CFDJ

e-TEST™ your site

INTERMEDIA.NET NT web hosting FREE SETUP: code CFWEB

Cold Fusion Hosting virtualscape

AbleCommerce FREE TRIAL

wired and wireless web synergynow

ColdFusion Developer's Journal .com

SUBSCRIBE | COLDFUSION FORUM | DIGITAL EDITION | SOURCE CODE
ADVERTISING | PRODUCT REVIEWS | SEARCH

SUBSCRIBE TO ColdFusion Developer's Journal AND SAVE CLICK HERE

March **2001**

What's Online This Month...

www.coldfusionjournal.com

Learn what events and developments are happening in the industry by clicking on www.coldfusionjournal.com for up-to-the-minute news. Don't be the last on the block to find out what's making the headlines.

Digital Edition

Don't have your print edition on hand? Can't wait for the next issue to arrive in the mail? For access to the most current output, grab your computer and a connection to the Internet and navigate to ***ColdFusion Developer's Journal*** from anywhere, anytime – guaranteed. Wondering about safe scripting, or the use of XML and XLST with ColdFusion? Not to worry. We're here any time of day to bring you what's current. Move at the speed of technology, even in the dark.

ColdFusion Jobs

Find out the best companies in the industry with direct access for IT professionals. Here's a great site for assisting IT professionals who are curious about the job market. Do you demand privacy and don't want to waste your time looking all over for something that can be so easy to find? Visit ***ColdFusion Developer's Journal.***

Post your résumé, make yourself known to thousands of HR managers who can help you grow your career at their companies! Expand your horizons…reach new heights…discover yourself and your abilities. Let us assist you.

JDJStore.com

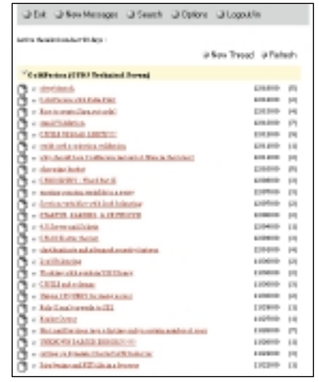
You're just clicks away from what's new in the market. There are new tools and new programs to help you get to where you need to be. From the latest edition of ColdFusion Studio to Allaire's Spectra, it's easy to get them right on your doorstep. They're all here – just visit the ***CFDJ*** Web site.

Ben Forta's ColdFusion Tip of the Day

Click here for ColdFusion tips, links, tags, and resources from Allaire's CF evangelist. A new tip every day!

CFDJList

Join our ColdFusion mailing list community. You and other IT professionals, industry gurus, and ***ColdFusion Developer's Journal*** writers can engage in CF discussions, ask technical questions, and more. Voice your opinions and assessments on topical issues, or hear what others have to say. Monitor the pulse of the ColdFusion industry!



Directory: Structures and You

Choosing the best directory structure for your application

BY JOHN WILKER

No doubt some of us have been involved in one or more projects in which we're acting as the developer or project manager/team lead and are responsible for creating the underlying directory structure for a Web site.

It doesn't matter if the site is on an intranet or Internet site, it still needs some sort of structure. In this article I present two possible ways to lay out the files for your site.

The wrong directory structure (from here on referred to simply as "structure") can cause minor or serious headaches down the road. You could end up with a site that requires numerous page changes every time you make any modification. Obviously that's not a very desirable outcome.

By using ColdFusion we can create a site with a great deal of flexibility and dynamic content control. We can use as few physical files as possible by rewriting the same template over and over again. The first example (Site A) uses a structure that doesn't lend itself to rapid site updates: numerous pages with identical content would have to be changed, which is less than ideal. The second (Site B) uses a structure I've implemented for the intranet at my office by leveraging a product by Ektron Inc. called EmPower (www.ektron.com), and some handy URL variable passing and checking. With this, you can have a dynamic and easy-to-maintain site.

In Site A we lay out the structure as illustrated in Figure 1. This design doesn't scream any terrible wrongs; however, let's go a little deeper. As you can see, the structure is broken down into departments, which sounds good at first. We definitely don't want the marketing and finance content to mix. We also have several folders under each department: CFML, images, docs, and includes. Basically this site has many templates that grab content from the includes folder. A marketing department page will grab body, sidebar and/or navigation content, and anything else the developer includes in the template he or she is creating. For each and every page in the marketing department section there's a template that sits in the CFML folder, as well as a main index file that's located at the root level of the marketing folder.

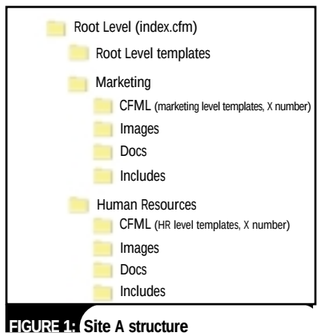


FIGURE 1: Site A structure

In Site B we'll take another approach (see Figure 2). In this example we have a similar breakdown of the site. Each department has its own subfolder that contains folders for docs, images, and includes. I'm sure you noticed there's one folder missing – the CFML folder. The lack of this folder, as I'll explain in a moment, is what makes Site B much more developer-friendly. By friendly I mean that routine maintenance and content changes won't dominate the developer's day. This site works with a single template that resides at the root level of the site. This file contains ColdFusion code that reads URL variables that have been passed in via the URL string of the site. From these variables we can determine which includes folder to bring the content in from. In many cases these variables are numbers that correspond to database IDs of some sort, for example, "http://intranet/template.cfm?Doid=2&S1=a&S2=C&S3=NO". This URL tells Site B to go to the marketing subfolder (marketing has a DepartmentOID [Doid] of 2), pull the S1 file "a", pull the S2 file C, and turn off the S3 section altogether to provide more room for S2. With just a few variables we've rewritten the entire page to suit our needs.

Let's go back to Site A for a more in-depth view of how this site functions. Site A incorporates a great deal of static pages (as static as any CFM page could be). First we have the index page that sits at the root level of each department's folder. This page is a template that calls files from the includes folder to build the final result, for example, "http://intranet/ marketing/index. cfm". For those not familiar with includes, ColdFusion has a tag called <CFINCLUDE> that will incorporate an outside file into your page at the exact point you place the <CFINCLUDE> tag. We have *x* number of templates in the CFML folder at each department level, for example, "http://intranet/marketing/cfml/somepage.cfm", and we have *n* number of department folders in this site. I say *n* because the number differs at each company. Each department may have only one or two pages; others, say human resources, may have 20. Keep in mind that some departments, such as HR, may have numerous forms and informational pages.

A little math shows how just making small changes could turn Site A into a very big and potentially costly (in developer hours) nightmare. Let's say we have *n* number of departments and *x* number of templates for each department, substitute a few numbers and…. For example, a company has 12 departments, each with an average of 10 pages on the intranet. Management has decided to remove the footer piece of code because it's no longer useful and the departments want that little bit of extra real estate. Now one poor developer will be on the hook modifying 120 identical templates just to remove a table row. I hope that developer isn't me. Clearly this isn't an appealing outlook. The more departments, the more possible templates you have. The problem can grow exponentially.

There's a pro to the Site A model – should your site have limited growth potential, then this would be the easiest to implement. The design and construction would be simpler since there wouldn't be any URL variables to capture and utilize. If your site isn't likely to grow past *n* number of departments, you don't have much to worry about in regard to future maintenance or modification issues.

Next we'll look at the footer change in Site B, which uses a dynamic content framework. We'll apply the same model company we used with Site A to Site B. Our company has 12 departments each with an average of 10 templates. To remove the table row that has our now defunct footer, include, we have only one file to modify (the one sitting at the root of our site). This is because we used some clever URL variable passing and a simple query to control dynamically the source content that goes into one template file.

The major con of the Site B model is that we have to manage the URL variables. New links to content within the site would have to be created by the development staff due to the need to pass specific variables with specific values. This is made a bit less difficult and dynamic by creating a list of the URL variables that can be passed instead of the individual ones. However, most departments have only one or two pages that they continually modify, so keeping an eye out for new links is a relatively light chore.

Conclusion

We've covered two types of site structure, static and dynamic. Since the future is unwritten, it's difficult to tell how much growth your site will experience. With that in mind, a model like Site B is probably your best bet for accommodating future growth, as long as you have the time to do the work up front. Site B has the most room for growth and the least amount of potential problems as the needs and layout of the site change. I'll leave you with this thought: imagine our model company with 12 departments and 10 pages per department. The CEO decides to completely change the layout of the entire site, and he wants it done quickly (which means yesterday). Which site would you like to be assigned to modify? Site A with its 120 pages or Site B with its one?

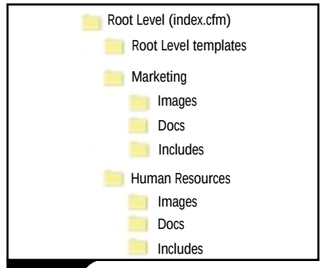


FIGURE 2: Site B structure

ABOUT THE AUTHOR

John Wilker, a Web developer for Directfit, has been developing in HTML for six years and using ColdFusion for more than three years. John recently passed the ACP exam.

JOHNWILKER@ONEBOX.COM

UltraDev 4 **Fireworks 4 Studio**

Leveraging integration for higher productivity

BY
JERRY BRADENBAUGH

Here's a marriage that might actually work out – Macromedia has integrated the data-driven Web site building and management capabilities of Dreamweaver UltraDev with the graphics and animation capabilities of Fireworks.

VITALS

UltraDev 4 Fireworks 4 Studio:
Macromedia

Address: 600 Townsend Street
San Francisco, CA 94103

Phone: 415 252-2000

Web: www.macromedia.com

Environment:
Mac OS: 8.6/9.X
Windows: 9.5/98/2000/ME/NT

Pricing: $699 for the full version. Upgrades are $399 if you already have Dreamweaver 4, $199 if you already own UltraDev. (Buying UltraDev and Fireworks separately would cost you $599 and $299, respectively.)

This will bring you UltraDev 4 Fireworks 4 Studio (not to be confused with Dreamweaver 4 Fireworks 4 Studio, which is the integration of Fireworks 4 with Dreamweaver 4, not Dreamweaver UltraDev).

How Does It Work?

Dreamweaver UltraDev is really a beefed-up version of Dreamweaver, Macromedia's visual authoring tool for Web pages and applications. Dreamweaver enables Web designers and content authors to create attractive, user-friendly, cross-browser Web pages with to-the-pixel positioning. Dreamweaver hides the complexities of HTML, cascading stylesheets, and JavaScript from the designer. Web pages are created with a GUI, so little or no coding knowledge is required…for coders and noncoders alike

Dreamweaver generates code for Web pages, and manages the files of the site and the Web server they reside on. Web sites are rendered as branching hierarchies for easy navigation and organization. The product's site synchronization lets you update a remote Web server without worrying about which files you modified.

Now add to that the ability to create dynamic content from ASP, ColdFusion, JavaServer Pages, and a database. You make your page changes and preview the outcome within the development application. Macromedia calls this *editing live data*. Put it all together, and that's Dreamweaver UltraDev.

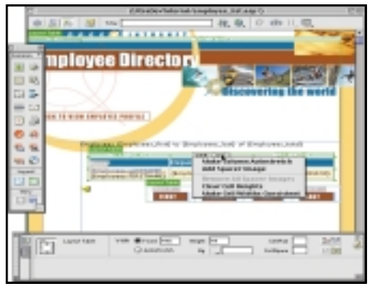
Fireworks is Macromedia's Web graphics and animation tool. It supports bitmap (GIFs, JPGs, etc.) and vector graphics editing, and can be used for live animation. Fireworks also generates (sometimes complex) JavaScript for image rollovers and pop-up DHTML menus. Designers just drag and drop; no coding experience is necessary.

Fireworks has other features, such as image optimization, extensive import/export capabilities including support for wireless bitmaps, and batch processing. Advanced users can leverage the Fireworks 4 (and Dreamweaver 4) scriptable environment, meaning you can write JavaScript code from the provided APIs to extend the application functionality and even allow the two apps to work together.

What's New?

A few of Studio's features:

- ***Split view:*** I've been waiting for this one. Now you can see Code view (the page source code) and the Design view (what your page looks like) in the same window. Any updates you make in one are immediately reflected in the other.
- ***Quicker graphics/HTML editing:*** Using Roundtrip Graphic editing, you can edit and optimize your graphics right in the HTML design space.
- ***Add interactivity easily:*** Add image rollovers, nav bars, and pop-up menus with no JavaScript coding knowledge.
- ***Improved image optimization:*** The image optimization feature enables you to generate the highest-quality image files at the lowest file sizes.



The UltraDev 4 Fireworks 4 Studio layout view

The product touts other features including an integrated text editor and keyword identification, JavaBean integration, tighter source code control using Microsoft's Visual SourceSafe, and the use of Compound Objects to create common functionality such as user authentication and database updates.

Putting It All Together

The advantage here is integration; Dreamweaver and Fireworks in the studio become one. You can build your application using live data, manage your site's files, and design bitmap and vector graphics all in one environment. Just as UltraDev removed the need to toggle between the development environment and the browser, Studio lets you develop and preview everything in a single realm. This speeds your overall development time. Plus, advanced users can tap into each product's API to extend Fireworks and UltraDev, and enable them to work together for increased productivity.

ABOUT THE AUTHOR
Jerry Bradenbaugh is a Web architect for CynicProductions.com.

JERRY@CYNICPRODUCTIONS.COM

SHOP ONLINE AT JDJSTORE.COM FOR BEST PRICES OR CALL YOUR ORDER IN AT 1-888-303-JAVA

JDJ STORE.COM GUARANTEED LOWEST PRICES!

Guaranteed Best Prices
JDJ Store Guarantees the Best Prices.
If you see any of our products listed anywhere at a lower price, we'll match that price and still bring you the same quality service.
Terms of offer:
- Only applicable to pricing on current versions of software
- Offer does not apply toward errors in competitors' printed prices
- Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

Attention Software Developers:
To include your product in JDJStore.com, please contact tony@sys-con.com

ALLAIRE JRun™ Server 3.0 Enterprise
2 CPU LICENSES
JDJStore.com $8602.99

ALLAIRE ColdFusion® Studio 4.5
was $495 NOW
JDJStore.com $449.99

ALLAIRE HomeSite® 4.5
JDJStore.com $93.99

ALLAIRE
ALLAIRE Spectra 1.5

Building on the power and flexibility of the Allaire Business Platform, application frameworks accelerate time to market for delivering online business.The release of Allaire Spectra™ represents the first application framework to address the three critical areas of every successful online business content management, e-commerce, and personalization. With a core set of tailorable components and services, Allaire Spectra™ provides the fastest way to build and manage successful online businesses that unite content, commerce, and personalization.

ALLAIRE Spectra 1.5 . . . . . $14,034.99

ADOBE
Adobe Illustrator 9.0

Adobe Illustrator 9.0 software puts the power of editable vector graphics to work for the Web. Plus it expands yor creative range and enhances your productivity with unlimited transparency capabilities, pwerful object and layer effects, and other innovative features. And its tight integration with other Adobe software ensures a smooth, efficient workflow. Showcase your graphics in print, on the Web, or in dynamic media with complete creative freedom.

Adobe Illustrator 9.0 CD Mac or Win 98/NT/2000. . . . $398.99

INTUITIVE SYSTEMS
Optimizeit 4.0 Professional

Optimize It!

Optimizeit provides a powerful solution for developers looking to rapidly track down and fix performance issues in any Java program. Optimizeit offers complete profiling capabilities including memory leak debugging, CPU profiling and monitoring of object allocations. Easy to use, comprehensive and accurate, Optimizeit provides real-time thorough information about CPU and memory usage, presented in easy-to-read graphs and relevant metrics. Using Optimizeit, developers get complete, coherent information on any part of their program down to the responsible line of source code.

Optimizeit 4.0 Professional $448.99

ADOBE
After Effects 4.1 W9X/NT Production Bundle

The ultimate tool for motion graphics and visual effects, Adobe® After Effects® 4.1 software offers extraordinary creative freedom and control for designing sophisticated motion graphics and visual effects for film, video, multimedia and the Web. Its tight integration with Adobe Photoshop®, Illustrator® and Premiere® gets you up and running fast and makes you more productive.

Adobe After Effects 4.1 . . . . . $1427.99

ALLAIRE
ColdFusion Studio 4.5 with 2 Year Subcription

Allaire ColdFusion® Studio 4.5 with 2 year subscription: Allaire ColdFusion® Studio is a complete environment for rapid Web application development. Based on the award-winning technology in Allaire HomeSite®, ColdFusion Studio gives you a powerful suite of integrated tools for quickly and easily developing applications. From e-commerce to business automation and more, ColdFusion Studio will get you there fast.

Allaire ColdFusion Studio 4.5 . . . . . $656.99

eHELP
RoboHELP Office 2000

RoboHELP Office provides a user-friendly WYSIWYG authoring environment for creating JavaHelp. RoboHELP guides you through the process so you can create a great JavaHelp system - with point-and-click and drag-and-drop ease. Now you can create JavaHelp systems as easily as you create WinHelp, Microsoft HTML Help and WebHelp (cross-platform Help) from the same source product – all with RoboHELP Office.

RoboHELP Office 2000 . . . . . $898.99

ORDER TODAY! 888-303-JAVA WWW.JDJSTORE.COM

CFDJnews

Ektron to Provide Web Content Authoring Tool

(*Amherst, NH*) – Ektron's eWebEditPro content editing tool is embedded in Allaire Spectra, Version 1.5, which started shipping in February. eWebEditPro's functionality offers Allaire Spectra 1.5 users improved "anywhere authoring," including an HTML editor with spell check, and table and cell support. It also features browser independence that enables users to edit content from Netscape or Microsoft browsers.

www.ektron.com

1,000,000th Download of Allaire's HomeSite

(*Newton, MA*) – Allaire Corporation announced the 1,000,000th download of its HTML design and editing tool, HomeSite. HomeSite allows Web developers to maintain pure HTML while integrating with many leading Web technologies such as JSP, CFML, and WML. It features an intuitive WYSIWYN ("what you see is what you need") interface that provides onscreen access to everything needed to build great Web sites. Its project management, quality control, and site deployment features help build, deploy, and manage an entire site.

www.allaire.com

Macromedia Reports Results, *Allaire* Merger Plan

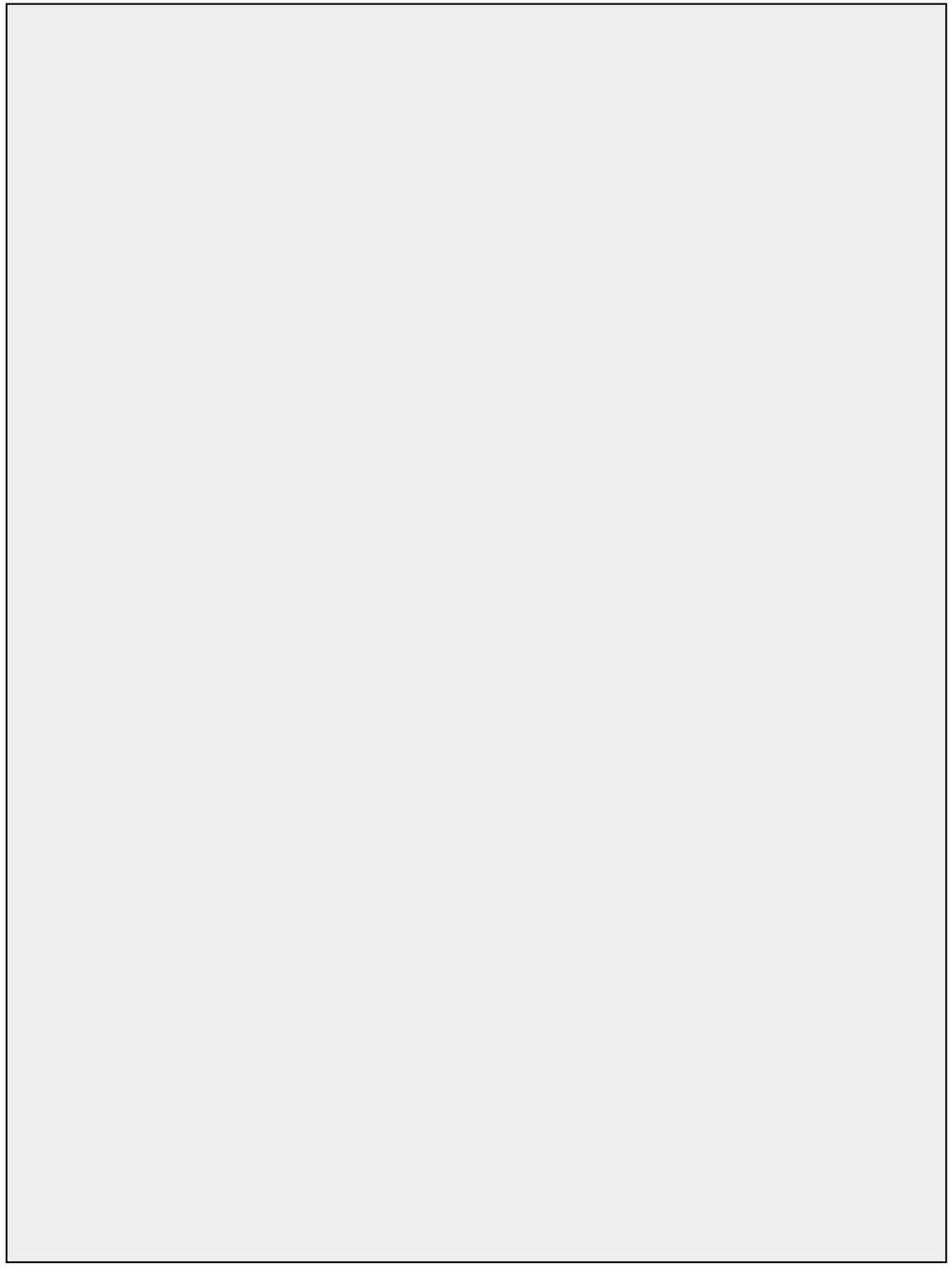
(*Palo Alto, CA*) – Macromedia Inc. announced better-than-expected third-quarter financial results and said it had struck a $360 million deal to merge with Allaire Corp. Under the merger deal, Macromedia will exchange 0.2 shares of its stock and $3 in cash for each Allaire share. Including Allaire's results, Macromedia expects to report pro forma earnings of about $1.70 per share on revenues of $630 million for its 2002 fiscal year. That's 13¢ higher than First Call's consensus forecast for fiscal 2002, when Macromedia's revenues were expected to be $537.8 million.

www.macromedia.com

Ipswitch Integrates WS_FTP Pro with Allaire's Visual Tools

(*Lexington, MA*) – The engine of Ipswitch, Inc.'s, popular Internet FTP product line, WS_FTP Pro, is now integrated with Allaire Corporation's Visual Tools, JRun Studio, ColdFusion Studio, and HomeSite. The integration provides customers with industry-leading file transfer software that increases efficiency and reliability of Web site creation and maintenance.

www.ipswitch.com

ADVERTISER INDEX

ADVERTISER	URL	PH	PG
ACTIVEPDF	WWW.ACTIVEPDF.COM	888.389.1653	2
ALLAIRE CERTIFIED PROGRAM	WWW.VUE.COM/ALLAIRE	877.460.8679	11
ALLAIRE CORPORATION	WWW.ALLAIRE.COM	888.939.2545	41
ALLAIRE CORPORATION	WWW.ALLAIRE.COM/USERGROUPS		19
AMERICAN CYBERNETICS	WWW.MULTIEDIT.COM	800.899.0100	21
CFXHOSTING	WWW.CFXHOSTING.COM	866.239.4678	35
COLDFUSIONJOURNAL.COM	WWW.COLDFUSIONJOURNAL.COM	201.802.3020	49
CONCEPTWARE AG	WWW.CONCEPTWARE.COM/EYE	011.781.275.6171	3
CORDA TECHNOLOGIES	WWW.CORDA.COM	801.802.0800	9
CYSCAPE	WWW.CYSCAPE.COM/BHFREE	800.932.6369	48
EVOLUTION B	WWW.EVOLUTIONB.COM/SYNERGYFORFREE	877.622.7551	55
HOSTMYSITE	WWW.HOSTMYSITE.COM		48
INTERMEDIA	WWW.INTERMEDIA.NET	800.379.7729	56
JAVAEDGE CALL FOR PAPERS	WWW.SYS-CON.COM/JAVAEDGE	201.802.3069	39
JAVAEDGE CONFERENCE & EXPO	WWW.SYS-CON.COM/JAVAEDGE	201.802.3069	28-29
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	53
MACROMEDIA	WWW.MACROMEDIA.COM	415.252.2000	13
PACIFIC ONLINE	WWW.PACONLINE.NET	877.503.9870	43
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	23
SYS-CON CUSTOM MEDIA	WWW.SYS-CON.COM	201.802.3022	43
SYS-CON MEDIA, INC.	WWW.SYS-CON.COM	800.513.7111	47
SYS-CON MEDIA REPRINTS	WWW.SYS-CON.COM	201.802.3024	47
TECHIEGOLD	WWW.TECHIEGOLD.COM	617.267.4700	51
WIRELESS BUSINESS & TECH	WWW.WIRELESS-MAGAZINE.COM.	201.802.3020	20
WOODBOURNE SOLUTIONS	WWW.WOODBOURNESOLUTIONS.COM	877.668.8575	51
XMLEDGE CONFERENCE & EXPO	WWW.SYS-CON.COM/XMLEDGE.COM	201.802.3069	45
YESSOFTWARE	WWW.CODECHARGE.COM	650.754.9810	4