SHOW REPORT: WEB SERVICES EDGE CONFERENCE & EXPO page 46

U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

May 2003 Volume: 5 Issue: 5

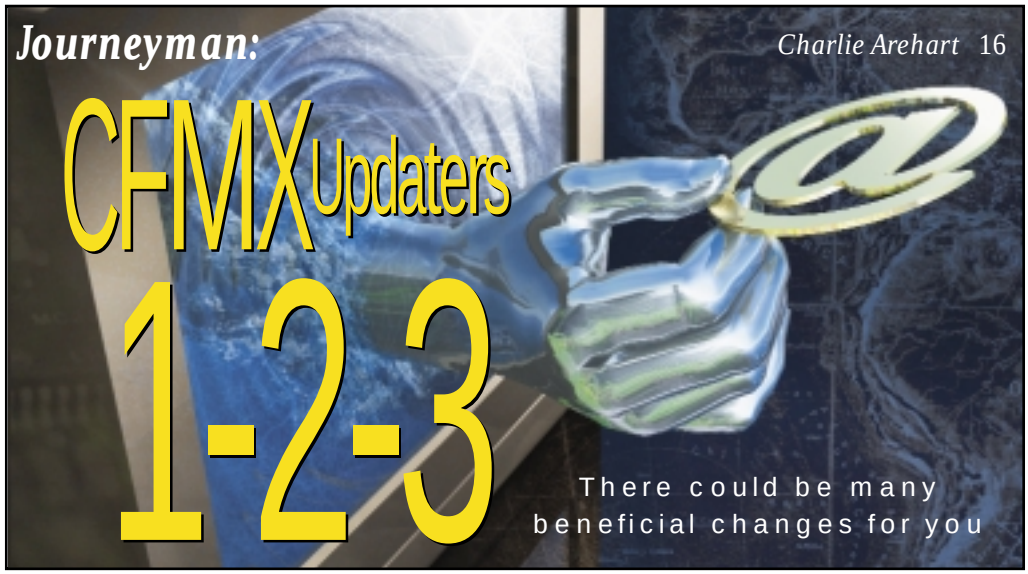
RETAILERS PLEASE DISPLAY UNTIL JULY 31, 2003

$8.99US $9.99CAN

06

0 71486 02689 1

SYS-CON MEDIA



ColdFusion Developer's Journal

editorial advisory board

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

editorial

editor-in-chief
Robert Diamond robert@sys-con.com
technical editors
Charlie Arehart charlie@sys-con.com
Raymond Camden raymond@sys-con.com
editorial director
Jeremy Geelan jeremy@sys-con.com
executive editor
Jamie Matusow jamie@sys-con.com
editor
Nancy Valentine nancy@sys-con.com
associate editors
Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com
assistant editor
Jennifer Stilley jennifer@sys-con.com

production

production consultant
Jim Morgan jim@sys-con.com
art director
Alex Botero alex@sys-con.com
associate art directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Tami Beatty tami@sys-con.com

contributors to this issue

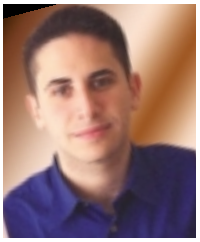
Charlie Arehart, Jason Clark, Ben Forta, Hal Helms, Simon Horwith, Jon Kurz, Greg Lang, Brendan O'Hara, Dominic Plouffe, Stephen Rittler, Christian Thompson

editorial offices

SYS-CON MEDIA
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) by **SYS-CON Publications, Inc.**
postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645

Speed!

One of the topics that comes up most often when talking with CF developers about ColdFusion is performance. Recently, I assisted two CF developer friends with some site speed problems that they were having. One of them is a beginner, a developer who couldn't understand why his site was running slow, or why his Web hosting company was yelling at him that his pages were spiking CPU usage on their server (the horror!). He runs a fairly well-trafficked Web site, and is truly representative of the fact that CFML is a language that will let a novice get up and running fairly quickly, with just an idea and a little bit of technical know-how to put together a professional looking site.



By Robert Diamond

It also shows that while CF enables folks to do this, a few months down the line, as site traffic starts to build – most will hit a bit of a roadblock. After a few hours of work going through his code, I had his site up and running significantly faster, mostly from fixing some of the usual "rookie mistakes." Among the most notable of these: every query involved a 'select *' grabbing much more data than necessary; most pages were selecting the same data over and over again using some of the strangest CFLOOPs that I've ever seen, many running the same query several times over instead of using joins or relational databases, and of course, not a single thing on his site was cached. Fixing all of these was relatively straightforward, and later that week he received a lovely thank you e-card from his Web site provider.

The second friend's site was a bit more tricky (more advanced developers always get themselves into bigger problems), and required more than just run-of-the-mill scans over code. Everything looked okay at a quick glance, and it wasn't until we started to use CF's built-in query debuggers to display execution times that we began to get to the bottom of his problem.

Ultimately, we were able to improve things by using a combination of new stored procedures in his SQL database, better passing along of session variables to eliminate redundant queries, and a run-through of the settings in CF Admin to tweak the number of pages processing. Again, he had no queries cached.

The bottom line here is that the ColdFusion server can often run only as well as its worst queries and its most poorly written pages. There's a never-ending list of tips here each month in ***CFDJ***, as well as all over Macromedia's and the other major CF sites out there, for improving code and site performance. All of them are worth looking at. I recommend setting aside a day, and focusing just on improving site performance, going through all that code you wrote while starting out and never went back to fix, etc. It's all good info, and you, your server, and your customers will appreciate it.

A few small items of note from around the industry… The fascinating progress over at Macromedia.com has continued this month, as they give an unprecendented look "under the hood." Not only is it refreshing to see a company using their own products and putting their challenges out front, but for those of us whose livings are intertwined with Macromedia's products, it ranks near the top of the most useful case studies we've ever seen. On the *CFDJ* front – voting for the *ColdFusion Developer's Journal* 2003 Readers' Choice Awards is now under way. Hundreds of developers have cast their votes so far, and polls will be open through the end of August. Don't delay, come vote today! (www.sys-con.com/coldfusion/readerschoice2003)

One other note – only a few months after I said I would, I've begun blogging. Follow my nutty little world at www.robertdiamond.com.

About the Author

Robert Diamond is vice president of information systems for SYS-CON Media, and editor-in-chief of **ColdFusion Developer's Journal.** *Named one of the "Top thirty magazine industry executives under the age of 30" in* Folio *magazine's November 2000 issue, Robert holds a BS degree in information management and technology from the School of Information Studies at Syracuse University. Visit his blog at* www.robertdiamond.com.

robert@sys-con.com

SYS-CON MEDIA

president & ceo
Fuat A. Kircaali fuat@sys-con.com

business development
vp, business development
Grisha Davida grisha@sys-con.com

advertising
senior vp, sales & marketing
Carmen Gonzalez carmen@sys-con.com
vp, sales & marketing
Miles Silverman miles@sys-con.com
advertising director
Robyn Forma robyn@sys-con.com
director of sales & marketing
Megan Mussa megan@sys-con.com
advertising sales manager
Alisa Catalano alisa@sys-con.com
associate sales managers
Carrie Gebert carrieg@sys-con.com
Kristin Kuhnle kristin@sys-con.com

sys-con events
president, events
Grisha Davida grisha@sys-con.com
conference manager
Michael Lynch mike@sys-con.com

customer relations
senior customer care/circulation specialist
Niki Panagopoulos niki@sys-con.com
manager, jdj store
Rachel McGouran rachel@sys-con.com

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

web designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com

online editor
Lin Goetz lin@sys-con.com

accounting
financial analyst
Joan LaRose joan@sys-con.com
accounts receivable
Kerri Von Achen kerri@sys-con.com
accounts payable
Betty White betty@sys-con.com

subscriptions
Subscribe@sys-con.com
1 888 303-5282
For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $8.99/issue
Domestic: $89.99/yr (12 issues)
Canada/Mexico: $99.99/yr
all other countries $129.99/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. $15 all others

Tales from the List

Macromedia.com gets a new look and feel... makes perfect sense to me!

In March, Macromedia created quite a stir in the community by releasing a completely new version of their Web site (www.macromedia.com). This beta version of the site reflected the Macromedia vision of the way Flash can be used with ColdFusion (or any other application server) to deliver rich Internet content on the Web. The site redesign was met with mixed response from the community.

By Simon Horwith

As you will soon see, whether you like the new site or not, it can serve as an opportunity to learn something new about developing rich Internet applications. The new site was the subject of several small discussions on the List – I will focus on one in particular that I found interesting.

Macromedia compiled a report on what they learned from the experience of developing and releasing the beta version of their site, which can be found at www.macromedia.com/special/progress_report/beta1/. Angela McGregor posted a link (http://rss.com.com/2009-1122-992201.html?type=pt&part=rss&tag=feed&subj=news) to a CNET review of the new Macromedia site, stating that she agreed with the review and thought the article should be titled "Drunk on Flash."

This article criticized Macromedia for heavy use of Flash in order to make the site more "flashy" as opposed to using it more as a tool to "deliver value." It also reviews Forrester's three best practices for Web site design, which the author of the article obviously feels Macromedia did not follow. Devendra Shrikhande's response speculated whether or not Macromedia had read the article before compiling their own report.

Macromedia's own Server Community Manager, Christian Cantrell, chimed in to clarify that Macromedia had read the article, listened to user feedback, and had moved on to a beta 2 version of the site, which he advised us to visit (www.macromedia.com). Christian was responded to by TheaterMania Tech Department, who asked why the beta version was public rather than private, especially given the "bad press" that had resulted from replacing the existing macromedia.com site with the beta version. This is in reference to some Macromedia customers' and developers' complaints about slow load times, heavy use of Flash, and difficulties navigating the site, among other things.

The post sparked a discussion that brought up some very excellent points and gave credibility to Macromedia's decision to make the new site "live." Christian didn't deny that they

—*continued on page 45*

About the Author

Simon Horwith, senior consultant at Fig Leaf Software in Washington, DC, has been using ColdFusion since version 1.5. He is a Macromedia-certified Advanced ColdFusion and Flash developer and is a Macromedia-certified instructor. In addition to administering the CFDJList List Serve and presenting at DC-area CFUGs, Simon is a contributing author to Professional ColdFusion 5.0 *(WROX) and to* ColdFusion MX - The Complete Reference *(McGraw-Hill), as well as technical editor of* The ColdFusion 5.0 Certification Study Guide *(Syngress).*

shorwith@figleaf.com

The Power of Antipatterns

Studying failure can actually lead to success

It seems that lately, you can't pick up a book or magazine without hearing about design patterns. If you're new to the idea of design patterns, they're simply time-tested solutions to common problems.

By Hal Helms

Design patterns began with the work of Christopher Alexander, a PhD in architecture (as in buildings). Alexander noted that certain problems have optimal solutions, and designated these solutions as "design patterns."

Here, Alexander describes design patterns (from his Web site, patternlanguage.com): "For example, if you are building a house you need to go from outside to inside and there are centuries of experiments on how to do this in a 'just so' way. Sometimes the transition is marked not by just a door but by a change in elevation (steps, large, small, straight, or curved), or a shaded path, or through a courtyard. We wrote up this knowledge in the form of a pattern about entrance transitions."

While design patterns got their start in architecture, they were even more successful in influencing the object-oriented programming community. Four brilliant programmers and theorists (dubbed the "Gang of Four") saw analogies between building architectures and software architectures, and decided to document design patterns in software. The result was a seminal work called *Design Patterns*.

It's not light reading, but it does reward the student with insight into "just so" solutions. Experienced programmers can document the designs they have found to work, sparing learners from repeating mistakes. As they say, a good programmer learns from his/her mistakes, but a great programmer learns from the mistakes of others.

I think the interest in design patterns in the ColdFusion community is great – it indicates that our community is maturing and growing. But this isn't an article on patterns, but one on antipatterns.

An antipattern is just what it sounds like – the opposite of a pattern. While a pattern documents a proven solution, an antipattern documents a proven fiasco. Perhaps the spirit of antipatterns was best captured by the American newspaperman and curmudgeon, H.L. Mencken in this quote: "For every complex problem, there is an answer that is clear, simple – and wrong."

Antipatterns are those clear, simple, and wrong ideas that, in our haste or inexperience, we may find ourselves drawn to. While design patterns provide us with valuable advice on what to do, antipatterns provide us with invaluable advice on what not to do.

Table 1 shows some antipatterns that are not software related to help get you into the spirit.

Antipattern name	Synopsis	Refactored Solution
Blackout	The oil light in your car comes on. Knowing that this is not a good thing, you apply duct tape to the dashboard in order to black out the warning sign.	Take your car into a service station and find out the problems before your engine seizes.
Due Color	While attending a training class in Las Vegas, you notice that the helpful electronic tracker by the roulette wheel indicates that the last nine numbers have all been red. Excited by this great opportunity, you put down a large bet on black.	Study basic statistics textbooks until the concept that "the wheel has no memory" is deeply meaningful to you. Until then, stay out of casinos.
Monty Hall	You find yourself on a game show where you must choose one of three doors. Behind one door is a prize; the other doors hold nothing. The host then opens one door (always a losing one) and asks you if you would like to switch your choice to the remaining door. Figuring that the odds haven't changed simply because you saw a losing door, you stay with your original choice.	Actually, you should switch your choice of doors when offered to do so by the host. Your odds are better by switching.

Table 1: Antipatterns that are not software related

As can be seen from the Monty Hall antipattern, sometimes the wrong choice just feels right. No wonder, then, that we get suckered into an antipattern. (If you're absolutely sure that I'm wrong on my advice regarding the game show, head over to www.halhelms.com, where I'll try to show you that your intuition is wrong in this case.)

Some time ago, I wrote an article about how to write really bad code, inspired (as I claimed) by the realization that the sheer amount of bad code out there indicated that people really wanted to write bad code, and I stepped up to help them.

While it's fun to write (and hopefully, read) such articles, there is a deeper truth at work. Sometimes, we can learn more from failures than successes. When we succeed, we may not know why we succeed and any lessons we "learn" from our successes may be nothing more than coincidences. We see this caricatured in superstitious people who wear lucky clothing or work through rituals that they have somehow associated with good things happening.

But failure has a way of clearing the mind of such specious associations. Sometimes pain can actually be therapeutic, if it leads us to question our assumptions and be relentless in searching for the truth. That's just where antipatterns can help us.

Some antipatterns are strictly technological while others have broader implications. Look at the real antipatterns shown in Table 2, taken from a Web site devoted to this topic, www.antipatterns.com.

Studying antipatterns tells us about things we don't already know. As my students can attest to, I consider cut-and-paste programming one of the true evils of programming, but I had never considered the solution to the Boat Anchor problem – and there lies the value of antipatterns: they can help us see things in a new light.

Antipatterns are growing in popularity as people find that studying failure can actually lead to success. Among the resources available are these:

- www.antipatterns.com: Good information on the site and links to buy various books on antipatterns.
- www.bitterjava.com: A Java-centric resource (and book), but an excellent one that can be utilized by developers working with ColdFusion components.
- http://mindprod.com/unmain.html: This wonderfully written site is both fun to peruse and instructive.

Antipattern name	Synopsis	Refactored Solution
Boat Anchor	A costly technology purchased by a systems development project goes unused.	Send competent engineers to the product training course, in order to evaluate, before you buy the product.
Cut-and-Paste Programming	Code reused by copying source statements leads to significant maintenance problems.	Black box reuse reduces maintenance issues by having common source code, testing, and documentation for multiple reuse.
Fire Drill	Management waits until the last possible moment to allow developers to proceed with design and implementation; then they want results almost immediately.	Proactive design and prototyping are often warranted; even if customers and management staff are not completely onboard.
Golden Hammer	A familiar technology or concept is applied obsessively to many problems.	Expanding the knowledge of developers through education, training, and book study groups exposes developers to new solutions.

Table 2: Four real antipatterns

I've been involved in many "post mortems" where the project went wrong – sometimes badly wrong. Too often, these sessions turn into a search for personalities to blame. Why were we late on delivering the software? Because Sue's group was late with the SQL queries! One way out of this ugly situation is to direct the attention of the group toward discovery of antipatterns rather than toward a hunt to punish the guilty.

Years ago, W. Edwards Deming, justly famous for being the "man who taught the Japanese quality," invented a game that, he said, mirrored the quality situation in virtually all American companies. He called it the "Red Bead Game." In the game, thousands of white beads are placed into a drum, and for every nine white beads, a red bead is added, making the percentage of red beads to the total number of beads 10%.

Certain audience members representing workers were given paddles into which 100 indentations had been made. Each indentation held a bead. They were then told to insert their paddles into the drum of beads and withdraw it with the paddle filled with beads. The number of red beads would then be measured. "Quality" was defined as having fewer than ten red beads. The game was played with great gusto, with "managers" inventing quotas and rewarding "good" workers who got less red beads while punishing "bad" workers.

Of course, none of this worked in the slightest since the workers had no control over how the beads fell into the holes made for them. The problem with quality, Deming drove home, is almost never in the people themselves; it is in the system. Or, putting it just slightly differently, there are antipatterns at work that produce bad results.

The same is true with software development. Rather than trying to identify "bad" programmers, we're much more likely to succeed if we identify antipatterns and work together to refactor a solution in which everyone can win.

About the Author

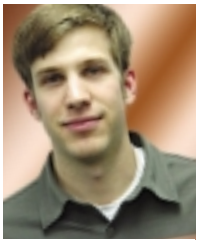
Hal Helms (www.halhelms.com) is a Team Macromedia member who provides both on-site and remote training in ColdFusion, Java, and Fusebox.

hal@fusebox.org

Use the JavaMail API to Build an IMAP Custom Tag

Integrating with Java is easier than ever with MX

"Don't reinvent the wheel." We've all heard, and perhaps uttered, this maxim many times during our programming careers. ColdFusion MX has introduced new features to help CF developers avoid reinventing the wheel, including tag-based user-defined functions (UDFs) and ColdFusion components (CFCs).



By Christian Thompson

As more CF developers begin to use these features, more code will be available for other developers to download and use. However, code reuse in ColdFusion is not limited to ColdFusion code. Integrating with Java is easier than ever with MX. Since the developer base for Java is large and it has been around much longer than ColdFusion UDFs and CFCs, there is a lot of code already out there that CF developers can use. I discovered this firsthand on a recent project when I didn't think about using existing Java libraries before starting down another path.

This project required me to programmatically access e-mail messages in a subfolder of an e-mail inbox. I couldn't use the CFPOP tag to do this because the protocol implemented by CFPOP – the "Post Office Protocol" – is designed to allow users to download messages from the "inbox" folder only. Another protocol is required to access subfolders – the Internet Message Access Protocol (IMAP).

After discovering that I needed to use IMAP to accomplish my task, I looked for a way to use it with ColdFusion. Unfortunately, since there is no built-in ColdFusion tag that implements IMAP, and I had been looking for an excuse to play around with Java, my first reaction was "Cool! I'll write my own!"

I learned a few IMAP commands and started playing around with the low-level Java networking libraries. Two days and many lines of code later, I had written a marginal IMAP client. Although it mostly worked, it still needed additional functionality and I wasn't at all confident in its stability. I knew there had to be a better way.

JavaMail API

Fortunately, I discovered that there is a much better way – use the JavaMail API. According to Sun's JavaMail Web site (http://java.sun.com/products/javamail), this API "provides a set of abstract classes that model a mail system" including support for IMAP – the same functionality (plus a lot more) that I had been struggling with for two days. Best of all, since it's included in the J2EE platform, on which ColdFusion MX is built, I didn't have to download anything or make any configuration changes to use it. This was the option I should have considered from the start.

Design Goals

Now that I had found a library, I wanted to make it simple to use from ColdFusion. Since I was already comfortable using the CFPOP tag, I decided to encapsulate some of JavaMail's functionality into a custom tag that, as much as possible, provides the same interface as CFPOP. The tag's full code is shown in the code listing. Listing 1 shows the functions that interact with the API.

Listing 2 shows the tag's main logic. For the sake of brevity, the code shown in the listing includes only one of the actions provided by CFPOP's interface – the "GETHEADERONLY" action. This excludes the "GETALL" action, which provides the ability to view a message's contents and attachments, and the "DELETE" action.

The Code

The JavaMail API consists of a bunch of classes that model a mail system. The highest-level class is the "Session" class. The session class allows you to specify properties like "port" and "timeout." To specify properties, first create and set a "Properties" object. The following sets the protocol to IMAP and the port to 143:

```
obj_Properties =
   createObject("Java",
   "java.util.Properties");
obj_Properties.init();
obj_Properties.put(
   "mail.store.protocol",
   "imap");
obj_Properties.put(
   "mail.imap.port",
   "143");
```

Now create the session object and pass in the properties object:

```
cls_Session =
   createObject("Java",
      "javax.mail.Session");
obj_Session =
   cls_Session.getInstance(
      obj_Properties);
```

The session class also provides a method for getting the "store." The store class abstracts the underlying protocol (in this case IMAP) and provides methods for logging into the mailbox.

```
obj_Store = createObject("Java",
   "javax.mail.Store");
obj_Store =
   obj_Session.getStore();
obj_Store.connect(
   "my_mailserver",
   "my_username",
   "my_password");
```

Now you are ready to start accessing folders. The only folder that you can access using POP is the "inbox" folder. With IMAP, you can access subfolders below the inbox. The following code accesses and opens the folder "test," which is a subfolder of "inbox":

```
obj_Folder =
   obj_Store.getFolder("inbox/test");
obj_Folder.open(
   obj_Folder.READ_ONLY);
```

Note that this code opens the folder with read permissions only. This is a good idea if you only want to browse the contents of the folder. If you want to delete, move, or update messages, open the folder with read and write permissions:

```
obj_Folder.open(
   obj_Folder.READ_WRITE);
```

The next step is to get a list of messages in the folder. With the JavaMail API, this is done by retrieving an array of "Message" objects.

```
ar_Messages =
   obj_Folder.getMessages();
```

This code returns all messages in the folder. If you only need a subset of the messages, you must pass an array of message numbers to the "getMessages" function. Because I modeled my IMAP tag after the CFPOP tag, I wanted to allow the user to pass a *list* of message numbers to the tag. This discrepancy was easily remedied by using the built-in "ListToArray" ColdFusion function.

```
ar_Messages =
   obj_Folder.getMessages(
      ListToArray("1,2"));
```

Once you have the messages, you are ready to read the message headers using methods of the "Message" object. The following shows how to access the "subject" and "to addresses" of the first message in the folder (which is located at position 1 of the messages array):

```
// Get the received date
str_Subject =
   ar_Messages[1].getSubject();

// Get the to addresses
cls_RecipientType =
   CreateObject("Java",
      "javax.mail.Message$RecipientType");
ar_To =
   ar_Messages[1].getRecipients(
      cls_RecipientType.TO);
lst_ToAddresses =
   arrayToList(ar_To);
```

Note that the "getRecipients" method of the "Message" object returns an array of addresses since an e-mail message can be addressed to more than one person. This is also true when getting the "from", "reply-to", "CC", and "BCC" addresses. Also note the dollar sign ($) syntax used to invoke the "RecipientType" class. RecipientType is an "inner class" of the "Message" class. Although you can access inner classes in Java code by using dot notation (e.g., "Message.RecipientType"), this doesn't work in ColdFusion code.

Now that you have read the messages, the next step is to delete the unwanted ones. To do this, you set the "deleted" flag of the message and then call the folder object's "expunge" method:

```
cls_Flag =
   CreateObject("Java",
      "javax.mail.Flags$Flag");
ar_Messages[1].setFlag(
   cls_Flag.DELETED,
   true);
arguments.obj_Folder.expunge();
```

Where to Go from Here

The code presented here only scratches the surface of the functionality provided by IMAP. Most obviously, it lacks the ability to read message contents and attachments. Another useful feature would allow users to retrieve a listing of subfolders. If you are interested in adding these or other features, check out the JavaMail tutorial at http://developer.java.sun.com/developer/onlineTraining/JavaMail/contents.html and the API specification at http://java.sun.com/products/javamail/1.3/docs/javadocs/index.html.

Summary

With the release of MX, ColdFusion developers have more options for code reuse, including native access to a wide array of predeveloped Java software. Sometimes, especially for developers who don't have extensive Java experience, using existing Java libraries is not always the most obvious option. However, in ColdFusion MX, it can be a very worthwhile option to consider. It certainly is one that I strongly recommend the next time you start a project and think "Cool, I'll just write it myself..."

About the Author

Christian Thompson is a certified advanced Macromedia ColdFusion MX developer. He is a senior software engineer for Inserso, a technology consulting firm headquartered in Annandale, VA, where he has specialized in ColdFusion application development for over two years.
cthompson@inserso.com

Listing 1: IMAPfunctions.cfm

```
<cfscript>
   function GetSession(int_Port, int_Timeout)
   {
      var cls_Session = createObject("Java",
         "javax.mail.Session");
      var obj_Properties = createObject("Java",
         "java.util.Properties");

      // Change timeout to milliseconds
      var int_MilliTimeout =
         arguments.int_Timeout * 1000;

      // Set properties
      obj_Properties.init();
      obj_Properties.put("mail.store.protocol",
         "imap");
      obj_Properties.put("mail.imap.port",
         arguments.int_Port);
      obj_Properties.put(
         "mail.imap.connectiontimeout",
         int_MilliTimeout);
      obj_Properties.put("mail.imap.timeout",
         int_MilliTimeout);
      obj_Session =
         cls_Session.getInstance(obj_Properties);
      return obj_Session;
   }
```

```
function GetStore(obj_Session,
   str_ServerName,
   str_Username,
   str_Password)
{
   var obj_Store = createObject("Java",
      "javax.mail.Store");
   obj_Store = obj_Session.getStore();
   obj_Store.connect(arguments.str_Servername,
      arguments.str_Username,
      arguments.str_Password);
   return obj_Store;
}

// Get and open the folder
function OpenFolder (obj_Store,
   str_Folder, bln_ReadWrite)
{
   var obj_Folder =
      obj_Store.getFolder(arguments.str_Folder);
   // Open the folder (writable if necessary)
   if (bln_ReadWrite is true)
      obj_Folder.open(obj_Folder.READ_WRITE);
   else
      obj_Folder.open(obj_Folder.READ_ONLY);
   return obj_Folder;
}

function GetMessages(obj_Folder,
   lst_MessageNumber)
{
   var ar_Messages = "";
   // If the user has passed in a list of message
   // numbers, get only those numbers. Otherwise,
   // get all messages in the folder
  var ar_Numbers = ListToArray(
     arguments.lst_MessageNumber);
   if (ListLen(arguments.lst_MessageNumber) gt 0)
      ar_Messages =
         obj_Folder.getMessages(ar_Numbers);
   else
      ar_Messages = obj_Folder.getMessages();
   return ar_Messages;
}

// This function populates and returns a query
// containing message header information
function GetHeaders(lst_MessageNumber, obj_Folder)
{
   var ar_Messages =
      GetMessages(arguments.obj_Folder,
         arguments.lst_MessageNumber);
   var str_Columns = "date,from,messagenumber"
      & ",replyto,subject,cc,to";
   var qry_Messages = QueryNew(str_Columns);
   var int_CurMes = "";
   var obj_Message = "";
   var ar_From = "";
   var ar_To = "";
   var cls_RecipientType =
      CreateObject("Java",
      "javax.mail.Message$RecipientType");

   for (int_CurMes = 1;
      int_CurMes lte arrayLen(ar_messages);
      int_CurMes = int_CurMes + 1)
   {
      obj_Message = ar_Messages[int_CurMes];
      ar_From = obj_Message.getFrom();
      ar_To = obj_Message.getRecipients(
         cls_RecipientType.TO);

      queryAddRow(qry_Messages);
      querySetCell(qry_Messages,"date",
         obj_Message.getReceivedDate());
      querySetCell(qry_Messages,"from",
         arrayToList(ar_From));
      querySetCell(qry_Messages,"messagenumber",
         obj_Message.getMessageNumber());
      querySetCell(qry_Messages,"subject",
         obj_Message.getSubject());
      querySetCell(qry_Messages,"to",
         arrayToList(ar_To));
   }
   return qry_Messages;
}
</cfscript>
```

Listing 2: IMAP.cfm

```
<!---
Include the functions defined in Listing 1.
Alternatively, you can combine both listings
into one file.
--->
<cfinclude template="IMAPfunctions.cfm">

<cfparam name="attributes.server" default="">
<cfparam name="attributes.port" default="143">
<cfparam name="attributes.username" default="">
<cfparam name="attributes.password" default="">
<cfparam name="attributes.action"
   default="getHeaderOnly">
<cfparam name="attributes.folder" default="inbox">
<cfparam name="attributes.name" default="">
<cfparam name="attributes.messageNumber" default="">
<cfparam name="attributes.timeout" default="60">

<!--- Open the session and store --->
<cfset obj_Session = getSession(attributes.port,
   attributes.timeout)>
<cfset obj_Store = getStore(obj_Session,
   attributes.server,
   attributes.username,
   attributes.password)>

<!--- Manipulate the mailbox based on
   the desired action --->
<cfswitch expression="#UCase(attributes.action)#">
   <cfcase value="GETHEADERONLY">
      <!--- Open the folder read only --->
      <cfset obj_Folder = OpenFolder(obj_Store,
         attributes.folder,
         false)>
      <!--- Get the headers --->
      <cfset qry_Messages =
        GetHeaders(attributes.MessageNumber,
          obj_Folder)>
      <!--- Return the headers query --->
      <cfset "caller.#attributes.name#" =
        qry_Messages>
   </cfcase>
</cfswitch>

<!--- Clean up the objects --->
<cfset obj_Folder.close(false)>
<cfset obj_Store.close()>
```

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

CFMX Updaters 1-2-3

There could be many beneficial changes for you

Is your CFMX server running with the latest set of fixes? Macromedia now uses "Updaters" as a means to apply patches and extensions. There have been three so far. Are you running the latest, or any at all? How would you know? And why should you bother?

In this month's **Journeyman** article, we'll review the latest and third Updater, pointing out some of the key changes. Updater 3 came out in mid-March, and though it may be old news to some readers, there are many who are perhaps slower to apply the Updaters. This is one for which you shouldn't wait any longer.

As Macromedia puts it, Updater 3 "delivers more than 100 new enhancements to ColdFusion MX. While the primary focus of this updater is server stability, other areas of improvement include updates to the Macromedia Flash Remoting service, internationalization and character encoding fixes, enhanced database integration, and additional fixes for COM object integration." There's still lots more that may interest you, including enhancements in Web services, Web server connectors, search engine safe URLs, WDDX, Linux support, J2EE session handling, server-side redirects, and much more.

By Charlie Arehart

Even so, many may not know if their server has been updated or which changes may benefit them. Indeed, since Updater 3 incorporates the previous two Updaters, it's possible that you may not be aware of what's changed since the initial release of CFMX. There could be many beneficial changes for you. There are 36 pages of Release Notes. While all of us should read them, I realize that many will not.

I'll review some (just some) of the key features of all three Updaters. I'll also explain how to find out which Updater, if any, you're running. And of course, I'll point you to the Updaters themselves at Macromedia's site, as well as the various documentation provided.

What Updater Are You Running?

Before discussing the Updaters themselves, it's useful to know which Updater you're running, if any. Some may know that the ColdFusion Administrator offers a page called "Version Information" that shows which version of CF you're running. It might report a value like this: 6,0,0,58500. This indicates you're running CFMX (the first number, 6); the sub-version 58500 indicates that you're running Updater 3. This information is also reported by the CF variable #Server.ColdFusion.ProductVersion#.

The problem with both of these approaches is that they're just data, a number. You wouldn't know that 58500 indicates Updater 3, unless someone told you. And in fact Macromedia does have a page that tells you which version number indicates which Updater. See www.macromedia.com/support/coldfusion/ts/documents/mx_version_info.htm.

To make things easier, I've put together a small program that does this lookup for you! Just save the code in Listing 1 as getCFMXVersion.cfm, and run it in your environment. It might produce a report like:

```
CF Version Analysis:
This server is running ColdFusion Server, Developer
The current version number is 6,0,0,58500 which means it's running with
Updater 3.

There are Updaters for CFMX available at
http://www.macromedia.com/software/coldfusion/special/updater/faq
```

This indicates that I'm running Updater 3. What Updater are you running?

Where to Obtain the Updaters?

You'll note that the report from that program also indicates that you can find information about the Updaters at www.macromedia.com/software/coldfusion/special/updater/faq. This Macromedia page not only offers the latest Updater itself, but also answers frequently asked questions. Indeed, I'm sure there may be some questions you still have about the Updaters after reading this article. I direct you to this page to see if the questions are already answered for you.

Another important link on that page is the Release Notes, which are at www.macromedia.com/support/coldfusion/releasenotes/mx/releasenotes_mx_updater.html. It's my sense that many tend to ignore the Release Notes, perhaps figuring that all they discuss is how to install the Updater. They do indeed do that, but they also discuss what the Updater fixes (as well as what was fixed in previous Updaters). They also list known problems that continue to exist even with the Updater applied.

What's Changed in Updater 3?

So, should you apply Updater 3? The consensus, at least at this writing in early April, is that you should indeed apply the Updater. Not only does it solve many important problems that have caused a good deal of grief for many developers and administrators, it also solves some "smaller" problems that might not have garnered a lot of acclaim but might very well prove important to you (or developers on your server). As I noted above, there are 36 pages of Release Notes, and all but 10 of those are about Updater 3.

Again, Updater 3 also applies all the fixes that were implemented in Updaters 1 and 2. The Release Notes cover those changes as well, and I'll highlight some of them later.

And those aren't the only Release Notes to read. ColdFusion MX Server is built upon an integrated version of JRun, Macromedia's J2EE application server. This is generally transparent to us, but it's worth noting that Updater 3 for ColdFusion MX Server updates that underlying server to JRun4 SP1a, and some may find value in perusing the JRun4 SP1a Release Notes for more information on changes brought by that (www.macromedia.com/support/jrun/releasenotes/4/releasenotes_4_sp1a.html).

It may be worth clarifying at this point that there are two product lines for CFMX: the Server product (which most of us run, available in Pro and Enterprise editions) and the J2EE Server product, for implementing CFMX on an existing J2EE server you may already have (like WebSphere, WebLogic, SunOne, JRun, etc.). It's perhaps important to note that this is the first Updater that applies to both the server and J2EE products (though not the WebSphere version if purchased through the IBM Passport licensing program). Okay, enough background. What about the changes?

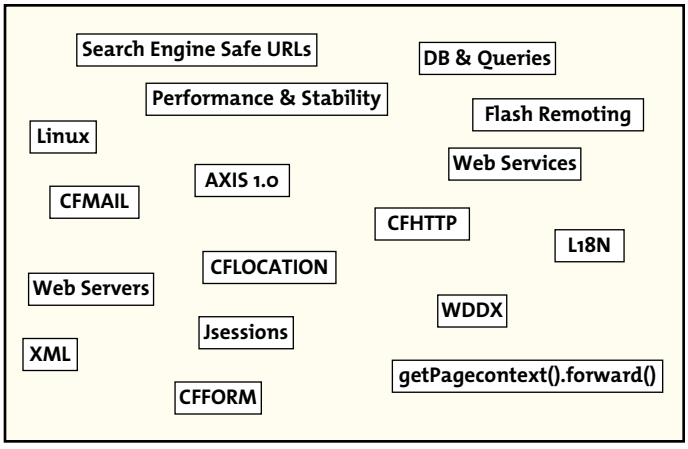


Figure 1: Some of the key changes in Updater 3

Web Server Connector Issues

Perhaps the most important set of changes in Updater 3 revolves around issues with Web server connectors, such as IIS and Apache. There are changes that address problems of supporting multiple sites in IIS, memory leaks, as well as problems under load for both IIS and Apache, and more. Indeed, the notes indicate that "Updater 3 was rearchitected to be more user friendly for ISPs or users with multisite installations."

The changes are so substantial that there is even a "FAQ for ColdFusion MX connector configuration," which offers additional information beyond the Release Notes, at www.macromedia.com/support/coldfusion/ts/documents/connector_install_faq.htm. But be sure to read the Release Notes as well. There are also new batch files for listing and updating connectors.

Regarding Apache, if you're using Apache 2.0.x, Updater 3 requires 2.0.43 or later, whereas Updater 2 required 2.0.41 through 2.0.43, and Updater 1 required 2.0.40.

Some Important Challenges Solved

Beyond Web server connector issues, there are still many other useful and important changes. A few of them are related to issues I've identified in previous articles:

- I pointed out in the June 2002 issue of ***CFDJ***, "New Possibility in CFMX: Server-side Redirects" that CFMX now supported a new function, getPageContext().forward(), that finally allowed us to do true server-side redirects (as opposed to CFLOCATION, which is really a client-side redirect in that it sends a request to the browser to have it request the new page). See www.sys-con.com/coldfusion/article.cfm?id=450 for more info. The only problem was that the function would fail if you used it on a page that was called as a form post (a form action page) and there were any form variables present. This problem is fixed in Updater 3. Hurray! This will open the door to some interesting new possibilities for MVC-style development with CFMX.

- I pointed out in the August 2002 issue of ***CFDJ***, "New Possibilities for Session/Client Variable Handling in CFMX," that CFMX now supported optional J2EE sessions. See www.sys-con.com/coldfusion/article.cfm?id=494 for more information. As I point out in comments on that page (posted after the article was printed), there is a problem with them that can occur if you perform a CFLOCATION when J2EE sessions are used. If the page doing the CFLOCATION does not have the JSessionID cookie variable present (either because the browser doesn't support cookies or it's the first CF page the user has executed on your site), then CFMX will pass the JSessionID variable as filename.cfm;jesssionid=nnn. The problem is that this form of URL is not supported by IIS, so if you're directing the user to a page on an IIS server, it will fail with a 404. Updater 3 fixes this, to an extent, by at least allowing you to use AddToken="no" on the CFLOCATION to stop it appending that jsessionid. (The problem still arises with the new URLSessionFormat function.)

Some other problems I've been annoyed by that I'm glad to see fixed are:

- If you tried to use the GetHTTPRequestData function to process an XML stream sent into CFMX with a content-type of text/xml, as might be sent to your page from an XML client or the CFHTTPPARAM TYPE="xml" attribute, you'd get an error. That's been fixed.
- The Code Compatibility Analyzer, available in the CFMX Administrator, was seriously flawed (in my opinion). Unless you chose the "advanced" option to select the tags, functions, and other constructs to analyze, it didn't identify any code that would be incompatible. What a shame that many may have simply pressed "run analyzer" and been misled into complacency, thinking their code was compatible. The Updater fixes this.
- In Updater 2, ColdFusion did not properly handle escaped single quotes in queries. This is fixed.

Several fixes relate to Web services processing:

- First, Updater 3 finally gives us the latest version of the Axis engine, which is an Apache project toolset that underlies CFMX's Web services processing. This is an important fix that solves many problems that were related to our running a beta .9 release of that engine until now. We now have the Axis 1.0 RC 1 version, which is fully JAX-RPC compliant.
- While this may for the most part make some things that failed previously now work, it also solves a problem for which developers may have implemented a workaround that must now be removed. If you were referring to Web service parameter names that had underscores, you had to remove them. You must now use the underscore if it's expected by the Web service.
- .NET Web service clients can now consume ColdFusion component (CFC) Web services that return a query.
- If you're using CFMX for J2EE on JRun 4, there are also extra steps needed to enable Web services in CFMX on that platform.

Other Fixes

Other things fixed that may interest some folks include, to quote the Release Notes:

- Search engine safe URLs of the form *.cfm/* did not work properly, so a URL such as http:/server/test.cfm/alpha/beta would not execute test.cfm.
- The security certificates used for ColdFusion tags that incorporate Java applets, including CFGRID, CFSLIDER, and CFTREE, expired in December 2002.
- The CFHTTP tag, when used to perform a GET operation on a remote URL over HTTPS, recognized certificates from VeriSign and Thawte only. ColdFusion now recognizes many more certificate authorities, including Entrust and Equifax.
- The Rand and RandRange functions did not work properly in a custom tag that was called more than once on the same page.

There are also some performance-related fixes. For instance, to quote the Release Notes:

- The jrun.xml file was configured such that ColdFusion MX Administrator Simultaneous Requests settings up to 19, including the default value of 10, could cause server instability or reduced performance.
- Client-variable database storage has been optimized to improve performance during insert and update of client variables into a CDATA table. The core table could lock at times, causing performance issues.

There are also several changes of note regarding Flash Remoting. First, Updater 3 applies Flash Remoting's own Updater 1. For more information on that, see www.macromedia.com/go/flashremotingupdater_releasenotes. Going back to Web services for a moment, another interesting change is that the ability to call Web services from a Flash client through the Flash Remoting gateway is now disabled by default. If you want to enable Flash clients to request Web services (even outside ones) through your server's Flash Remoting gateway, you must now enable that via some XML to be placed in a configuration file. Finally, another Remoting change of note is that all exception objects thrown from the Flash Remoting gateway now have the same format in the Macromedia Flash client. The exception fields are as follows: Details, Description, Code, Type, RootCause.

There are several security-related fixes, including all the relevant patches described in the Security Zone as of March 19. There are some issues to be aware of regarding Sandbox/Resource Security. If you're unfamiliar with Sandbox/Resource security, see my multipart series in the Macromedia Devnet, starting at www.macromedia.com/devnet/security/articles/sandbox_01.html.

There are also several fixes that relate to all databases as well as specific ones like SQL Server, Oracle, DB2, and Sybase.

There are still other interesting changes, such as whether CF returns a recordset when a CFQUERY produces none (such as an update), as well as issues related to WDDX, internationalization, and lots more. And there are many fixes to Linux issues that were described in www.macromedia.com/v1/handlers/index.cfm?ID=23524.

Going back to CFMX for J2EE for a moment, there is also a change required for the JVM's security policy file, for all but WebSphere 4. This is different from the similar settings documented in the installation instructions. There are also various other matters related to installing and supporting the CFMX for J2EE product. Be sure to search the Release Notes for "j2ee" to learn more.

Indeed, everyone should read the Release Notes carefully to determine what things might apply to you. I've tried to highlight things that I think might impact most developers, but there are lots of details in the notes.

Compilation Issues

On another front, and perhaps a bit of a good news/bad news scenario, Updater 3 changes the CFML compiler; therefore, some existing compiled code must be compiled to work correctly. Macromedia's solution is simply to force all code to be recompiled. When you install Updater 3, it copies your current WEB-INF\cfclasses directory to cfclasses_backup in the same location.

The unfortunate implication of this change is that all CF templates will therefore have to be recompiled. Of course, CFMX does this automatically the first time someone tries to run the template, but as I discuss in the two-part series, "Compilation and Precompilation in CFMX Templates" in the October and November 2002 issues, starting at http://www.sys-con.com/coldfusion/article.cfm?id=519, there is quite a cost for that first person running the template. And in a multipage app, it could mean the first users suffer quite a bit of pain and delay.

As I discuss in those articles, you can alleviate that burden by precompiling the code before the user runs it. Macromedia has finally provided a batch file to do it, called updater_compile.bat.

Sadly, it's the same limited-function version that, as I discuss in the article, was passed around early in the post-release timeframe last summer. For instance, from the Release Notes describing how to name the directory whose code is to be precompiled:

cf_app_file_dir is the directory that contains the CFML files to be compiled. It must be underneath the Web root directory.

In other words, the directory pointed to must be under the same Web root where the WEB-INF is located. This means you can't use it on files outside that root (such as files in virtual directories outside inetpub\wwwroot on IIS or in a directory mapped using the <virtual-mapping> element in jrun-web.xml).

The shame is that there is a much better version. Now the version I printed in the October article had some problems itself, and the comment area for the article on the ***CFDJ*** Web site offers a corrected version. But you can also see my 12/8/02 blog entry, at http://cfmxplus.blogspot.com/2002_12_08_cfmxplus_archive.html, for my proposed alternative.

What Had Changed in Updaters 1 and 2?

The focus of this article is Updater 3, and while Macromedia thinks you should make the upgrade, perhaps you may not be persuaded by what you've seen so far. But Updater 3 also includes all the fixes in Updaters 1 and 2, and maybe there are fixes or changes applied in those that may be more important to you. Let's take a moment to review some of the more interesting fixes. They include:

- The ability to access various administrator-related and other sensitive information using the ColdFusion.ServiceFactory object, which might compromise the security, has been restricted. See Ben Forta's article on part of the previously available feature, at www.sys-con.com/coldfusion/article.cfm?id=500.
- Several changes related to COM processing were implemented.
- You can now use a variable for the value of CFSETTTING's new REQUESTTIMEOUT attribute. See my September 2002 article, "Hidden Gems in CFMX" for more on that new feature, at www.sys-con.com/coldfusion/article.cfm?id=503.
- Use of server-side validation by way of "_required" hidden fields on a form now shows any specifed VALUE as the custom error validation message.
- There were a couple of changes to CFMAIL and spool processing that increase the robustness of the mail process.

Again, there are still more changes, including more related to Web server connectors, database processing, security, internationalization, WDDX, Administrator issues, and CFML tag and function fixes. These are addressed in a specific document within the Release Notes, "ColdFusion MX Issues Fixed in Previous Updater Releases," at www.macromedia.com/support/coldfusion/releasenotes/mx/releasenotes_mx_updater02.html.

Known Issues that Remain

While the Updaters have addressed hundreds of challenges, truly bringing CFMX beyond a ".0" release, there are still plenty of issues that remain. Before you complain that something is broken, you might want to see if perhaps Macromedia has already recognized the issue. In addition to a section in the Updater 3 Release Notes about "known problems," there is yet another "Known CFMX Issues" tech note 18325 at www.macromedia.com/support/coldfusion/ts/documents/tn18325.htm. The lists are not the same.

And among those "known issues" that still remain, there are ones related to "page has expired" messages that arise when using the back button (with a workaround), scheduled tasks, mail processing, query of queries, service restarting, CFC lookups, and more.

Indeed, there's still another "known issues" page for the CFMX for J2EE product line, at www.macromedia.com/support/coldfusion/releasenotes/mx/knownissues_mx_j2ee_p2.html.)

Do check out these and all the other documents I've referred to here. I hope I've highlighted enough to motivate you to do so, if not at least to seriously consider applying the Updater. And if you've been holding off on moving to CFMX because it's a ".0" release, perhaps you'll reconsider that as well.

About the Author

Charlie Arehart is co-technical editor of **ColdFusion Developer's Journal**. *He's also a certified Macromedia trainer/developer, Team Macromedia member, and CTO of SysteManage. He contributes to several CF resources, is a frequent speaker at user groups throughout the country, and provides training, coaching, and consultation services. He is also now a partner in CommunityMX.com.*

carehart@systemanage.com

Listing 1

```
<!---
Name: getCFMXVersion.cfm
Author: Charlie Arehart
Version: 3/26/03
Published in the May 2003 CFDJ
--->
<h2>CF Version Analysis:</h2>

<cfscript>
versinfo = arraynew(1);
versinfo[1] = structnew();
versinfo[1].name = "the base version, with no updater";
versinfo[1].number = 48097;

versinfo[2] = structnew();
versinfo[2].name = "Updater 1";
versinfo[2].number = 52311;

versinfo[3] = structnew();
versinfo[3].name = "Updater 2";
versinfo[3].number = 55693;

versinfo[4] = structnew();
versinfo[4].name = "Updater 3";
versinfo[4].number = 58500;

curversion = listlast(Server.ColdFusion.ProductVersion);
</cfscript>

<cfoutput>
This server is running <b>#Server.ColdFusion.ProductName#, #server.coldfu-
sion.ProductLevel#</b>
<p>
The current version number is <b>#Server.ColdFusion.ProductVersion#</b>
<cfif left(Server.ColdFusion.ProductVersion,5) is "6,0,0" >
     which means it's running with
     <cfif server.coldfusion.appserver is "j2ee" and curversion is 58096>
          Updater 3 on ColdFusion MX for J2EE.
     <cfelse>
          <cfloop from="1" to="#arraylen(versinfo)#" index="i">
               <cfif curversion eq versinfo[i].number>
                    <b>#versinfo[i].name#</b>.
                    <cfbreak>
               <cfelseif curversion lt versinfo[i].number>
                    <b>patches not yet up to the final release of
                     #versinfo[i].name#</b>
                    <cfbreak>
               <cfelseif i is arraylen(versinfo)>
                   a version <b>greater than  the final release of
                   #versinfo[arraylen(versinfo)].name#</b>.
                    This tool has not been updated yet to recognize that
                    version number.
               </cfif>
          </cfloop>
     </cfif>
<cfelse>
     <!--- there were no updaters for CF 5 or before, so there's no
      reason to try to report on them prior to CFMX, and this program
      will not report on any version greater than CFMX (productversion > 6)
--->
</cfif>
<p>
There are updaters for CFMX available at <a
href="http://www.macromedia.com/software/coldfusion/special/updater/faq/">
http://www.macromedia.com/software/coldfusion/special/updater/faq/</a>.
</cfoutput>
```

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

Dynamic Client-Side Input Validation

You may not need these rules very often, but it's nice to know they're there

Using JavaScript to perform client-side input validation can be simple. However, the task can become more complicated when the set of inputs changes or increases in number, the rules change, or the validation depends on other inputs. This article will show how to create a system to dynamically maintain input fields and validation rules that can be applied to the inputs.

By Jon Kurz

Input Validation

Most of us have used JavaScript to validate the inputs on a form. Validation is important in order to make sure that the inputs conform to some rules and do not cause problems when these inputs are then saved to a database, or used for some other purpose. When strings are used as input, you may simply want to remove any leading or trailing spaces before the form is submitted. In the case of numbers, there are several common validation rules that might apply:

1. Null is allowed
2. Null is not allowed
3. The value must be 0 or greater
4. The value must be greater than 0
5. The value must fall within a range

For the first example, consider having one form, one input, and one validation rule, in this case that the value must be greater than 2000. Here are the segments of code that we are concerned with:

```
<script language="JavaScript1.2">
function validateInputs () {
   if (document.form.year.value <= 2000) {
          document.form.year.focus();
          alert("The Year field must be over 2000.");
          return false;
   }
}
</script>
```

```
<form action="myUpdate.cfm" method="post" name="form"
 onsubmit="return validateInputs();">
Enter start year <input type="text" name="year" value="">
<input type="submit" name="submit" value="submit">
</form>
```

This will validate numbers, as long as they are entered properly, but will not check strings the way we would want. Because of that, it will let strings go through. If the validation function finds that an input does not pass the rule, an alert box is displayed, and the focus is set to the input box. Next, let's say that the validation rule has changed. We want to allow values greater than or equal to 0. The function would look something like this:

```
<script language="JavaScript1.2">
function validateInputs () {
        if (document.form.year.value < 0) {
                document.form.year.focus();
                alert("The Year field must be 0 or
greater.");
                return false;
        }
}
</script>
```

We've made a couple of simple changes, and as a result, updated the page to reflect the new validation requirements. What if we add another input field? Not a problem; we just add the input box and a new section to the validateInputs function, as follows:

```
<script language="JavaScript1.2">
function validateInputs () {
   if (document.form.year.value < 0) {
          document.form.year.focus();
          alert("The Year field must be 0 or greater.");
          return false;
   }
   if (document.form.month.value > 5) {
          document.form.year.focus();
          alert("The Month field must not be greater than 5.");
          return false;
   }
}
</script>
<form action="myUpdate.cfm" method="post" name="form"
```

```
onsubmit="return validateInputs();">
Enter start year <input type="text" name="year" value=""><br>
Enter start month <input type="text" name="month" value=""><br>
<input type="submit" name="submit" value="submit">
</form>
```

That's a little work, but not too much at this point. The question is: How much is too much? How many inputs would have to be added before it becomes cumbersome to maintain? Also, how much work will be required if the validation rules change? How much time will be required to make the changes and move them to your production site? One solution is to build a utility application to maintain the inputs and validation rules. This will give us a great deal of flexibility and control, and will allow us to make changes without changing code.

This utility will consist of two main parts: database and interface. The database will contain the list of input fields that will be used, and also a list of all validation rules. The interface will give us the ability to manage the lists.

The Database

To get started, we will have a table called Fields, defined as follows:

"Fields" table

Column	Type
ID	int (identity)
Name	varchar(128)
DisplayOrder	int
ValidationID	int
Active	bit

The Name is what will be displayed on the input form, to describe each field where values are entered. The DisplayOrder is used to display these fields in an order other than Name. If this is not important to you, you can leave it out. The ValidationID is the ID of the validation rule that is used to validate the given input field. The Active field is used to turn the fields on and off, thus determining whether or not to display them on the form.

Next, we will have a table called Validation, defined as follows:

"Validation" table

Column	Type
ID	int (identity)
Name	varchar(50)
DisplayOrder	int
ValidationTemplate	varchar(256)
ValidationMessage	varchar(128)
Active	bit

The Validation table will contain a Name that describes the validation rule, for example, "Cannot be blank." This will make it easy to understand for someone administering the fields and associated rules. The DisplayOrder is simply used to display the validation rules in a particular order in the admin system. You could simply order them by Name, in which case you don't need the DisplayOrder field. The ValidationTemplate field will contain JavaScript code that will later be read-in and populated in the validation functions. The ValidationMessage is part of the message that will be displayed when a given validation rule is not met. The Active field is used to turn validation rules on and off.

Now let's look at a sample of data as it might appear in the Fields table:

ID	Name	DisplayOrder	ValidationID	Active
1	Work Hours	3	2	1
2	Efficiency	2	3	1
3	Widget Throughput	1	3	0
4	Number of Employees	4	1	1

So, looking at this information, the "Widget Throughput" field would be displayed first; however since it is currently inactive, it will not show on the form. The "Efficiency" field will then be displayed first, and it uses Validation rule number 3, which we will see shortly. What you can see from this is that everything about the fields and their validations is stored and maintained in the database, which will make it much easier to manage. If the validation rule changes, we simply change the ValidationID for that field, and we're done.

At this point, you're probably wondering what the validation data looks like. Here's a sample:

ID	Name	DisplayOrder	ValidationTemplate	ValidationMessage	Active
1	0 or greater	2	(v < 0)	must be 0 or greater.	1
2	Hours per Day	1	(v < 0) || (v > 24)	must be from 0 to 24.	1
3	Between 0 and 100%	3	(v < 0) || (v > 100)	must be between 0 and 100%.	1
4	Not NULL	4	(v == "")	cannot be null.	1

As you saw earlier, "Efficiency" and "Widget Throughput" both use the same validation rule, which is defined as being between 0 and 100%.

Putting It All Together

So, how does all this work? Besides the admin tool, we still have to create the code that knows what to do with the Fields and their associated validation rules. Let's go back to the myUpdate page and look at the new sections of code.

First, we will run a query to get a list of all the Fields to be displayed on the form:

```
<cfquery name="qryFields" datasource="myDataSource">
    SELECT
            *
    FROM
            Fields
    WHERE
            Active = 1
    ORDER BY
            DisplayOrder
</cfquery>
```

Now we will combine JavaScript code with ColdFusion in order to populate the JavaScript validation functions:

```
<script language="JavaScript1.2">
function validateInputs () {
```

```
<cfloop query="qryFields">
   <cfset validationid = qryFields.ValidationID>
   <cfquery name="qryValidation" datasource="myDataSource">
          SELECT
                    *
          FROM
                    Validation
          WHERE
                    ID = #validationid#
   </cfquery>

   var f = document.form.i#qryFields.ID#;
   var t = "#qryFields.Name#";
   f.value = Trim(f.value);
   var v = f.value;
   if (#qryValidation.ValidationTemplate#) {
          f.focus();
          alert(t + " #qryValidation.ValidationMessage#");
          return false;
   }
</cfloop>
}
</script>
```

The validateInputs function will contain all the validation rules, implemented as a series of *if* statements. For each input field, we run another query that gets the particular validation rule for that Field. *Note:* You could modify the qryFields query that gets the list of fields, so that it also retrieves the validation rules, but I am explicitly running the qryValidation query in order to clarify how this section works.

We then set the variable *f* to the ID of the given Field, prefixed with the letter *i*. The variable *t* will contain the actual name of the input field, as it is stored in the database. Next, we use a Trim function, which is assumed to exist prior to the validateInputs function. For the examples here, you must have a Trim function. You can create your own Trim function, but there are also many available on the Web.

The next line is very important. We set the variable *v* to the Trimmed value of *f*. The *v* variable will be the convention that we use in the Validation table to construct our validation rules. Whether you use *v* or something else, the variable must be the same between the function and the ValidationTemplate field in the Validation table.

The if statement contains the ValidationTemplate for the given Field. If the value in *v* is found to be invalid, as determined by the ValidationTemplate, then the focus is set to the corresponding form field, and an alert message is displayed that contains the name of the field with the validationMessage appended to it. Finally the function returns false so the form will not be submitted. If all of the validations pass, then the form is submitted.

To create the form with the input fields, we loop through the qryFields recordset to display the input fields:

```
<cfquery name="qryFields" datasource="myDataSource">
          SELECT
          Fields.ID,
          Fields.Name,
          Fields.DisplayOrder,
          Fields.ValidationID,
          Fields.Active,
          Validation.Name AS ValidationName,
          Validation.Active AS ValidationActive
   FROM
          Fields INNER JOIN
          Validation ON Fields.ValidationID = Validation.ID
   WHERE
          Fields.Active = 1
   ORDER BY
          Fields.DisplayOrder,
          Fields.Name
</cfquery>

<form action="myUpdate.cfm" method="post" name="form"
onsubmit="return validateInputs();">
<table>
<cfloop query="qryFields">
   <tr>
          <td>#qryFields.Name#</td>
          <td>
                    <input type="text" name="i#qryFields.ID#" value="">
          </td>
          <td>
                    #qryFields.ValidationName#
                    <cfif qryFields.ValidationActive is "0">
             *
                    </cfif>
          </td>
   </tr>
</cfloop>
   <tr>
      <td>
         <input type="submit" name="submit" value="submit">
      </td>
   </tr>
<table>
</form>
```

Notice the "*" in the output of the validation name. It just means that fields with a validation rule that is currently inactive show up with an "*". There are a number of ways to handle this situation, like not even showing a field if it has an inactive validation rule, but this is just one way to identify such fields.

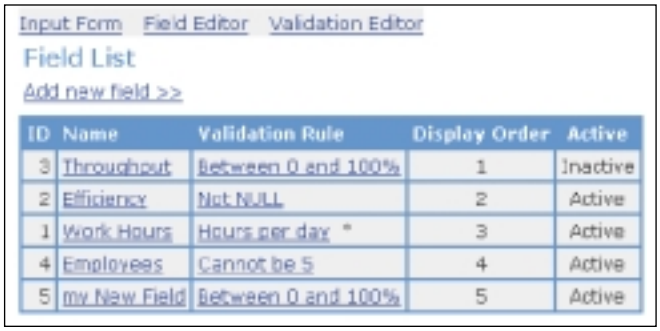
This shows how the input page with validations works. That's half the work right there. The other half is to create a system to manage everything.

The Interface

All the code for this utility, including a sample input form, is available online at sys-con.com/coldfusion/source.cfm. The utility consists of nine files. File names and descriptions are as follows:

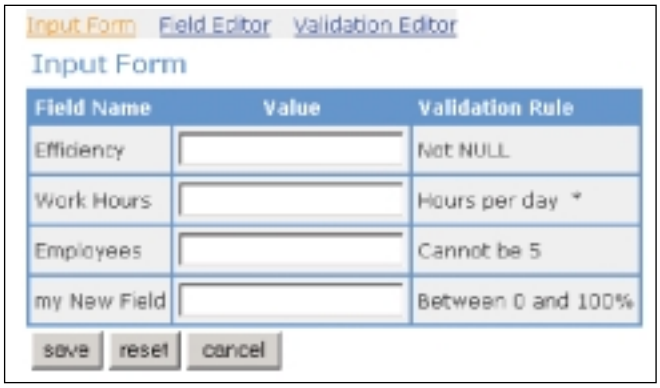
- **Listing 1: *Application.cfm:*** Sets up the data source and verifies that the required tables exist
- **Listing 2: *Menu.cfm:*** Displays menu options for the application

- **Listing 3: *MyEditor.cfm:*** Lists the available input fields

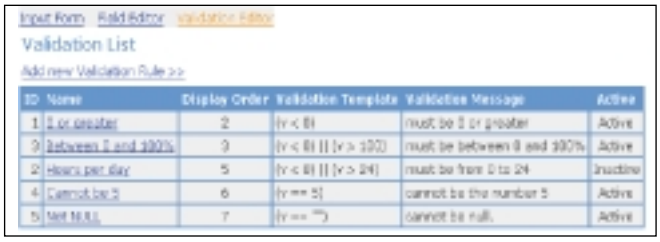


- **Listing 4: *MyEditorForm.cfm:*** Saves or updates available input fields

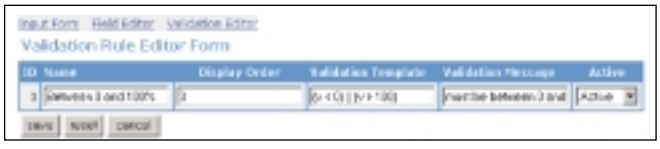
- **Listing 5: *MyUpdate.cfm:*** Shows the input fields that are stored in the database, and demonstrates how dynamic validation is performed



- **Listing 6: *MyValidation.cfm:*** Lists the validation rules



- **Listing 7: *MyValidationForm.cfm:*** Saves or updates validation rules



- **Listing 8: *Style.css:*** The style sheet used by the application
- **Listing 9: *TableScripts.SQL:*** The scripts that can be used to build the required tables (MS-SQL2000)

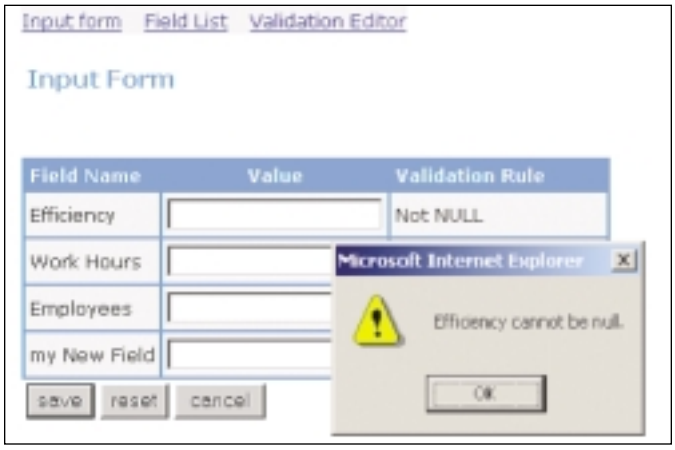
The first step in using the application is to install it into the location where you want to run it. Place all files in the same folder. You do not need the TableScripts.SQL file except to create the tables.

Next, build the tables from the TableScripts.SQL statements.

Create a data source in ColdFusion Administrator called "MyDataSource," which points to the database that contains the new tables.

The page to call first is myUpdate.cfm. If you do not already have data in the tables, you won't have any input fields at this time. You can go to the "Validation Editor" page to add validation rules, and then go to the "Field Editor" to create input fields and assign validation rules.

In my example, one of the fields I have defined is Efficiency, which has a validation rule that does not allow NULL values. When you are on the myUpdate page and you try to save without having anything in the Efficiency field, you will see a message that describes the validation rule for that input:



As you can see, the name of the field is automatically displayed along with the description of why the value is not accepted.

There are a number of things you can do to change how this system works. You can build some functions into the JavaScript validation area that will always be executed, for example, skipping validation for fields with blank values. However you decide to use it and modify it, this system gives you a great deal of flexibility to maintain what could otherwise be a time-consuming and error-prone method of maintaining fields and validations.

This demonstration concentrates on numeric values, but this can also be used for strings as well. The types of validations that can be performed are completely up to you. You may want to include helper files, for common tasks such as removing leading and trailing spaces from values before they are checked, to see whether or not some value was entered at all.

A Special Case

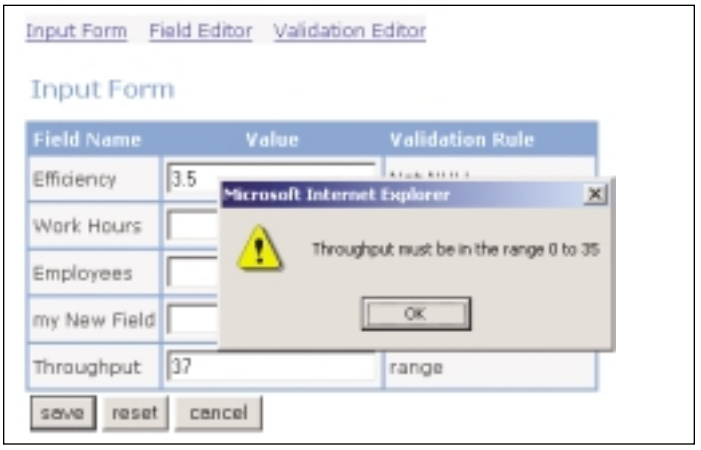
The previous examples show validations for inputs by themselves. There may be times when the validation for one input depends on another input. There is an easy way to do this within this dynamic system.

Let's say that we have a validation rule that says the value of an input must have a lower bound of 0 and an upper bound of another input field, times 10. For this, the format is basically the same as the other validation rules, except that when we create this rule, we must know the ID number of the other input field, which this new validation rule refers to.

For example, we are going to create a new validation rule that refers to the Field called "Efficiency". By looking at the "Field List" page, we see that "Efficiency" has an ID of "2". So, our validation rule looks like this:

```
ID                  5
Name                My New Rule
DisplayOrder        2
ValidationTemplate  (v < 0) || (v > document.form.i2.value * 10)
ValidationMessage   must be in the range 0 to " + document.
                    form.i2.value * 10 + "
Active              1
```

The only tricky thing about this is that you have to make sure to have the right number of double quotes in the ValidationMessage; otherwise your message will not be complete and may be missing characters. On the input form, you would see the message appear like this:



The validation for Throughput took the value of Efficiency, 3.5, and multiplied it by 10, which is 35. The validation rule found that the value for Throughput was outside of this range, so a message was displayed with the name of the field being validated, plus the exact description of why the value was not accepted. You might not need rules like this very often, but it's nice to know that you can do it if necessary.

About the Author

Jon Kurz is a senior software engineer at RR Donnelley & Sons Company, where he designs and builds enterprise applications using ColdFusion and Java. In addition to programming, he does research in artificial intelligence.

jon.kurz@rrd.com

Notes on the Environment: The following software was used in this demonstration: ColdFusion 5.0, MS-SQL2000, Windows Explorer 5.5

Managing Stale Updates

Three popular solutions

I receive lots of e-mail from lots of ColdFusion developers, and make every effort to respond to each and every one of them. Knowing the problems that ColdFusion developers are attempting to solve helps me know what topics to write and speak about. Indeed, the inspiration for most of my ***CFDJ*** columns comes from questions and comments from readers. So when I received two e-mails in one week asking about how to handle database locks when locks can't be used, well, the result is the column you are reading.

By Ben Forta

The Problem

Let's start with the problem. You write an application that allows users to update database rows. Simple, right? You allow the user to select the row to be updated, you provide an edit form, and then you create an UPDATE statement to save the changes. Simple.

Or is it? Consider this scenario:

- User A selects a row to edit and is presented with an update form.
- User B selects the same row to edit and is thus presented with the same update form.
- User B submits the form and saves the changes.
- User A then submits the form and saves the changes, overwriting user B's changes.

Not so simple after all.

This is a classic database problem, a situation that every database front end needs to address. The solution usually involves database locks. A database lock is just that, a lock implemented at the database level; locking a row prevents any other requests from making updates or even placing a lock, so using database locks, the prior scenario would play out like this:

- User A selects a row to edit, the row is locked, and user A is presented with an update form.
- User B tries to select the same row and is notified that the desired row is in use.
- User A submits the form, the changes are saved, and the row is unlocked.
- User B is then free to lock and edit the row.

This type of locking occurs within the database (not in the client), and most databases provide mechanisms with which to lock and unlock rows as needed.

But this is where things start to get messy – database locks are generally only of use within a database session, as long as clients are connected to the database. Database locks are rather useless when the client is a Web browser accessing the database via an application server. The page and script development model used in browser-based applications makes database locking very tricky – there is typically no clean way to lock a row in one page and then unlock it in another page. And even if there was, because Web browsers are not connected clients (connections are made and broken as needed), you'd run the risk of having locks that never get unlocked, page refreshes unable to access data as the same client already has a lock, and more.

This is not a ColdFusion problem, this is a fundamental limitation of Web clients – the statelessness of the Web makes traditional database locking not very useful. Database locks can work on a single page but not across pages. As for ColdFusion developers, <CFLOCK> can't help you here. For starters <CFLOCK> only locks within ColdFusion and would be useless if other clients were used too. But more important, <CFLOCK> can only lock within a request and cannot span requests (just like database locks). <CFTRANSACTION> has the same limitations.

So, are there any options? Actually, there are several, all usable and all imperfect. I am going to present three of these to you and explain the pros and cons of each.

Solution 1: Database Flags

One obvious solution is to implement your own database-locking scheme to be used in lieu of actual locking. This usually involves adding a flag field to each database table, maybe called "locked." You could set this field value to true (1) to place a lock and to false (0) to release a lock. You (and all developers) would need to check this field before any update (or delete) operations are attempted, and would be responsible to set the value yourself.

This is actually a very workable solution, one that can even work when multiple clients are used (not just your ColdFusion code).

It is also a very risky solution. Why? First of all, you and your developers will need to be sure to always check and set the flags, and it is dangerous to rely on developers remembering to do this. You could enforce the process by not allowing direct access to the tables and forcing the use of stored procedures, but that is a lot of work and the wrapper code is not trivial. But the biggest problem with this solution is that you'd need to implement a way to deal with orphan locks. What would happen if a user requested a row (locking it) and then closed the browser or went to another page? What would happen if the user tried to refresh the page and was told that the row was locked even though it was that user who placed the lock? The risk of orphan locks is very high, so you would need to implement a timeout mechanism of your own.

Solution 2: The 'Lock List'

Another option is to maintain your own lock list. In ColdFusion this could be an array in the APPLICATION scope that stores the primary keys of rows that are "locked." When a user wants to lock a row, your code would scan the array to see if that row's primary key was already listed as locked. If yes, you'd not allow the lock, and if not, you'd lock it by adding it to the array.

As a ColdFusion array the code would be easy to implement, and timing out entries would not be difficult either (you'd store the lock time with the primary key and would check for timeouts regularly, maybe prior to each scan).

But this is also a very dangerous solution. Why? Again, you and your developers would need to be careful to always scan the array, but you'd not really be able to enforce this process. But the biggest problem is that the lock list is ColdFusion specific – there is nothing to prevent another client from updating (or even deleting) the row while you have it "locked." This would make the solution useless unless you were absolutely certain that only your ColdFusion code made table changes.

Solution 3: Just Don't Do It

If database locks aren't usable, and manual locking schemes are too risky, then maybe the best option is to not even bother locking.

Database locking is based on the premise that users should never be able to reach an edit screen if that edit cannot be completed. But what if you did allow them to proceed with an edit even though it might cause a conflict? What if instead of worrying about locking so as to avoid dueling edits, you allowed the edit and instead notified the user upon form submission if the edit could not be completed? How could you do this? It's quite simple.

All you need to do is keep a copy of the pre-edit data (perhaps in SESSION variables), and then when the user submits the edit form, SELECT the row to be edited and check that it is still the same as it was when the edit form was first created (you could also check that it still exists and was not deleted). If the row contents match the saved values you would save the edits, and if not you would notify the user (perhaps showing the various changes and prompting as you see fit). Of course, the data could change even within the time between checking the record and proceeding with the update. <CFTRANSACTION> and its ISOLATION attributes can help with that, depending on the database.

This is a very workable solution. The beauty of it is that there are no locks to clean up or time out, and the solution works even if edits are made using other clients.

Is there a downside? Yes, first of all you and your developers need to remember to do this for each update; the process cannot be enforced. In addition, this solution would be highly inappropriate for applications where there is a frequent risk of concurrent edits (fortunately this is usually not the case). And finally, the ColdFusion code is not trivial (although a well-written Custom Tag or CFC could make the job easier).

Summary

The inability to use standard database locks is a real problem, and one too often overlooked by application developers. This is not a ColdFusion limitation; it is a byproduct of the very nature of the Web. Talented developers have created all sorts of solutions to this problem, and I've presented three popular ones above. Try them, experiment, and if you come up with creative solutions of your own I'd love to hear about them.

About the Author

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

ben@forta.com

Breathing Life into a Time-Tested Curriculum

Macromedia redefines course syllabus to teach new MX skills to developers

I recently reviewed Macromedia's new "Developing Rich Internet Applications" (***CFDJ***, Vol. 5, issue 2) course to make developers interested in adding Flash to their arsenal aware of this terrific new offering from Macromedia Training.

By Simon Horwith

It is equally important to note that Macromedia's other two ColdFusion classes – "Fast Track to ColdFusion" and "Advanced ColdFusion for Developers" (now called "Advanced ColdFusion MX Development") – were revamped to include material covering the new features and functionality in ColdFusion MX.

In this article, I will review the Advanced ColdFusion MX Development (ACFD) course – as it has undergone some major changes.

Before examining the course syllabus itself, I'll begin by stating two important facts. The first is that while some of the units cover topics that were also in the prior version of the class, these chapters have been updated for MX, and the walkthroughs and labs are much longer and more challenging (which is a good thing). I'll also mention that the course not only has longer exercises (and more of them), but more material. The prior version of the class covered nine units in three days whereas the new version covers eleven units in three days! What does this mean for the student?

ACFD, now more than ever, delivers an abundance of knowledge over its three-day duration. Each of the units uses walkthroughs and labs to emphasize the material – the average unit contains five walkthroughs and a lab (at the end), and the average exercise takes anywhere from 20–30 steps or more to complete. For students, the course is more hands-on than ever before. That said, let's examine the skills learned by ACFD students.

The topics covered in ACFD include but are not limited to:

- Understanding, creating, and manipulating data stored in query, array, and structure complex data object format

- Persisting data across pages with the application, session, and client scopes
- Creating user-defined functions to encapsulate commonly used functionality
- Building applications with the business logic and database access encapsulated in reusable ColdFusion Components
- Using persistent ColdFusion Components
- Understanding how to write custom tags and to use them to achieve code reuse
- Optimizing application performance with query-caching strategies
- How to deal with database concurrency issues by leveraging transactional processing
- Graphing data with ColdFusion
- Properly locking access to application resources
- Building structured exception handling within an application
- Indexing and searching both documents and record sets using the Verity search engine
- Exchanging data across the Internet using <CFHTTP>, <CFWDDX>, XML, and Web services

The course itself starts students off with an application (the Coffee Valley application as it should look upon completion of the Fast Track to ColdFusion course) that they will enhance throughout the course of three days. Students build a shopping cart application using ColdFusion Components and user-defined functions to manipulate the complex data

object (shopping cart) they have created. They also implement query-caching strategies and query-of-queries to boost the performance of Coffee Valley, add charting and search functionality, create a "next-n" interface product catalog, and practice other techniques to enhance the site's performance and functionality.

The end result is that students leave the class with an impressive skill set. First and foremost they develop an excellent understanding of how to work with complex data – a requirement for building robust applications with CFML – including how to work with the variable scopes used by the ColdFusion Application Server under the hood. They understand how to effectively use locking and caching to optimize performance and stability in dealing with databases, and how to leverage code reuse techniques (custom tags, UDFs, and CFCs) in order to be more productive and to better-architect applications. In addition to these fundamental skills, developers practice using Verity to search documents and database data, use error-handling tags to build self-repairing code, query existing recordsets in order to boost performance by minimizing trips to the database, and more.

Advanced ColdFusion MX Development is not a course for beginners. It is recommended that developers have three months of solid ColdFusion development experience under their belt before enlisting in ACFD, though anyone comfortable with the basics of CFML should be able to successfully complete the class.

Not only do I recommend this class to developers ready to take their skills to the next level, but also to those developers who already consider themselves advanced but want a good overview of how the new features in ColdFusion MX may impact on the development process. I also think this class is invaluable to anyone studying for the ColdFusion Developer Exam. The topics covered in ACFD cover most of the exam, and the course materials are without a doubt the best study guide I have seen to date.

More information can be found about ACFD and about registering for ACFD, by visiting Macromedia's training site at www.macromedia.com/go/cfdj_adcf/ or by visiting the Fig Leaf Software ACFD training page at http://training.figleaf.com/figleaftraining/Courses/Advanced-ColdFusion.cfm.

About the Author

Simon Horwith, senior consultant at Fig Leaf Software in Washington, DC, has been using ColdFusion since version 1.5. He is a Macromedia-certified Advanced ColdFusion and Flash developer and is a Macromedia-certified instructor. In addition to administering the CFDJList List Serve and presenting at DC-area CFUGs, Simon is a contributing author to Professional ColdFusion 5.0 *(WROX) and to* ColdFusion MX - The Complete Reference *(McGraw-Hill), as well as technical editor of* The ColdFusion 5.0 Certification Study Guide *(Syngress).*

shorwith@figleaf.com

Bring the Rich Internet Experience to Your Desktop Using Screenweaver MX

Build a taskbar app using Flash MX and CFMX

It's a week before your new ColdFusion application is set to go live, and your boss saunters in and asks, "If the client's browser is closed, and they aren't in their e-mail client, how will we notify them of new changes?" Of course, this should never happen in a well-managed project, but there is a relatively simple answer.

By Jason Clark

By Dominic Plouffe

When Macromedia launched their new MX suite of products, the theme was (and still is) "Building Rich Internet Applications." It's true that you can build rich Internet applications in the browser, and add life to your Web applications. But until recently, there was a disconnection between the user and the application upon exiting the browser. Typically, this separation is mitigated by e-mail, which could be considered a fairly ancient and noninteractive method.

Recently, at FlashForward 2003, Macromedia announced their solution for taking the rich Internet experience to your desktop. Macromedia Central is Macromedia's solution to extending the rich Internet experience beyond the browser. This new technology does more; it allows developers to build applications and distribute them over the Internet through Central's built in e-commerce system. Macromedia expects to ship this new product by summer 2003.

Macromedia Central may seem like an "industry first" to most users, but there is another tool that competes quite well: Screenweaver MX, which is an extension to the Flash MX player. Released in January 2003 by Rubberduck (a company based in the Netherlands), Screenweaver MX allows you to take your Flash-based applications and run them within their container. Your Flash applications behave just as they would in a Web browser, but they run right on your desktop. The ability to display Flash movies on your desktop already existed prior to these two applications using a "projector" application. Screenweaver may be similar in some ways to a projector, but in reality it is much more than that; think of it as a Flash container on steroids The end user experience is almost boundless when combining Screenweaver with the richness of the Flash player.

Under the covers, Screenweaver MX is a prebuilt container within which your Flash movie runs. To allow your Flash movie to interact with the container, Screenweaver exposes a rich set of APIs that are simple to use. These APIs enable you to do just about anything that your skill set can handle. Aside from using their prebuilt APIs to construct your applications, you can integrate with Windows DLLs and build COM objects that extend your applications' functionality.

So what types of applications can you build when combining Flash MX and Screenweaver? Taskbar-driven notes applications, instant messengers, product order systems, MP3 players, just about anything is the short answer. In this article, we'll build a small notification application that will demonstrate how to bring Flash, ColdFusion, and Screenweaver together.

Sample Notification Application

The sample application is a Flash movie that will be able to receive notifications from a server and will reside on your taskbar. In our example, we are using XML, created by a ColdFusion script, to transfer the notification from the server to the Flash movie, but alternatively, you could use Flash Remoting/Web Services, loadVariables, or Flash Communication Server. Please note that all the code to create the application in this article is not included, but rather, only code for the most important sections of this application is highlighted. There is a link at the end of this article to the complete application for you to download and try out.

Our sample notification application that we have created is an icon that sits on your taskbar, and checks for new notifications from a server every 10 seconds (see Figure 1). If a new notification has been received, sound is played. You can then click on the taskbar icon to view the notification text (which will be your Flash movie). The application consists of three different components: the Flash movie with the Screenweaver MX components

Figure 1: Windows taskbar icon

and ActionScript, an HTML/ColdFusion-based form, and a ColdFusion template that will be used to transfer the notification between the server and the Flash movie. Because the subject of this article concentrates on Screenweaver MX, more focus is placed on how to get your Flash movie running on your desktop, rather than on the HTML/ColdFusion components.

The ColdFusion Templates

Our example will use an HTML form that will be submitted to a ColdFusion script, which creates an XML object residing in the Application scope. Although we have used a form, we see the notification application being a ColdFusion Component/Web service that resides on your server. The component could then be initiated from anywhere on your Web site, or if you created a public Web service, anywhere in the world. Refer to Listing 1 for the HTML and the ColdFusion scripts that create the XML object.

The next template outputs the application variable (which contains the XML object) when it is called by the Flash movie (see Listing 2). As you will see, we will call this template from the Flash movie to retrieve the XML object.

The Flash Movie

In our sample application, we will only use the taskbar component that Rubberduck provides during the installation of Screenweaver MX. However, there are a number of other features and components that can be used, such as interacting with multiple Flash movies in different windows, common dialog boxes, and interacting with Windows DLLs.

Putting an icon on your Windows taskbar for your Flash movie is relatively straightforward (see Figure 2). Unfortunately for Mac users, Rubberduck has not released a Mac-compatible version of their Flash movie container; these examples will work only in Windows. The first step would be to drag the taskbar component that can be found in the Screenweaver Component Set section of the Component Panel onto your scene. Then, you can click on it and view its properties. Insert a name for the components (we have used the name "myTrayIcon" for our example), and set the Show on Startup property to True. This will result in the icon being displayed in the taskbar when the Screenweaver MX project is started. *Note:* you will not see the Windows taskbar icon if you test your application from the Flash editor because the Screenweaver MX component works only from its own container.

The next step is putting in the necessary ActionScript to make it all work. To see how this is done, review Listing 3.

The next part of our Flash movie is to create the script to read and retrieve the XML object created by the ColdFusion script shown in Listing 4. Although it may not be the best approach, we chose to use an interval that loads an XML object every 10,000 milliseconds, which means that every 10 seconds the Flash movie will poll the server for new notification. Because of the load that polling puts on a server we don't recommend you use this method, but it is adequate for our demo. We'll leave it up to the readers to think of a better technique

(Indeed, there is one option that may be worth exploring as a more appropriate means to support such a polling application. As Kevin Towes explained in his March 2003 ***CFDJ*** article, "Persistent Data Communications," at www.sys-con.com/coldfusion/article.cfm?id=583, you can have a Flash Communication Server application serve as the means to keep participating Flash client applications notified and synchronized with server data.)

Figure 2: Flash screen with the notification icon component

The Screenweaver MX Wizards

The people at Rubberduck have really done a superb job of developing a wizard to walk you through creating a Screenweaver MX project. Also take note that the Screenweaver MX application itself is a Flash movie, that runs in the Screenweaver MX container. It is itself a testament of how great their stuff works.

When you first enter the Screenweaver MX application, you are presented with three choices: New Project Wizard, Open a Project in the Wizard, and Switch to Editor Mode. The wizard will assist you in setting up your project quickly by presenting you with a number of simple questions. But, you'll need to use the Editor mode if you want to change any of the default wizard settings, or if you are doing a complex project dependent solely on Screenweaver MX.

When you are finished with the wizard, you are presented with the sixth and final step – the Build Section. Click the "Build My Project Now!" button to create your project right away. Building the application will create an executable. You can then run the executable to test your application. If you need to tune the Screenweaver MX settings further, you can do so by switching to the Editor. This can be accomplished by clicking on the "Continue in Editor" button. As soon as you are satisfied with your project, you can use the Screenweaver MX installer to deploy it.

Debugging Your Application

Because most of the Screenweaver MX features work only inside a Screenweaver container, you will have to use Screenweaver MX's debugging capabilities. If you encounter any problems upon launching your project, use the debugging features of Screenweaver MX to find the problem. Once debugging is enabled from the Editor mode, you'll notice an extra window that will open (next to the project window) after the project has been launched. The window is called the Debug Window.

In this window, various messages will be displayed from the Screenweaver MX engine. You'll also add your own messages

"It's true that you can build rich Internet applications in the browser, and add life to your Web applications"

by using the following command in your Flash movie:

```
swDebug.Trace();
```

Conclusion

This application is a simple example of how you can blend the power of a ColdFusion-driven application with the integration of Screenweaver and Macromedia Flash. Allowing users to interact with your Web application from their desktop will increase the user experience, and allow further productivity from your Web applications.

For more information on Screenweaver please visit www.screenweaver.com; for a complete ZIP file containing this sample application please visit www.fusetalk.com/cfdj/screen weaver1.zip.

About the Authors

Jason Clark, cofounder and CTO of Fusetalk Inc., has more than 10 years of experience in programming specifically and the IT sector in general.

Dominic Plouffe, cofounder and VP R&D of FuseTalk Inc., has been designing Web applications since 1997.

jason@fusetalk.com
dominic@fusetalk.com

Listing 1: (index.cfm)

```
<cfif isDefined("SEND_NOTIFICATION")>
  <cfif len(form.description) eq "0">
    You must enter a description before submitting the form.
  <cfelse>
   <cfoutput>
     <cfxml variable="application.notificationxml">
     <notification_example>
       <notification_text>#form.description#</notification_text>
          <notification_date>#dateformat(now(), "mm/dd/yyyy")# #timefor
           mat(now(), "HH:mm:ss")#</notification_date>
     </notification_example>
   </cfxml>
   </cfoutput>
   Notification has been sent.
  </cfif>
</cfif>

<form action="index.cfm" method="post">
Notification Description<br>
<textarea name="Description"></textarea><br>
<input type="submit" value="Send Notification"
name="SEND_NOTIFICATION"><br>
</form>
```

Listing 2: (check.cfm)

```
<cfsetting showdebugoutput="No">
<cfif isDefined("application.notification")>
  <cfoutput>#tostring(application.notificationxml)#</cfoutput>
</cfif>
```

Listing 3: Screenweaver ActionScript

```
//Initialize Screenweaver MX
swInterface.init();

//Initialize the Popup menu that will be displayed when
//clicking on the taskbar icon with the left mouse button.
var popupMenu =
[{   //The text of the menu item
       caption: "Exit",
      // The action that will happen when the menu item is clicked.
       callback: _root.quitApplication
    }]

//Specify the function that is triggered when the left button
//is clicked on the TaskBar icon.
myTrayIcon.onTrayLeftClick = function()
{
   //This command will show the window
   swWindow.show();
   swApplication.setForegroundWindow(main);

}

//This command will tell Taskbar Icon component to display the menu.
myTrayIcon.setMenu( popupMenu );
```

Listing 4: Get XML from Server

```
//Initialize the Server URL
_root.XMLUrl = 'http://www.myserver.com';

function getnotification() {
   function myLoad() {
     //Parse XML object
   }
   //Create the XML Object
   thisXML = new XML();
   thisXML.ignoreWhite = true;
   //Specify the function to run when the onLoad event is triggered.
   thisXML.onLoad = myLoad;
   //Load the XML from your server
   thisXML.load(_root.XMLUrl + "/check.cfm");
}

//Create the interval to retrieve the notification every 10 seconds.
setInterval( getnotification, 10000 );
```

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

Premiering
June 2003
at
JavaOne

www.sys-con.com

Millions of Linux Users One Magazine

Linux Business & Technology

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *Linux Business & Technology* is aimed squarely at providing this group with the knowledge and background that will allow them to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *Linux Business & Technology* will not feature low-level code snippets but will focus instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month will see a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features will include:

- *Advice on Linux Infrastructure*
- *Detailed Software Reviews*
- *Migration Advice*
- *Hardware Advice*
- *CEO Guest Editorials*
- *Recruiting/Certification Advice*
- *Latest News That Matters*
- *Case Studies*

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE $71.76

YOU PAY ONLY

$49.99

12 ISSUES/YR

**OFFER SUBJECT TO CHANGE WITHOUT NOTICE*

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

LINUX EDGE conference&expo

June 3–5.....LONDON
June 24–26.....BERLIN
September..HONG KONG
October.....CALIFORNIA

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR VISIT WWW.SYS-CON.COM

SYS-CON MEDIA

The World's Leading i-Technology Publisher

Teamstudio Screensurfer

Bring screen-based applications to the browser in HTML format quickly and painlessly

Every developer will at some point have to integrate his or her work with a legacy system running on a mainframe platform. These projects tend to be both painful and challenging due in no small part to the limited information-sharing capabilities of outdated, albeit stable, systems. XML and Web services didn't exist when some of these applications were written. The concept of browser-based applications was years away. Green-screen applications were the norm and dinosaurs ruled the earth (just kidding about that last part).

Reviewed by
Stephen Rittler
and
Greg Lang

Teamstudio has a product called Screensurfer that brings screen-based applications to the browser in HTML format quickly and painlessly. They do this through a 3270/5250 Web gateway controlled by easily modified ASCII text files. Using a series of wizards, a developer can quickly generate the necessary files and have a terminal session open and operational in the browser within minutes. Teamstudio calls this their "Screen to HTML" feature.

"Screen scraping" applications (such as Screensurfer) are not new. By extracting the data from the existing user interface, they enable a company to utilize these systems in new ways and repurpose the data for other applications. Screensurfer's ability to move screen-based applications to the browser also removes the requirement that system users have terminal emulation software installed on their desktop. This allows the application to be accessed by many more people than traditionally possible.

We are also given the option of running Screensurfer in a "co-server" environment. This is where Screensurfer sits between your application server of choice and the mainframe. ColdFusion is supported through a provided CFX tag. A COM object is available as well for those who are running ASP/VB code. Additional application servers (PHP, JSP, .NET) are supported through XML-over-HTTP calls. In a co-server environment, the screen-scraped data is returned in a useful format (usually a recordset) to the calling application server for processing and HTML markup.

Teamstudio packages an IDE tool with Screensurfer called Teamstudio Express. This IDE can be used to modify the ASCII files mentioned above, which contain HTML and specialized markup called Surferscript (we will discuss this momentarily) that are used to control the output of the scraped screen data. Teamstudio Express features syntax highlighting (for HTML and Surferscript), tag completion, a terminal emulator, and some debugging capabilities. We found the integrated "trace" functionality to be extremely helpful.

Incremental Migration

With Screensurfer in place, the transition from "terminal in a browser" to a co-server-supported Web application can be made one screen at a time. This feature is particularly attractive to developers who have to deliver a solution fast and then refine the look and feel of the user interface afterwards. It's an easier transition for the users as well; that same text interface that they've always used remains exactly the same when delivered via the screen-to-HTML engine.

Our Experience

We decided to install Screensurfer on a developer's workstation and try it out with a green-screen application running on an AS400. Installation of the Teamstudio package was quick and painless. The Teamstudio server, development environment (Express), and associated help files were installed. We selected all the default options for installation of Screensurfer. The only configuration we had to do on the AS400 side was to ensure that we had TCP/IP installed and running (in our case we did). If you do not have this available, you will need an SNA server like Microsoft SNA Server to handle all the SNA5250 to TN5250 bridging.

Our first test was to see our application using the "screen to HTML" functionality. We cracked open the tutorial and followed along, starting with opening up Teamstudio Express and creating a new project. We were quite pleased to find that the Teamstudio Express IDE interface is similar to the ColdFusion Studio interface my colleague and I were both quite comfortable with.

Teamstudio Express provides a wizard for configuring your application. All of the necessary Surferscript files were automatically generated by this wizard; we just had to hit the Compile button and were ready to go! Officially, that's all that we needed to do to get Screensurfer to drop our green screens into a browser. There are two ways to see the fruits of our (admittedly meager) labor. The first is by hitting the Launch button in Express (it looks like an exclamation point) and the second is to go to the default URL

(http://localhost:82/surfer/home/) in your browser. The host access page is shown in Figure 1.

Select the protocol, enter the host name, and press "Connect." If you entered all the correct information, you'll get your mainframe's logon screen. The mainframe logon screen is shown in Figure 2.

It doesn't get much easier than that. We just eliminated the necessity of each mainframe user having terminal emulation software installed on their workstation and opened up all of those applications to anyone with a Web browser. The total development cost was about 10 minutes of developer time. All of your green screen users will feel right at home, as the application interface is the same as in the terminal emulator.

In a production environment, Screensurfer should be installed on your Web application server or another Web-accessible server. Important to note is that Screensurfer operates as its own Web server. If you install it on a machine with an operating Web server, you will need to give Screensurfer a port number other than the one currently being used by your Web server software. We did not see any documentation that suggested we could use Screensurfer in conjunction with an external Web server.

To enhance the user experience, Screensurfer ships with functionality to handle keymapping. For example, a developer could map the F11 key on the user's keyboard to send the F11 command to Screensurfer (and therefore to your mainframe application). This is accomplished through the use of an ActiveX control (distributed via .cab file) for Internet Explorer users and a Netscape plug-in for Netscape users. If you opt not to use this functionality, the end user will have to click on an image map that appears next to the mainframe display screen to actuate function key commands. It's a nice touch that will be much appreciated by terminal emulator power users.

Let's Customize!

As we mentioned earlier, Screensurfer uses a collection of ASCII files that contain HTML and specialized Screensurfer markup called Surferscript. Surferscript is a tag-based language, just like CFML. These Surferscript tags allow the developer to control Screensurfer's path through your application interface screens. You do not have to use Teamstudio Express to modify Surferscript code, but we found it much easier than working in Notepad.

Surferscript tags are prefixed with "TE" similarly to how ColdFusion tags are prefixed with "CF". ColdFusion code is stored in ".cfm" pages, while Surferscript code is stored in ".stml" files. These ".stml" files are what get compiled by Teamstudio Express to form your Screensurfer application. A complete tag reference document is installed with Teamstudio Express and does a good job of explaining the functionality behind each tag.

For our second experiment, we decided to retrieve multiple screens of data. In our green-screen application, the user has to hit "page down" repeatedly to navigate through all the information. We were curious as to how Screensurfer accomplished this task.

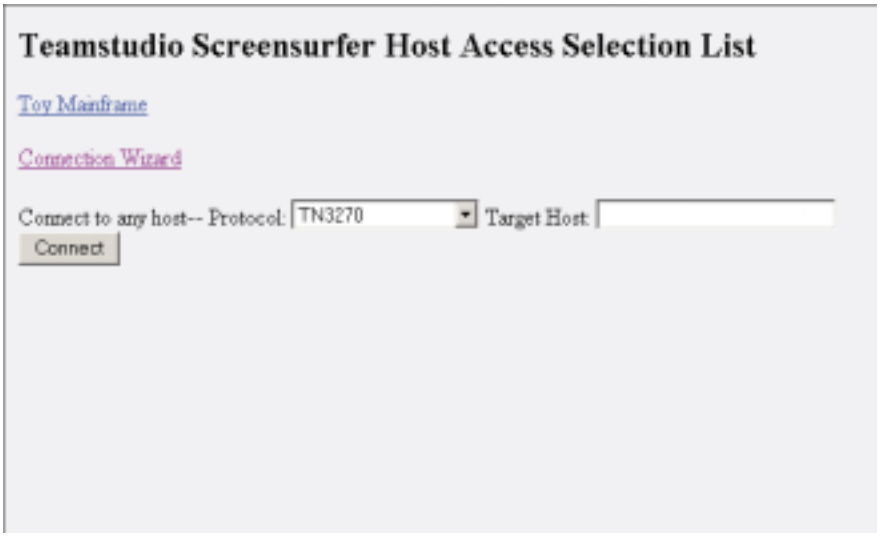


Figure 1: The host access page

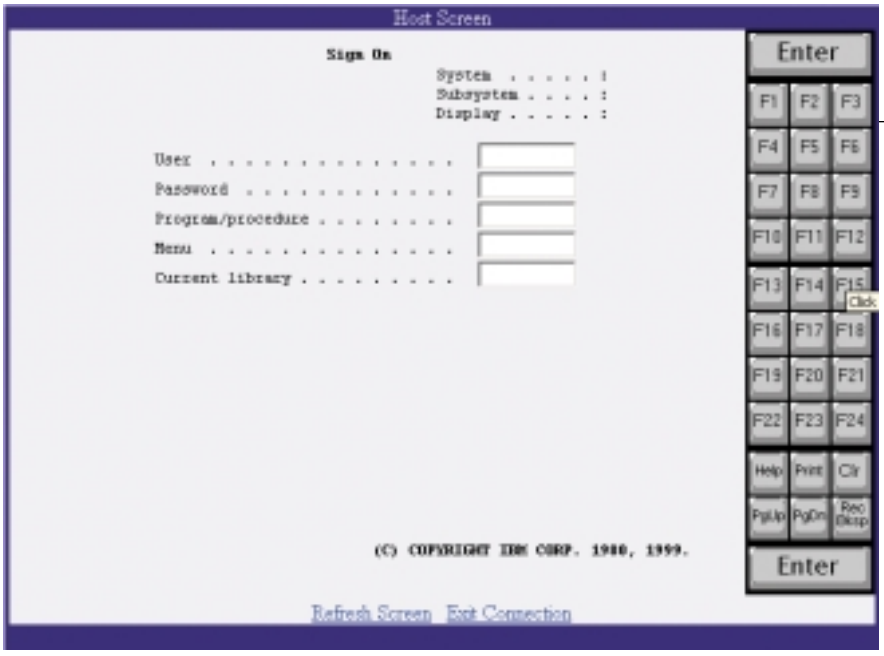


Figure 2: The mainframe logon screen

Fortunately for us, the tutorial coupled with the wizard in Teamstudio Express made the task of retrieving multiple screens of data fairly easy. We launched the wizard, marked up our terminal screens, and aside from a minor mix-up between page up and page down, we were good to go. Looking at the trace of the screens that Teamstudio processed made the task of debugging our code simple and also gave us a good look into what was happening behind the scenes.

Screensurfer was selecting the data from the marked regions on the AS400 display screens, sending a "Page Down" keystroke to the AS400, and repeating this process until the string "more…" no longer appeared on the screen. We noted that retrieving the data and generating the page output for display took slightly longer than an equivalent database call. Later on, we determined that compiling the Screensurfer application without the trace functionality enabled decreased runtime dramatically.

The Co-Server Environment

Teamstudio knows that developers need to integrate this data into their existing applications, not just access the same data the same way through pretty screens in a browser. Using the provided CFX tag and Surferscript, a developer is able to send to and retrieve data from the mainframe. We took our previous example (retrieving multiple screens of data) and modified it according to the tutorial instructions to return only the scraped data to ColdFusion.

This proved slightly more difficult to do and the instructions really only made sense after we completed the task. Most of our work was in adding custom Surferscript around the HTML output blocks in the STML files so Screensurfer would not be interjecting HTML into the return data stream. Once that part of the application was completed, we created a simple ColdFusion page containing the <CFX_Surfer…> tag call and a <CFDUMP>.

We compiled the Screensurfer application without trace functionality enabled. When we ran the ColdFusion template, performance was definitely acceptable. We compared the runtime of our <CFX_Surfer…> call with a stored procedure grabbing the same data from the same database and found that they were practically the same.

Observations

Developing a well-customized Screensurfer application will require a developer with a decent amount of familiarity with and comfort in both the green-screen and Web-based environments – that's why it took more than one person to write this article! We agreed that the tutorials would be somewhat more helpful if they provided a narrative logical overview of what we were going to do; using them as reference materials was difficult especially considering our lack of familiarity with Surferscript.

Some may ask what the point of the co-server environment is. As part of a larger migration scheme, the co-server environment is a key phase to migrating away from the green screen entirely. By implementing the "screen to HTML" functionality of Screensurfer alone, the potential user base of the application is increased and the desktop software

requirement (terminal-emulation software) is removed.

The users continue to work on screens in the browser that look the same as the ones they saw in their terminal emulation software. Meanwhile, the development staff can work on creating dynamic, data-driven applications that are fed by the Screensurfer co-server application in ColdFusion, ASP, JSP, or whatever supported language they choose. Once that is complete, system users can be smoothly migrated to a "Webified" user interface without leaving the browser. Ultimately, though, this ColdFusion/ASP/JSP application should move to the final phase of hitting the database directly, especially since database drivers are available for most mainframe databases. Screensurfer gives the developer the ability to smoothly transition the user base to a Web interface.

Conclusion

We were very happy with the screen-to-HTML features of Screensurfer. Sometimes referred to as "lipstick on a bulldog," the practice of screen scraping has been done for quite some time and usually at high cost. In relatively little time, we had opened up our normally closed AS400 application to the Web for far less than other solutions under consideration. In companies working hard to quickly Web-enable their business processes, Screensurfer can play a very useful transition role.

About the Authors

Stephen Rittler is a Certified Advanced ColdFusion MX Developer and manager of the Philadelphia Area ColdFusion User Group (www.pacfug.org). Stephen was a contributing author to the Advanced ColdFusion MX Application Development *book from Macromedia Press.*

Greg Lang is a Senior Application Developer and Project Manager with Amkor Technology in the Philadelphia suburbs. Greg has over 20 years of experience working with legacy systems and emerging technologies.

scr2@alumni.lehigh.edu
glang@amkor.com

Vitals

Teamstudio Inc.
900 Cummings Center • Suite 326T
Beverly, MA 01915
Phone: 800-632-9787
sendinfo@teamstudio.com
www.teamstudio.com

Test Platform:
Compaq Deskpro, 396MB RAM, 500MHz Celeron
Windows 2000 SP3
ColdFusion MX Updater 3

Supported Terminal Types:
3270, 5250, and VT100

Pros: Easy setup and quick terminal emulation in the browser; application server integration
Cons: Restricted to screen scraping only; cannot directly access database or RPG programs
Target Audience:
Companies looking to move their existing screen-based applications to the Web for use in e-business and e-commerce
Pricing:
Per server depending on number of users

Design Patterns in ColdFusion: Strategy Pattern

PART 3 OF A SERIES

Making algorithms interchangeable and easily extensible

For the past couple of issues we have been getting to know how other languages use object-oriented design patterns and how those patterns can be implemented in ColdFusion MX. We jumped right into the Template Method and Iterator patterns and this month we will explore the Strategy pattern.

By Brendan O'Hara

This month's pattern has some things in common with the Template Method pattern we reviewed in March (***CFDJ***, Vol. 5, issue 3). The Strategy pattern helps us encapsulate a series of related algorithms, or strategies, and gives us a single interface, known as the context, with which to work with them.

What Is a Strategy Anyway?

As I've mentioned previously, the book *Design Patterns: Elements of Reusable Object-Oriented Software* is the preeminent work on the subject of design patterns. Its authors, the so-called "Gang of Four," established the general intent of the Strategy pattern as the following:

> *Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.*

Okay. For once that is actually fairly easy to understand. We have a number of different calculations from which we want our program to be able to choose the right one on the fly.

In the example you will see that strategies allow us to follow two basic principles of design patterns, which were laid out by the "Gang of Four" in their book. The first, "Favor object composition over class inheritance," encourages object-oriented programmers to rely more on composing new objects (or in our case CFCs) from other objects and not to expect to solve all our code issues by extending existing objects. The second, "Program to an interface, not an implementation," encourages us to program our CFCs to work on common interfaces and not to hard-code too much into one implementation.

In the example, I will attempt to address both of these points. However, if you have not worked with CFCs or know nothing about inheritance or object-oriented programming, then you may get more out of this article if you read the previous articles in this series first.

Strategy Pattern: When and Why?

So you want to learn design patterns and need to know how to recognize when it is appropriate to use the Strategy pattern? Here are a few questions to ask yourself about possible Strategy candidates:

- Do you have related CFCs that differ only in their behavior?
- Do you need to use a varying algorithm or make a different calculation based on parameters?
- Does an algorithm use data that should be available only to the algorithm?

We will see an example implementation of the Strategy pattern that includes a "context" CFC that encapsulates the interchangeable strategies. This also provides a way to call a polymorphic function, which will exhibit one of many different behaviors. The Strategy pattern can also avoid exposing complex, algorithm-specific data structures.

The real question may be why should we use a Strategy pattern instead of a standard switch-case statement. Here are a couple of reasons:

- The code of the strategy's context CFC, or any of the other strategies, doesn't have to be changed when a new strategy is added.
- Coding a single Strategy, although polymorphic, is very simple. <cfswitch><cfcase> statements, especially when multiple switch-cases are nested inside each other, can become extremely complex and result in the need to evaluate many variables to determine what should be processed.

Strategy Pattern in the Real World

Recently I was contracted to build an online ticketing application for TicketLeap.com using ColdFusion MX

and Flash MX. One of the main tricky issues was the complex and varied calculations needed for pricing tickets. Tickets have a standard price and often have discounted pricing for children or senior citizens. Tickets also have a variable service charge and may charge sales tax in some cases. My first thought was to use design patterns and the Strategy pattern.

In our real-world but limited example, taken from the TicketLeap.com project, each Strategy will be its own object or CFC. This isn't unusual because in object-oriented programming programmers tend to think of everything as an object. This isn't 100% necessary but it gives us the flexibility to add additional CFC strategies that implement the same interface without changing a single line of existing code.

We are then establishing an interface with which to work with strategies and then programming to that interface as opposed to the specific file we are working on. This promotes polymorphism (the ability to override methods in derived CFCs) and code reuse, which are always good things to think about when programming CFCs.

First we start with the top-level AbstractStrategy.cfc, which defines the interface that all future Strategy CFCs will exhibit. Here we have a single method called Execute(), which in this case causes a <cfabort> to be processed since this CFC is abstract and should never itself be instantiated.

```
<cfcomponent displayname="abstractStrategy">
  <cffunction name="Execute" access="public">
    <cfabort showerror="This Method is Abstract
     and needs to be overridden">
  </cffunction>
</cfcomponent>
```

Next we have the derived PricingStrategy.cfc, which has an Execute() method that takes a price that is returned by default without any additional processing. This is the simplest case which, though it does nothing, is needed to make the polymorphic behavior of the execute method work for all other cases. You have undoubtedly come across situations like this before. If a parameter is not "" [Empty String] then do something. This acts the same but with a default "empty" implementation to avoid the <cfif> and make the method truly polymorphic. This is the case for a standard adult ticket.

```
<cfcomponent extends="abstractStrategy"
             displayname="PricingStrategy">
  <cffunction name="Execute" access="public">
    <cfargument name="price" required="true"
                type="numeric">
    <cfreturn arguments.price>
  </cffunction>
</cfcomponent>
```

Derived from PricingStrategy.cfc is ChildPricingStrategy.cfc, which has an Execute() method that takes a price argument and returns that price multiplied by the child discount rate. For this example I have hard-coded at .5 or 50%, but it could be passed in as an argument, or the ChildPricingStrategy.cfc could call some function to query it from a database.

```
<cfcomponent extends="PricingStrategy"
             displayname="ChildPricing">
  <cffunction name="Execute" access="public">
    <cfargument name="price" required="true"
                type="numeric">
    <cfreturn arguments.price * .5>
  </cffunction>
</cfcomponent>
```

In a real implementation we could have senior pricing and other pricing categories as well.

Another type of strategy is one that hides the implementation detail of its internal data structures. This is another CFC deriving from AbstractStrategy.cfc, which adds sales tax to a price for a particular zip code. This SalesTaxStrategy.cfc calls another CFC method, which returns the sales tax rate of the passed-in zip code. It then multiplies the price by the tax rate divided by 100 plus one.

```
<cfcomponent extends="abstractStrategy"
             displayname="SalesTaxStrategy">
  <cffunction name="Execute" access="public">
    <cfargument name="price" required="true"
                type="numeric">
    <cfargument name="zipCode" required="true"
                type="numeric">
    <cfinvoke component="TaxRates"
     Method="GetRateByZip"
     ReturnVariable="thisTaxRate">
    <cfreturn arguments.price * (1 +
     thisTaxRate/100)>
  </cffunction>
</cfcomponent>
```

Recall that I said earlier that the "context" in a Strategy pattern is the single interface that encapsulates interchangeable strategies. When we utilize one of our Strategy CFCs from within its context CFC, we are composing a CFC that is made up of other CFCs. We could also achieve similar results by extending the context CFC and implementing or overriding the Execute() method within it. This is less useful because we cannot develop other contexts that may contain these strategies without also creating a separate derived class for each strategy needed for each context.

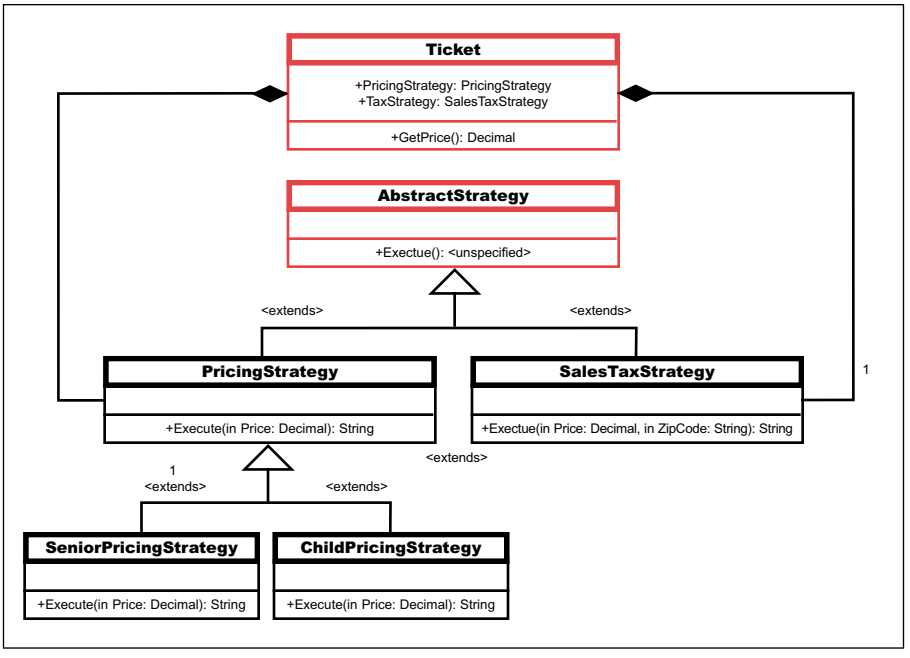


Figure 1: UML diagram showing various strategies

Favoring composition over inheritance makes for less code, and allows maximum code reuse and a single point for all changes to a particular strategy. The context object within which we will contain the Pricing Strategy and Tax Strategy will be Ticket.cfc, described below. Before we get into the code for the Ticket component let's look at the UML diagram shown in Figure 1.

First we have the AbstractStrategy.cfc, which is extended by a PricingStrategy.cfc and SalesTaxStrategy.cfc. PricingStrategy.cfc itself is extended by ChildPricingStrategy.cfc and SeniorPricingStrategy.cfc. What gets tricky is that the Ticket.cfc in the diagram is shown to have two attributes. These two attributes, TaxStrategy and PricingStrategy, can be either CFC instances or simply the textual name of the type of strategy they represent. In our case they are simply the names of the appropriate TaxStrategy and PricingStrategy. Let's take a look at the code for Ticket.cfc.

```
<cfcomponent displayname="Ticket">
  <!--- Instance Structure --->
  <cfset My = StructNew()>
  <cfset My.PricingStrategy = "">
  <cfset My.TaxStrategy = "">

  <cffunction name="init" access="public">
    <cfargument     name="TaxStrategy"
                    required="true"
                    type="string"
                    default="Sales">
    <cfargument     name="PricingStrategy"
                    required="true"
                    type="string"
                    default="">
    <cfset My.PricingStrategy =
      arguments.PricingStrategy>
    <cfset My.TaxStrategy =
      arguments.TaxStrategy>
  </cffunction>
  <cffunction name="getPrice" access="public">
    <cfargument name="price"
                required="true"
                type="numeric">
    <cfargument name="PricingStrategy"
                required="true"
                type="string"
                default="#My.PricingStra-
                         tegy#">
    <cfargument name="ZipCode"
                required="true"
                type="string"
                default="19341">
    <cfset priceObj = createObject("compo-
                     nent","com.TicketLeap.
                     #arguments.Pricing
                     Strategy#Pricing
                     Strategy")>
    <cfset taxObj =  createObject("compo-
                    nent","com.TicketLeap.
                    #my.TaxStrategy#
                    TaxStrategy")>
    <cfreturn taxObj.Execute(priceObj.
     Execute(arguments.price),
     arguments.ZipCode)>
  </cffunction>
</cfcomponent>
```

In the two createObject() method calls I am using the name of the specific pricing strategy or tax strategy to dynamically create the appropriate CFC instance. I am also using a fully qualified component path since these components are most likely not in the application's directory. The string com.TicketLeap.SalesTaxStrategy will look for a CFC named SalesTaxStrategy.cfc in the /com/TicketLeap folder.

Ticket.cfc has an Init() method, which mimics a Java constructor. It takes two strings for the TaxStrategy and default PriceStrategy that would normally be blank. It has an additional method GetPrice(), which calculates the price using the appropriate PricingStrategy and TaxStrategy CFCs. A simple implementation using these strategies is shown in Listing 1. This includes a sample query resultset of "cartitems" so you can test on your own without a database.

First we create our Ticket CFC and call the Init() method, passing in the string "Sales" for the TaxStrategy argument. We are defaulting to undiscounted pricing so we do not need to pass in a default PricingStrategy. The string "Sales" is used to indicate we are charging sales tax on this transaction. We then have a title row before we begin to loop through the query of "cartitems". Let's look at the fields returned by this query:

- ***Event:*** The textual name of the event the tickets are being sold for
- ***Price:*** The standard price before any discounts
- ***DiscountType:*** The name of an applicable discount (i.e., child, senior, etc.)
- ***ZipCode:*** The zip code for calculating tax for tickets, if applicable

For each row we then output the Event name, the price (dollar-formatted of course), the discount type if any, and then we calculate the actual price by passing the price, discounttype, and zipcode from the query to the Execute() method of the ticket.cfc we instantiated previously. We then output the actual price after discounts and sales tax are calculated. The output of MyCart.cfm is shown in Figure 2.

My client was delighted with the way the cart worked and my fellow developers were delighted with the way the Pricing Strategy worked. My team and I were most happy knowing we could easily extend the system to include any new pricing class with virtually no change to existing code. Let's take a look at what we have accomplished:

- We made the pricing system somewhat future-proof. It should work for

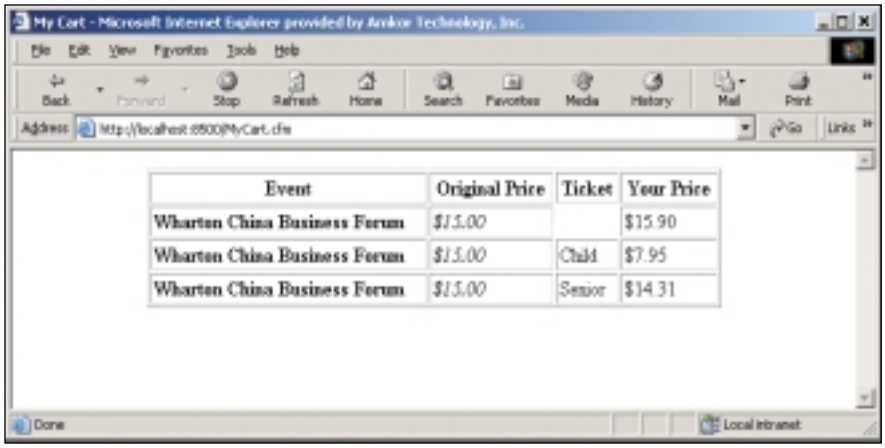


Figure 2: The output of MyCart.cfm

any conceivable discount type that may be added in the future.
- We helped ourselves with code reuse and polymorphism by segmenting out our code using a general strategy interface, which was inherited from AbstractStrategy.cfc.
- We favored composition over inheritance in implementing our strategies.
- We programmed to the strategy interface established in AbstractStrategy.cfc instead of to a specific implementation.

The actual finished page from the TicketLeap.com cart is shown in Figure 3.

Conclusion

Overall the Strategy pattern is extremely useful in working with families of algorithms and making them interchangeable and easily extensible. As with everything there can be drawbacks to any design or implementation. Strict adherence to or overuse of the Strategy pattern (or any pattern) can cause additional communication overhead between CFCs. Strategic use of all patterns is recommended.

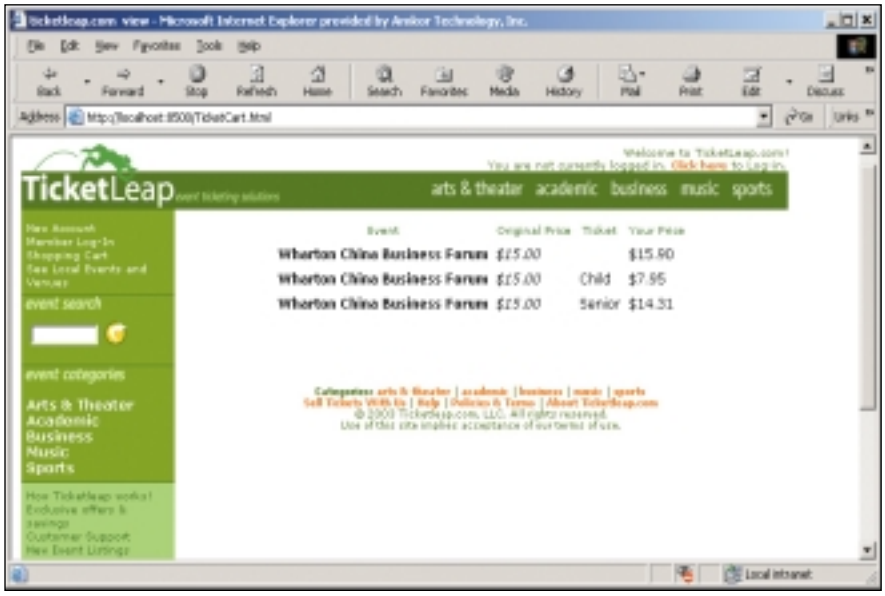


Figure 3: Actual finished page from the TicketLeap.com cart

About the Author

Brendan O'Hara is one of the coauthors of Advanced Macromedia ColdFusion MX Application Development, *published by Macromedia Press. Brendan has a Macromedia ColdFusion MX Developer Certification, along with Java and Linux certifications from Penn State University. He works as a ColdFusion architect and Java developer for e-tech solutions and Amkor Technology in the Philadelphia suburbs.*

bohar@amkor.com

Listing 1:

```
<!--- Test Implementation --->
<cfset cartitems = querynew("price,discounttype,event")>
<cfset queryaddrow(cartitems, 3)>
<cfset QuerySetCell(cartitems, "price", 15.00, 1)>
<cfset QuerySetCell(cartitems, "discounttype", "", 1)>
<cfset QuerySetCell(cartitems, "event", "Wharton China Business Forum",
                    1)>
<cfset QuerySetCell(cartitems, "price", 15.00, 2)>
<cfset QuerySetCell(cartitems, "discounttype", "Child", 2) >
<cfset QuerySetCell(cartitems, "event", "Wharton China Business Forum", 2)
>
<cfset QuerySetCell(cartitems, "price", 15.00, 3) >
<cfset QuerySetCell(cartitems, "discounttype", "Senior", 3) >
<cfset QuerySetCell(cartitems, "event", "Wharton China Business Forum", 3)
>
<html>
<head>
<title>My Cart</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<cfset ticketpricing = createObject("component","com.TicketLeap.Ticket")>
<cfset ticketpricing.Init("Sales")>
<table width="70%" border="1" cellspacing="2" cellpadding="2" align="center">
<tr>
<th>Event</th>
<th>Original Price</th>
<th>Ticket</th>
<th>Your Price</th>
</tr>
<cfoutput query="cartitems">
    <tr>
    <td nowrap><strong>#event#</strong></td>
    <td><em>#dollarformat(price)#</em></td>
    <td>#discounttype#</td>
    <td>#dollarformat(ticketpricing.getPrice(price, discounttype, zip
        code))#</td>
   </tr>
</cfoutput>
</table>
</body>
</html>
```

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

—continued from page 7

certainly did take a risk launching the site publicly, but went on to validate their reasons and explain just how much they learned from the experience. Christian went on to explain that the information they gathered in such a short period of time was invaluable not only to Macromedia but to Web developers in general, and recommended everyone read the beta 1 report. He also brought up another very good point – Macromedia not only wanted to beta test the code, but also the concept.

I then responded that beta 1 of the site kind of made me feel the same way I did when I first saw the <blink> tag 10 years ago; it was "neat" for the first 10 minutes, but its "flashiness" soon annoyed me and I discovered that it had a tendency to result in seriously ugly Web pages. Beta 2 of the site is out now, and is much better. The beta 1 summary report is one of the best educational tools I've ever seen based on a real-world example. The summary is now out for beta 2 (currently at www.macromedia.com/special/progress_report/beta2.html) and shows just how much Macromedia has done to better the user experience on their site.

There are a few things Macromedia has done that really need emphasis. First and foremost, they released the beta version of the site to the public. This was a brave and daring thing to do, and Macromedia should be applauded for this. The site made excellent use of surveys to gather information from visitors, and beta 2 showed that Macromedia really did listen to what we had to say. That's another thing that needs mention – hats off to Macromedia for taking a chance and staying on the ball by immediately addressing issues and striving to better those things that the community has spoken up about such as faster load times, easier navigation, etc.

Macromedia should be applauded for writing the beta version progress reports and making them publicly available so that we may all learn from their experience. I strongly encourage all developers, particularly those interested in developing rich Internet applications, to read the progress reports. As new versions are released, Macromedia is continuing to record their findings at www.macromedia.com/special/progress_report/. Surely, macromedia.com will further evolve in the months ahead.

Speaking of upcoming news from Macromedia, at the time of this writing Macromedia has just publicly announced their new product – "Macromedia Central." Macromedia Central promises to deliver one unified interface that pulls in and displays content from various sources of data on the Web. You can think of Central as a portal for which Flash developers can build portlets – a browser for Flash applications, if you will. This "browser," just like traditional Web browsers, resides on the client machine, as do the Flash applications it houses. Many of these applications will be capable of functioning without a network connection but will also leverage the Web when you're online.

The product will use Web services and XML to interact with servers on the network (Web) and to allow information to stream to its interface in real time. Not only will content be available on- and offline, but it will allow for the sharing of data between the information portals that a user has chosen to load into the interface. This product looks to allow Flash developers to take their skills and applications to a whole new level in terms of usefulness and functionality.

Not much more has been publicly said about Central just yet, but look for more information in the very near future. I'm sure that the product will create quite a buzz in the community in the months to come. Central is slated to enter beta toward the middle to end of April and for release sometime this summer. You can find out more in the news section in this issue and at www.macromedia.com/software/central.

Hynes Convention Center: Boston, MA

Web Services Edge 2003 East

INTERNATIONAL WEB SERVICES CONFERENCE & EXPO

Russ' Tool Shed: Russ Fustino shows how to use Visual Studio .NET

When SYS-CON Media's sister company, SYS-CON Events, began preparing last year for this spring's "Web Services Edge" Conference & Expo, one consideration was paramount: every effort in the nine-month preparation cycle should be geared toward making it indisputably the world's largest independent Java, .NET, XML, and Web services event.

That particular mission was accomplished on March 18–20, 2003, at the centrally located Hynes Convention Center in Boston, Massachusetts, when Web Services Edge 2003 East made its mark right from the get-go, with delegates from a wide variety of companies both technologically and geographically. Not only had they been attracted by the specific session tracks for Java, .NET, XML, and Web services, they had also come to take advantage of the all-day *i*-technology tutorials, whether it was the Sun Microsystems Java University, the IBM XML Certified Developer Fast Path, Russ Fustino's .NET workshop (Russ' Tool Shed), or Derek Ferguson's Mobile .NET tutorial.

The show opened with a very well-attended keynote from Oracle's John Magee, VP of Oracle9*i* Application Server. Magee stressed that the key to understanding why Web services, unlike its distributed-computing forerunners like COM and CORBA, is prevailing in the enterprise space is that Web services do more than merely enable interoperability between platforms and integration between applications – they also do so simply.

What drives their simplicity, Magee explained to the audience, is standards.

The afternoon keynote offerings on Day One of the conference were equally well received. First came a panel coordinated by the Web Services Interoperability Organization (WS-I). The WS-I is an open industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages, and the panel discussion took place against the backdrop of the WS-I Basic Profile 1.0, consisting of a set of non-proprietary Web services specifications. The working draft for this, the audience learned, was approved just four weeks before the conference.

But security, the panel agreed, was the primary priority. Now that corporations like Merrill Lynch and DaimlerChrysler have joined the organization, ensuring that everyone adheres to the same specification is more important than ever. Web services is moving beyond mere SOAP, WSDL, and UDDI toward addressing security, messaging, reliability, and transactions. Eric Newcomer, chief technology officer of IONA Technologies, emphasized the importance of the World Wide Web Consortium (W3C) approach to these challenges, an effort that centers on the W3C's Web Services Specification Effort.

The Web services keynote panel was quickly followed by the highlight of Day One for many of the delegates gathered in the keynote hall: an address by Miguel de Icaza, the impossibly young and extremely gifted founder and leader of the GNOME Foundation, cofounder of Ximian, Inc., and .NET expert extraordinaire – as anyone needs to be who leads a project designed to port .NET to the Linux operating system.

The Mono Project, as de Icaza's project is called, clearly fascinated the broad mix of developers attending the conference.

After explaining that GNOME – a desktop development platform and suite of productivity applications – is his compa-

ny's key focus and is mostly developed in C, C++, Python, and Perl, he went on to recount how for every new GNOME API (GNOME is component-oriented and supports many programming languages), GNOME developers needed to develop language-specific bindings. Thus .NET, which also addresses the multilanguage problem, was of immediate interest to de Icaza.

As soon as he learned about the .NET Framework, he told the spellbound audience, he got excited – a single Virtual Execution System for multiple languages, with a large and reusable factored class library, that was, in his view, just what was needed. As well as being a new way to do things, .NET's rich support for interop (COM, P/Invoke) meant you didn't have to rewrite everything all at once.

And so Mono was born: an open-source .NET Framework implementation.

It's based around the CLI ISO standard, de Icaza continued. It has a CLI-compliant execution system and a x86 JIT compiler. It's supported by Windows, BSD, Linux, and Solaris, and there has been lots of progress on the class libraries.

The Windows support, de Icaza said, was merely a function of the fact that 60% or so of Mono developers have a Windows background. Some of the code contributed to Mono was funded by Microsoft grants, he added.

At the end of his keynote address, scores of developers of every stripe got up from their chairs and surrounded de Icaza for further questions. The response to his good humor, rapid delivery, technical savvy, and sheer charm had been overwhelming and with his keynote, Web Services Edge 2003 (East) passed a significant milestone: no previous conference in the series had ever included so wide a range of technical content.

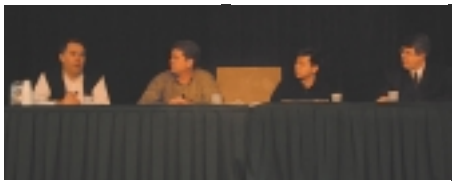
Day Two saw Sun's Mark Herring take the keynote stage and his mastery of the whole Web services paradigm was clearly in evidence. Extended coverage of both his Java keynote and the subsequent keynote address by Jesse Liberty are available on the main conference Web site, www.sys-con.com/WebServicesEdge2003East.

John Magee, Oracle: "Developing in a Services-Based World"

Mark Herring, Sun Microsystems: "Bridging the Gap Between WS-Myth and WS-Reality"

Miguel de Icaza, Ximian: "The Mono Project"

Jesse Liberty, Liberty Associates: ".NET Web Services"



WS-I Panel Discussion: "A Road Map for Web Services Standards"

.NET Panel Discussion: "Real-World .NET"

Java Panel Discussion: "The Future of Java"

The closing keynote discussion panel, which for many turned out to be the high point of the entire keynote program, was a wide-ranging and a sometimes heated debate about "The Future of Java." The whole intense and highly interactive hour exemplified very well how a SYS-CON *i*-technology conference program differs from that offered by any other conference organizer. This was panel discussion at its best.

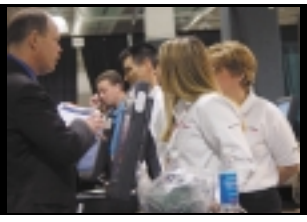
True to the enormously close links that ***Java Developer's Journal*** enjoys with the software development industry, the participants in this final panel at Web Services Edge 2003 (East) had come to Boston from far and wide. Sun's chief technology evangelist Simon Phipps had flown over from the UK and BEA's director of technology evangelism Tyler Jewell had traveled from Los Angeles. Sonic Software's VP and chief technology evangelist Dave Chappell may have nipped across from Bedford, MA, but Aligo's CTO Jeff Capone had flown in from San Francisco, and JBoss founder Marc Fleury had come up from the JBoss Group's company's HQ in Atlanta, Georgia.

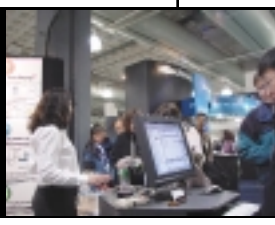
We fully expect the next Conference & Expo, Web Services Edge (West) in October, to be equally chock-full of the movers and shakers of the software development industry as it continues its headlong progress toward distributed computing with full application integration and interoperability.

All in all it was a marvelous conference, and the Expo hall was intensely busy from the moment it opened to the moment it closed two days later.

This is not the end of the Web services "story," nor is it even the beginning of the end; but March 18–20 in Boston's Hynes Convention Center may well have marked the end of the beginning.

Come join us for Phase Two…in October in California.





ColdFusion MX Migration Center

Thinking of upgrading to ColdFusion MX? Make sure you check out the new resource in the DevNet section of Macromedia.com. The new ColdFusion MX Migration Center includes a migration planning guide as well as links to useful articles, tech notes, performance briefs, and performance-tuning recommendations. Start planning your ColdFusion MX migration today, by visiting www.macromedia.com/go/cfmxmigration.

Sockeye Networks Helps Edge Web Hosting Improve Performance and Reduce Costs

(Waltham, MA) – Edge Web Hosting announced that they have deployed the Sockeye Networks GlobalRoute service to reduce the cost of upstream transit bandwidth while boosting Internet performance for their customers.

Edge Web Hosting offers mission-critical Web hosting services to enterprises. By automatically routing network traffic to avoid Internet congestion, GlobalRoute ensures that Edge Web customers receive superior network performance. GlobalRoute also balances traffic across Edge Web's Internet transits to minimize costs, taking into account different pricing models on upstream transits. GlobalRoute has helped Edge Web cut service provider costs by up to $6,000 per month.

"As a result of GlobalRoute, Internet brownouts and complaints about upstream Internet performance have dropped off almost completely," said Edge Web co-founder Vlad Friedman. "We no longer have to spend time hand massaging our edge routers to balance traffic across transits. GlobalRoute does this automatically, taking cost and performance into account. We save several thousand dollars on a monthly basis after the cost of GlobalRoute."

By continuously measuring end-to-end Internet traffic conditions, the Sockeye Networks GlobalRoute service makes routing decisions based on a comprehensive view of Internet performance. GlobalRoute also provides the ability to directly map ISP billing terms into routing actions within the network. The GlobalRoute dashboard and other reporting capabilities bring a new level of visibility to multiple-provider networks, enabling companies like Edge Web to constantly track results for cost and performance management.

"Hosting companies like Edge Web Hosting must continuously look for ways to differentiate service while controlling costs," said Brendan Hannigan, vice president of marketing and technology, Sockeye Networks. "GlobalRoute's ability to optimize performance while controlling costs enables Edge Web to achieve both objectives."
www.edgewebhosting.net
www.sockeye.com

Macromedia Unveils Future Direction for Internet Applications

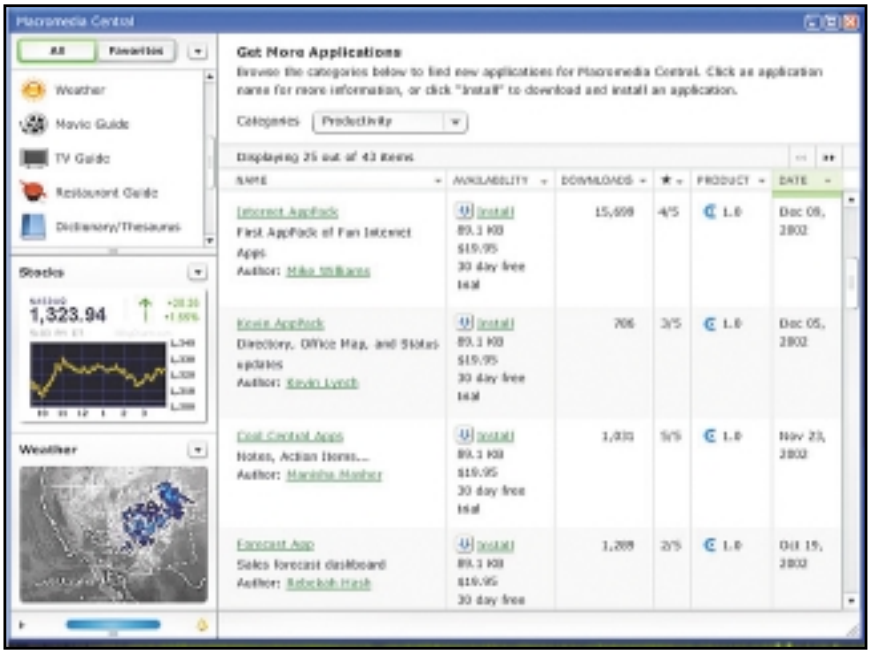
(*San Francisco*) – Macromedia, Inc., has announced Macromedia "Central," which extends Macromedia Flash "beyond the browser and provides a streamlined way for users to interact with information while freeing them from relying on an Internet connection." Macromedia Flash developers will be able to immediately leverage their existing skills to create and sell applications for this new environment, which is expected to launch this summer.

Macromedia Central takes advantage of the Internet's evolution into a worldwide platform that provides distributed data storage, distributed computing, and real-time communication. Central delivers an application metaphor to tackle this new world of information by providing instant delivery of applications, a try/buy infrastructure, support for occasionally connected computing, cooperative applications, and open data formats.

Macromedia Central builds on the power of Macromedia Flash Player for a better user experience by leveraging its ability to store data offline, consume Web services, and share information and activity across applications. Central has additional functionality to enable applications to work consistently offline, offer a persistent desktop presence, and provide notifications. The environment enables companies to deliver critical information outside of the browser paradigm with a higher level of experience and functionality.

"Central provides developers with a unique opportunity to combine their existing tools and skills to produce a new class of applications for this environment," said Kevin Lynch, chief software architect, Macromedia. "Together, we will be able to bring the world a better experience for Internet users."

Macromedia Central is perfectly aligned with current industry platforms such as the Intel Centrino mobile technology, which delivers an unwired computing experience while enabling extended battery life, outstanding mobile performance, and thinner and



lighter form factors. "Intel believes that the mobile experience is rapidly getting better with the deployment of Wi-Fi hotspots and the availability of Intel Centrino mobile technology–based notebook PCs," said Chris Thomas, chief strategist, Intel Solutions Market Development Group. "We're now challenging developers to use solutions like Macromedia Central to create innovative applications for the occasionally connected and unwired lifestyle."

Macromedia is already working with companies such as PriceGrabber.com to develop Central applications that enable users to aggregate comparative shopping information. "PriceGrabber.com enables users to make intelligent purchasing decisions by providing a central place for comparing products, prices, and reviews," said Lee Barth, manager of business development, PriceGrabber.com. "By delivering a Central application, users can access detailed product information from their desktop, review prices along with tax and shipping estimates, and get updates when the prices on those items go down."

Developers who create Macromedia Central applications will also be able to leverage an entire ecosystem to easily sell their applications. Central ships with try/buy functionality, as well as a transaction infrastructure to enable end users to easily purchase applications. Developers will be able to hook into the product's software update feature to ensure their users always have the latest version of their software. Central ships with an Application Finder to enable users to find details, cost, popularity rating, and other relevant information about specific applications. Macromedia will also deliver applications that ship with the product.

"Merging the best aspects of the Web with the way people actually use computers is a natural evolution," said Joshua Duhl, industry analyst, IDC. "A centralized place for personalized applications with a compelling business model to incent developers to create applications will lead to a new way for users to interact with Internet applications wherever they go whether or not they are online."

Macromedia Central is part of the Macromedia Information Convenience family of products, which provides solutions that enable organizations and business professionals to create and share information quickly, easily, and effectively without deep technical training.

Macromedia Central will be free and available for end users this summer. Some Central applications will be available for purchase. Developers interested in creating Central applications in advance of the product's availability can apply at www.macromedia.com/go/centralsdk for access to a Macromedia Central Software Development Kit. Macromedia DevNet Professional subscribers and Macromedia Alliance partners automatically qualify for beta participation.
www.macromedia.com/go/central

Macromedia Flash Player 6 Now Available for Pocket PC 2002

(*San Francisco*) – Macromedia, Inc., has announced the immediate availability of Macromedia Flash Player 6 for Pocket PC 2002, enabling developers and publishers to quickly and easily deploy rich content and applications to Microsoft Windows Powered Pocket PC 2002 devices.

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
ACTIVEPDF	WWW.CRUISE-N-LEARN.COM/COLDFUSION	866.GoToPDF	4
CFDYNAMICS	WWW.CFDYNAMICS.COM	866.233.9626	37
CFXHOSTING	WWW.CFXHOSTING.COM	866.CFX.HOST	31
COOLFUSION	WWW.COOLFUSION.COM	631.737.4668	15
EDGE WEB HOSTING	WWW.EDGEWEBHOSTING.NET	1.866.EDGEWEB	11
EKTRON	WWW.EKTRON.COM/CFDJ	603.594.0249	21
FUSETALK	WWW.FUSETALK.COM/NEW	866.477.7542	2
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM	877.215.HOST	45
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800.379.7729	52
IVIS TECHNOLOGIES	WWW.IVIS.COM	602.346.5045	3
LINUX BUSINESS & TECHNOLOGY	WWW.SYS-CON.COM	888.303.5282	35
MACROMEDIA	WWW.MACROMEDIA.COM/GO/DEVNET-CFDJ		25
MACROMEDIA	WWW.MACROMEDIA.COM/GO/CFDJ_TRAINING		51
NETQUEST	WWW.NQCONTENT.COM		9
NEW ATLANTA COMMUNICATIONS	WWW.NEWATLANTA.COM/BLUEDRAGON		6
PACIFIC ONLINE	WWW.PACIFICONLINE.COM	877.503.9870	29
PAPERTHIN	WWW.PAPERTHIN.COM	800.940.3087	39
SYS-CON SUBSCRIPTIONS	WWW.SYS-CON.COM/2001/SUB.CFM	888.303.5282	43
TERATECH	WWW.CFCONF.ORG/CFUN-03/	301.881.1440	19

Macromedia Flash Player is the leading rich client on the Internet, and Pocket PC 2002 users can download the player today at www.macromedia.com/go/flashppc.

For the first time, developers can publish standalone Macromedia Flash device content and applications that can play full-screen outside of the browser, such as games, applications, and kiosks. Developers interested in delivering executable Macromedia Flash projectors can license this version of the player for $499. There is no cost to end users.

"Macromedia believes experience matters, and that great experiences build great businesses," said Troy Evans, product manager, Macromedia. "This release is a key component of our mobile and devices strategy to deliver rich content experiences to platforms beyond the desktop."

"With Pocket PC, Microsoft provides developers with a robust and flexible platform to build the richest mobile applications found on a PDA," said Chris Hill, lead product manager, mobile devices division, Microsoft Corp. "From content and applications to standalone Macromedia Flash experiences, we are pleased to see that with Macromedia Flash Player 6, Macromedia is continuing to invest in the Pocket PC platform. Ultimately, this will give users a chance to experience compelling Macromedia Flash content on their Pocket PC."

With the availability of Macromedia Flash Player 6 for Pocket PC 2002, developers can incorporate video and application components within Macromedia Flash MX to deliver rich Internet applications, as well as bring e-learning, rich media, and gaming content to devices. For more information on Macromedia Flash Player 6 for Pocket PC 2002, go to www.macromedia.com/go/fp6_ppc.

Also available is the Macromedia Flash Player 6 developer kit for Pocket PC 2002. The kit includes a content development kit, an interface design policy kit, and a set of optimized user interface components for the Pocket PC. The components for the Pocket PC platform ensure that developers can easily author applications with consistent elements such as radio buttons and check boxes. Users can download the free developer kit at www.macromedia.com/go/ppcdevkit.

Macromedia Flash Communication Server MX 1.5 Now Available

(*San Francisco*) – Macromedia, Inc., has announced the immediate availability of Macromedia Flash Communication Server MX 1.5, including key functionality to extend its position as the premier platform for enabling compelling, interactive, rich media audio/video applications such as on-demand video, live event broadcasts, Webcam chat, and recorded video messaging. Macromedia Flash Communication Server MX 1.5 adds key features requested by customers to further enterprise deployment compatibility including support for HTTP Tunneling, Secure Sockets Layer (SSL), Linux, and MP3s; enhanced audio support and administration; and a new, improved model for adding capacity. For more information on the update, available for download, go to www.macromedia.com/go/commserver.

"Macromedia Flash Communication Server MX enables our customers to provide the best possible interactive rich media experiences on the Internet," said Pete Santangeli, vice president of engineering, Macromedia. "Corporations can now seamlessly integrate audio and video applications directly into their Web site with instant-on availability and unprecedented levels of interactivity."

Macromedia Flash Communication Server MX integrates support for streaming media, real-time collaboration, and multi-way video, audio, and text messaging into a single solution. With HTTP Tunneling and Secure Sockets Layer (SSL) gateway support, server traffic can now traverse firewalls and proxies and deliver secure information to ensure that everyone can experience the content. MP3 support enables developers to stream high-quality audio without requiring the user to download and play back audio files. Audio support is also improved in this update through better synchronization with video and better latency management for live streams.

With more hosting providers and ISPs supporting Macromedia Flash Communication Server MX, this new release includes better administration and management of virtual hosts. This server update adds support for Red Hat Linux 7.3 and higher. The new version also simplifies the addition of short-term capacity for sites that host events where short-term traffic spikes are expected. Ninety-day unlimited capacity packs, as well as the ability to stack capacity on existing servers, offer more flexibility and capacity planning options for developers.

CFUN-03

CFUN-03, the fifth annual ColdFusion conference, is set for June 21–22, 2003, in Washington DC. Last year's event was sold out with 300 attendees.

This year's conference has 18 nationally known speakers including Charlie Arehart, Ray Camden, Hal Helms, Michael Smith, Michael Dinowitz, Simon Horwith, Shlomy Gantz, and our own Robert Diamond, the editor-in-chief of ***CFDJ***! There will be four tracks with subjects for beginner ColdFusion, Advanced ColdFusion, MX Integration, and Empowed Programming (Fusebox, Project Management, etc.).

There will also be an exhibit zone where you can learn about the latest MX products. Last year's sponsors included Macromedia, PaperThin, New Atlanta, New Riders, ***ColdFusion Developer's Journal***, Open Demand, Fusion Authority, and Fusetalk.

Joe Hayes, an atttendee at CFUN-02, offered organizers the following feedback: "I wanted to drop you a note and say thanks for the CFUN-02 conference. I have been to many conferences in my 21 years of professional programming, and this by far was the best conference I have ever attended. The information provided by each and every presenter was top notch. I can't wait for CFUN-03 and if preregistration were available today, I would sign up today! This conference is an incredible value and to me was worth many times the price. Take care and thanks again for all the great speakers!"

CFUN-03 is run by MDCFUG and TeraTech (the winner of the ***CFDJ*** award for best consulting company). www.cfconf.org