SYS-CON

ANIMATIONS: THE FAB THREE P. 30

3.3

MX developer's journal

THE LEADING MAGAZINE FOR MACROMEDIA MX DEVELOPERS & DESIGNERS

volume 3 issue 3 2005 www.mxdj.com

take to the skies...

with captivate & flash

JetBlue uses MX for online training of its crewmembers

FIREWORKS
Uncovering Fireworks Masks

FREEHAND
Animated GIF with Vector Art

COLDFUSION
Printing Rich Document Formats

DIRECTOR
Dealing with the Flash Communication Server

$5.99US $6.99CAN

04

0 71486 02978 6

macromedia®

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

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

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

ARE YOU READY FOR

MX7

WE ARE

CFDynamics

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

CFDynamics, an innovative leader in ColdFusion hosting, caters to professional developers. We're here to help our clients use this brand new technology. While other developers waited patiently for the new version of ColdFusion, our clientele has already experienced powerful new features such as a full-feature reporting engine and flash forms through free hosting of Blackstone. CFDynamics is now ready for you to experience Coldfusion MX7.

Contact CFDynamics at 1.866.233.9626 and experience ColdFusion MX7.

*Blackstone is the code name for MX 7. Free BlackStone hosting is no longer available because ColdFusion MX7 is now released.

MX developer's journal

march 2005

7 Calling All MX Developers
You are all contributors
by charles e. brown

8 The MX Blogosphere
Around the MX world in 80 blogs
by mxdj news desk

58 vanguard
The Unusual Suspects
by james hill design

20 Take to the Skies with Captivate & Flash
JetBlue uses MX for online training of its crewmembers
by laura sehdava & chip moeser

34 A Model for Dealing with the Flash Communication Server in Director
Combining Director and the FlashCom Server
by nima azimi

42 Uncovering Fireworks Masks
Part 1
by kim cavanaugh

16 Flash, Web Services, and Data Binding
Part 2 - Using data binding through code
by darron schall

26 Between a Rock and a Soft(ware) Place
Streaming Web-based training development with Captivate & Flash
by bryan zug

30 Fireworks-Animated GIF with FreeHand Vector Art
It's fantastic, fabulous, and fun
by joanne watkins

46 Printing Rich Document Formats with CFMX7
How to print web pages in FlashPaper or PDF Format
by xu chen & sherman gong

50 Creating Better Forms Faster with ColdFusion MX 7
Macromedia makes time to give the cfform tags some love
by mike nimer

54 RollingYour Own Event Gateway
Writing and using event gateway in CFMX 7
by tom jordahl & jim schley

Seapine Software™
changing the world of software development

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

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

Surround SCM lets you...

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

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

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

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

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

Remotely access your source code repository from Dreamweaver MX.

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

Features:

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

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

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

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

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

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

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

Want to learn more?

Visit www.seapine.com/webdev to discover how Surround SCM can save you time and headaches. Be sure to download our white paper, **Change Management and Dreamweaver**, and learn why change management is the new must-have tool for web development.

www.seapine.com/webdev
1-888-683-6456

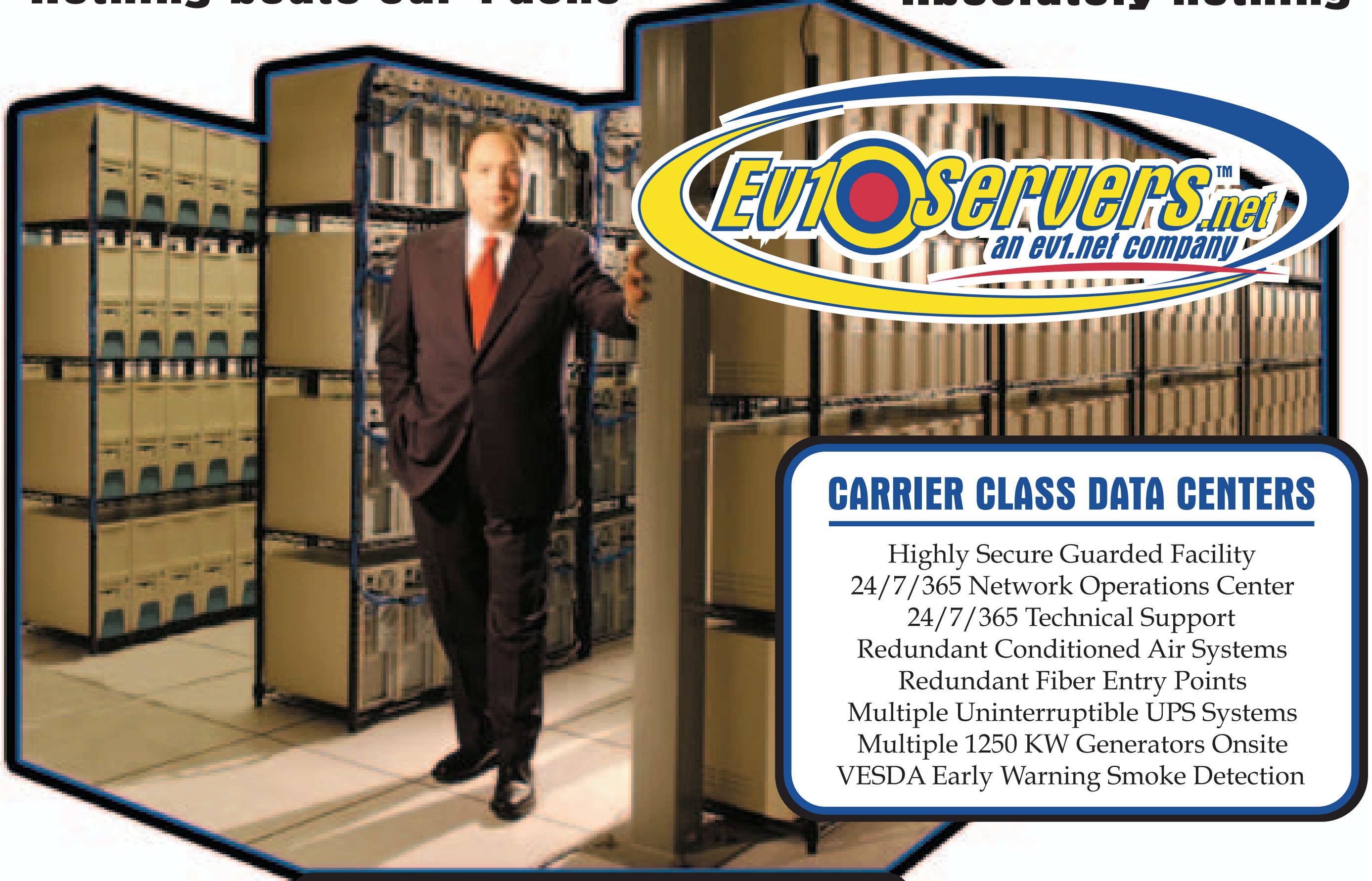
Surround SCM
Source Control

TestTrack Pro
Issue Management

QA Wizard
Automated Testing

macromedia ALLIANCE PARTNER

Nothing beats our racks **Absolutely nothing**



CARRIER CLASS DATA CENTERS

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

Robert Marsh, Head Surfer

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from **$299***

Instant Activation!

Dedicated Server

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

Over 20,000 Servers!

1-800-504-SURF | ev1servers.net

PLESK7 RELOADED
Preferred Control Panel

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

Calling All MX Developers

You are all contributors

by charles e. brown

When I took over this publication last September, I stated that I wanted to eventually change the tone of the publication so that the articles did not seem like "textbook" exercises. Instead, I wanted to take on a more pragmatic approach by showing how many businesses are using the Macromedia MX packages. This month is the fulfillment of that wish.

Last November, while going to New Orleans for the MAX conference, I had my first experience with JetBlue airlines. I was nothing short of astonished. In spite of its having lower fares than most other airlines, their waiting area had free Internet access, the seats on the plane were roomy and leather, each seat had its own television screen to see a variety or programming (including the news outlets), and the staff was friendly and courteous. No, I am not talking about first-class; I am talking about ALL the seats. Their economy fare offered more amenities than the first class of most airlines.

At the conference, I was pleasantly surprised to meet Laura Sehdeva (quite by chance) who works with Macromedia products for JetBlue. After telling her how much I enjoyed her company's airlines, I asked her to consider doing an article about their usage. It is with great happiness that that article will be this month's featured story.

In addition, Tom Green forwarded me an article from Bryan Zug and the Children's Hospital of Seattle.

Bryan is using Captivate to help implement training programs for the staff. If anyone needs proof of the importance of this, here's an excerpt:

"Imagine the differences between a critical medication order for a two-week old baby and a 17-year old teenager and you begin to understand the complexity of a pediatric implementation of this kind of system."

I applaud the work Bryan is doing and hope we, as a community, can help him more. They are doing fine work at this institution.

Over the next year, we will be featuring a lot of developers who use their talents for major companies and even a few very high-profile websites. However you don't have to work for a Fortune 500 company, or develop a major website, for your story to be important. When I teach my seminars, I get people ranging from a small insurance brokerage service to NASA; from an independent graphic designer to the Federal Reserve. All of them, without exception, teach me something new in the way they use the products.

For that reason I want ALL of you, the readers of this publication, to consider yourself the contributors. I would love to go to each and every reader and ask you to contribute. I don't care if it is something quite elementary or something involving very advanced techniques. If you use Macromedia products, we want to hear from your.

The benefits are many fold. However, consider this: by sharing your ideas with other users, you are helping the community to grow. In addition, when you write, you are also encouraging your fellow users to contribute. In the end, ALL will benefit.

Finally, in an age of networking importance, you are developing networks with your fellow users which will also benefit all.

It is the classic Win – Win scenario.

If anyone wants to write, please send me a proposal stating the topic, possible length in words, and when you will have it completed by. I will then forward it to the proper editor and set the proposal up. Our team of editors will guide you through the process.

This month we will be continuing our examination of the new ColdFusion MX7. I hope everyone has had the chance to download it from the Macromedia site and give it a test drive. If not, I strongly suggest doing so. Try it while reading the articles in this month's journal.

Finally, I want to thank author Nima Azimi for his unique article combining Flash Communication Server and Director. I feel that this is an important article that could open a lot of possibilities with the use of the two products. I will be anxious to hear feedback.

When I will I be printing your article?

Charles E. Brown is the editor-in-chief of MX Developer's Journal. He is the coauthor of Fireworks MX, Zero to Hero *and the auther of* Beginning Dreamweaver MX. *He also contributed to* The Macromedia Studio MX Bible. *Charles is a senior trainer for FMC on the MX product family.*
charles@sys-con.com

The MX Blogosphere

Around the MX world in 80 blogs

by mxdj news desk

If one of the true signs of a vibrant developer community is an active blogosphere surrounding a technology, then the MX suite of technologies certainly passes that test with flying colors. In case you're not yet active yourself, MXDJ brings you a comprehensive selection from some of the best known (and many of the not so well-known too). Don't forget that you can blog yourself now, too, at the *MXDJ* site – you can get started in just 3 minutes at http://blog-n-play.com.

Blog Topic: Dreamweaver

Dreamweaver Extension Virus?

By Tom Muck from "Blogging on MX" (http://tom.mxdj.com/)

Many users have problems with rogue extensions, bad extensions, or extensions that have been uninstalled improperly. Dreamweaver's Extension Manager is not entirely reliable when installing/uninstalling extensions. This is because of the way that modern operating systems work with multiple users, and because Macromedia, like other software companies are forced to comply with a multi-user configuration.

When you run Dreamweaver, you can be logged onto the operating system as yourself, the computer Administrator, or another user. When you install an extension, the files are not installed to the main program directory, as they were in the past with Dreamweaver 4 and earlier. They are installed in the local user directory, or in the All Users directory.

Danilo posted about the Dreamweaver configuration folder locations in a previous entry at CMXtraneous (http://www.communitymx.com/blog/index.cfm?newsid=27).

One of the major problems that has plagued extension developers from the beginning is when a rogue extension developer overwrites a Dreamweaver system file, or any Dreamweaver Configuration folder file. For example, some extension developers try to add additional functionality to the Dreamweaver Recordset. Instead of creating their own version of the Recordset files (such as Recordset.htm and Recordset.js), they modify the existing Dreamweaver files.

That works great if this is the only extension the user will ever install. If the user wants to install an extension from another developer, however, this could severely break the way that Dreamweaver works, because expected functionality might not be there or might act differently. Additionally, if two rogue extension developers overwrite the same file, one of them is out of luck because the second extension overwrites the first extension.

Make no mistake, when a Dreamweaver configuration file is overwritten by an extension developer, it is the equivalent of a virus, a spyware or other intrusive worm. The installation is now corrupt. Additionally, some extension developers are including a directive in the .mxi file that creates the extension package, putting a systemfile="true" directive for each system file that they overwrite. This causes problems as well, because when you uninstall the rogue extension, the bad system file is left in your Dreamweaver installation. The effect of this is that your Dreamweaver installation is corrupted. Dreamweaver is using the corrupted file rather than the file that Macromedia created.

Another problem exists: when Macromedia comes out with new versions of Dreamweaver, these rogue extensions are still being distributed, and are being used by users. Let's say, for example, that a new version of Dreamweaver has to change the Recordset.js file in order to work with some new functionality. A user installs an extension that overwrites this file using a version from the last version of Dreamweaver. The program is now broken with no way to recover other than deleting the local All Users or the local user folder, or just making it easy on yourself and reinstalling Dreamweaver.

More problems: If a Dreamweaver user decides he wants to modify a file in the Configuration folder, if the file has been overwritten by an extension developer, the user's changes are not respected.

An additional note: when I say that the file is "overwritten," it's not entirely accurate. In a modern operating system, the files are not overwritten in the main Program Files directory (Application directory on the Mac). The files stored in the All Users supersede the files stored in the Program Files directory. In other words, if Dreamweaver requires a Recordset.js file to run, the file will remain untouched in the Program Files directory, but a new version of the file is stored by the Extension Manager in All Users. However, to the uninformed user, this is the equivalent of a corrupt installation. The typical user does not know that there is a special hidden directory that contains rogue extension files.

What to do about this? Well, the first and most important step is to convince these rogue extension developers that what they are doing is harming the community. In their overzealous approach to adding functionality to Dreamweaver, they are basically spoiling the program for other extension developers, and setting the average Dreamweaver user up for a corrupted installation full of JavaScript errors every time they want to open a file. Secondly, Macromedia should step up to the plate and disallow extensions to be downloadable from the Exchange if the extension modifies core Dreamweaver configuration files. Lastly, users should write to extension developers that do this and demand that they stop.

It is a political issue, unfortunately. The extension developers who do this (I'm not naming names) insist they are trying to make Dreamweaver better. Some of us extension developers who have been around a while have always made an effort to extend Dreamweaver without ruining it for the next guy, though. Sadly, if it keeps up, more extension developers will do it and before you know it every extension you install will overwrite some other guy's file, and no extensions will work. What a mess that will be. . .

I issue a challenge to all Dreamweaver extension developers to stop this practice. It's already late, but we can minimize the damage going into the future. Extension development has come a long way since the early days of Dreamweaver, but if we can't develop extensions in a responsible way, it is going to impact the way that Dreamweaver users view extensions.

What to do if you have a corrupt installation? The first thing you should try is uninstalling the extension. Of course, that doesn't work always because of reasons I've stated. If it doesn't work, find the local Configuration folders mentioned above (not the main Program Files directory) and rename them or delete them. This will completely remove all extensions so that your Dreamweaver installation will be fresh again. Alternatively, reinstalling Dreamweaver should do this as well.

Blog Topic: MX Blogging

"Googlejacking"

By Cameron Childress from "Cameron Childress' Blog" (http://www.sumoc.com/blog/)

It all started with reading a thread on Slashdot about "Google Hikacking." For those just tuning in, here's a summary of what Googlejacking is (from http://clsc.net/research/google-302-page-hijack.htm):

"An explanation of the page hijack exploit using 302 server redirects. This exploit allows any webmaster to have his own "virtual pages" rank for terms that pages belonging to another webmaster used to rank for. Successfully employed, this technique will allow the offending webmaster ("the hijacker") to displace the pages of the "target" in the Search Engine Results Pages ("SERPS"), and hence (a) cause search engine traffic to the target website to vanish, and/or (b) further redirect traffic to any other page of choice."

Here's what happens:

1. Googlebot goes to scammer's site
2. Googlebot is given a 302 (redirect) to the victim's site
3. Googlebot indexes the victim's site as belonging to the original URL
4. Googlebot goes to the victim's site
5. Googlebot realizes this URL is already indexed and "belongs" (according to the Google code) to the scammer.
6. The victim's site gets lower rankings

Learn FreeHand!

Now you can ease any learning curve with nearly 200 movies and hands-on tutorials created by an artist who has used the program since it was introduced. Every menu item, feature, and tool is explained, and you're given a reason to use each of them.

...I learned that I've been working way too hard to accomplish various tasks since I was not aware of many of the time- and effort-saving tools.

The course looks great!

It has been very helpful.

I received the Casual FreeHand MX Course and love it

I am very impressed with your product.

Gain control over vector artwork, create layouts for brochures, booklets, instruction manuals, Web sites, Flash movies – and do it all at your own pace.

$45

Training from the trenches!™

www.brainstormer.org

Includes FREE FedEX domestic shipping and Global Priority Mail internationally.

as the page is not even indexed, the scammer's site gets a higher ranking.

A more detailed listing of how it works can be found in this Slashdot comment: http://slashdot.org/comments.pl?sid=143465&cid=12024264.
If you have a Macromedia-centric blog picked up by an aggregator and want to test if your blog has been Googlejacked, type the following into Google: allinurl:yourdomain.com

If some of the search results include pages containing the exact content and title as your blog, yet have a different domain, you've been Google Jacked. Admittedly, and per the descriptions in the above linked paged, this could be by accident some of the time, but the biggest offender in this case for me is edelina.com. Go ahead, type that domain into your browser (I'm not giving them any more link visibility by linking to them). It's a craptastic cornucopia of spammy junk, and they have a 302 redirect up an entire family of Macromedia-centric blogs. I've checked others, and we are virtually all there as far as I can tell.

Google Hijacking is worse than someone simply syndicating your blog content on their site because it's actually faking our Google to think that it is your site via 302 redirects, which mean "temporarily moved" as opposed to 301 which mean "permanently moved."

After a little more investigation, I found that the DNS host for edelina.com is Abadon Studios based out of Aliso Viejo CA. Searching for "Abadon Studios" in Google also reveals that they have a metric ton of other craptastic ethically questionable SEO domains for all sorts of things.

The worst part of it is that the slimeball behind all of this seems to be using Fusebox, which means he's "one of us."

If you are affected by this, instructions on what to do about it can be found posted by Google Guy on the Slashdot thread (http://slashdot.org/comments.pl?sid=143465&cid=12024599). It boils down to contacting Google's user support (http://www.google.com/contact/spamreport.html) and using the word "canonicalpage" in the complaint.

I would encourage anyone with an affected blog to make a complaint.

Blog Topic: Cold Fusion

Hide Your Errors

By Steve Bryant from "A Web Programmer's Exploration" (http://steve.coldfusionjournal.com)

I have noticed that a great many ColdFusion sites show the default ColdFusion error when something goes wrong. This is a bad idea for many reasons.

In the "Research-Based Web Design & Usability Guidelines" put out by Usability.gov, "Detect Error Automatically" was given an importance of 5 out of 5.
In his popular "Top 10 Web Security Tips" (http://www.sys-con.com/story/?storyid=46366&DE=1) article, Michael Smith listed "Have an error-handler" as his number one security tip.

In his article "Toward Better Error Handling" (http://www.sys-con.com/coldfusion/article.cfm?id=165), Charlie Arehart covers some techniques for error-handling in ColdFusion. As of the release of ColdFusion MX 7, a new method exists for handling errors in ColdFusion; the onError event of Application.cfc.

The onError event is only available if you are using Application.cfc. To get an introduction to Application.cfc, see my "Application Events:onRequest" (http://steve.coldfusionjournal.com/read/1059556.htm) blog entry or see the LiveDocs for Application.cfc (http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000692.htm).

To add the onError method to your Application.cfc, simply add code like the following to Application.cfc (replacing the comments with whatever code you want to run when an error occurs).

```
<cffunction name="onError">
    <cfargument name="exception"
      required="true">
    <cfargument name="EventName"
      type="String" required="true">
    <!--- Error handling code here
      --->
</cffunction>
```

Of course, ColdFusion will raise an exception (and run the code in onError) when it runs in to a <cfabort>.

In order to prevent that from adversely effecting your code, he suggests adding the following lines to the top of your onError method:

```
<cfif arguments.exception.rootCause
  eq "coldfusion.runtime.
  AbortException">
    <cfreturn/>
</cfif>
```

So now our onError method looks like this:

```
<cffunction name="onError">
    <cfargument name="exception"
      required="true">
    <cfargument name="EventName"
      type="String" required="true">

    <cfif arguments.exception.
      rootCause eq "coldfusion.
      runtime.AbortException">
        <cfreturn/>
    </cfif>

    <!--- Error handling code
      here --->

</cffunction>
```

So, now you can just take this code and add your own error-handling code (for example, display a user-friendly message and send yourself an email), right? Well, almost.

While ColdFusion MX 7 is a lovely product, it has a few bugs that still need to be resolved. One bug occurs if you are using jsessions and a session expires. It results in a "Session is invalid" error being displayed to the user.

My format is slightly different than Paul Kenney's (http://www.pjk.us/pjk/blog/index.cfm?event=showEntryForID&entry=9603C7B2-3048-28E9-DAD333835BEAFD8A), but the result should be the same. Here is a complete example of an Application.cfc from a site that used to use Application.cfm and OnRequestEnd.cfm:

```
<cfcomponent>

<cffunction name="onRequestStart"><
  cfinclude template="Application.
  cfm"></cffunction>

<cffunction name="onRequest">
    <cfargument name="targetPage"
      type="string" required="true">
    <cfinclude template="#arguments.
      targetPage#">
```

CommonSpot™.

Efficient Content Management

fast. easy. affordable.

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

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

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

For larger implementations, CommonSpot scales efficiently, delivering enterprise-level capabilities like replication, static content generation, multi-site clustering, personalization and custom authentication, across a diverse set of platforms.

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

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

Paper | Thin

```
</cffunction>
<cffunction name="onRequestEnd"><
  cfinclude template="OnRequestEnd.
  cfm"></cffunction>

<cffunction name="onError"
  returntype="void" access="public">
    <cfargument name="exception"
      type="any" required="true">
    <cfargument name="eventName"
      type="string" required="true">

    <cfset var Except = arguments.
      exception>
    <cfset var SessionExpiration =
      false>
    <cfset var redirectUrl= "/index.
      cfm">

    <cfif arguments.exception.
     rootCause eq "coldfusion.runtime.
     AbortException">
        <cfreturn/>
    </cfif>

    <cfif StructKeyExists(Except,"Root
      Cause") AND StructKeyExists
      (Except["RootCause"],"Detail")>
        <cfif Except["RootCause"]
         ["Detail"] CONTAINS "Session
         is invalid">
            <cfset SessionExpiration
              = true>
        </cfif>
    </cfif>

    <cfif SessionExpiration>
        <cfcookie name="JSESSIONID"
          value="" expires="NOW">
        <cfheader statuscode="302"
          statustext="Moved
          Temporarily">
        <cfheader name="location"
          value="#redirectUrl#">
    <cfelse>
        <!--- Code to output text to
          user and send email to site
          administrator goes
          here. --->
    </cfif>

    <cfreturn true>
</cffunction>

</cfcomponent>
```

This is the lazy route of leaving old code in Application.cfm and OnRequestEnd.cfm. I would suggest eventually moving the code around to take advantage of the structure of Application.cfc (hopefully more on that later).

Good luck!

Blog Topic: RIAs

RIA definition

By John Dowdell from "JD on MX" (http://www.markme.com/jd)

The term "RIA" was used frequently this past month, but it was only while reading the piping-hot comments at a Scoble article (http://radio.weblogs.com/0001011/2005/03/21.html#a9714) about "Microsoft Outlook Web Access being one of the first and best AJaX apps," that I remembered there actually is a functional definition of Rich Internet Application.

It's contained within that RIA whitepaper Jeremy Allaire wrote for the Macromedia site in March 2002, which was the first use of that phrase on the web (according to searches later done at recall.archive.org, which has since disappeared).

I've copied the first section which describes some of the requirements for rich internet applications:

"In the mid-1990s, explosive growth in the Internet and the World Wide Web drove widespread adoption of a new model for content and applications using personal computers connected to the Internet. Coined "thin-client" computing, this new model promised to lower the cost of developing and delivering applications to end-user desktops, customers and business partners, and to increase the range of application types that could be delivered. This model centered on a very thin client based on HTML, and powerful application servers that dynamically composed and delivered "pages" to web browsers.

So far this model has proven successful. However, it has also suffered from significant drawbacks and limitations, especially around the richness of the application interfaces, media and content, and the overall sophistication of the solutions that could be built and delivered. Indeed, for many traditional application developers, while the web has offered significant conveniences in terms of ease of deployment, the capabilities of the programming and user interaction models have forced users to suffer. In many respects, much of the web application development and deployment technology of the late 1990s has had to adapt to the challenges imposed by the architecture inherent in the web.

The Internet of 2002 will be different. End-users and businesses are demanding more from their investments in Internet technology. The ability to deliver true value to users is forcing many companies to look towards richer models for Internet applications; models that combine the media-rich power of the traditional desktop with the deployment and content-rich nature of web applications. Companies are also anticipating a growth in the use of web services, or reusable software components that are used as services over the network, and looking towards a world where applications will need to share functionality and data across many types of client devices. These trends are driving the industry towards next-generation rich clients.

This is the backdrop upon which Macromedia built Macromedia Flash MX and Macromedia Flash Player 6.
Before detailing the technical aspects of the Macromedia Flash MX client environment, it is important to note what we consider to be the crucial aspects of rich client technologies. Rich client technologies should:

- Provide an efficient, high-performance runtime for executing code, content and communications. The principle here is that the end-user experience of HTML-based web applications suffers from a variety performance related challenges. These include the request-response page rendering model; the need to dynamically generate large blobs of text for transmission of simple data; the lack of client-side data storage; the inability to easily invoke and use remote business logic, and even the basic graphics model of HTML. These all must be improved.
- Integrate content, communications, and application interfaces into a common environment. The end-user experience of the Internet today is fragmented into the HTML browser for textual content and basic application interfaces; multiple messaging clients for performing communications func-

tions; and multiple media players for handling audio, video, and other forms of media. Rich clients need to provide deep integration for all of these types of interaction.

- Provide powerful and extensible object models for interactivity. While web browsers have progressed in terms of their support for interactivity through the Document Object Model (DOM), JavaScript, and DHTML, they still lack the richness needed for building serious applications. Rich clients need to provide a powerful, object-based model for applications and events. This common object model must integrate user interface, communications, and system level services.
- Enable rapid application development through components and re-use. Rich clients should support powerful component-driven development, enabling both third party and corporate developers to easily reuse visual components to accelerate development, and give junior developers access to complex functionality. These components should integrate seamlessly into the designtime environment for ease of development.
- Enable the use of web and data services provided by application servers. The promise of rich clients includes the ability to cleanly separate presentation logic and user interfaces from the application logic hosted on the network. Rich clients should provide a model for easily using remote services provided by back-end components, whether hosted in an application server or accessed as XML web services.
- Embrace connected and disconnected clients. While many users have gotten used to having to be online and in a web browser to perform work, the reality is that most applications would benefit from the ability to be used offline on occasionally connected devices such as personal digital assistants (PDAs) and laptops. Likewise, many applications require support for persistent connections with two-way, notification-based communications. Rich clients must enable both of these types of applications to be easily built and deployed.
- Enable easy deployment on multiple platforms and devices. Internet applications are all about reach. The promise of the web is one of content and applications anywhere, regardless of the platform or device. Rich clients must embrace and support all popular desktop operating systems, as well as the broadest range of emerging device platforms such as smart phones, PDAs, set-top boxes, game consoles, and Internet appliances.

Macromedia Flash MX attempts to address and enable all of these opportunities."

Blog Topic: Aggregators

I Need a New Aggregator

By Christian Cantrell from "Christian Cantrell's Perspective" (http://www.markme.com/cantrell)

An unfortunate thing happened to me this morning. I have an old evaluation of NetNewsWire installed alongside the free version of NetNewsWire Lite which I use(d) extensively on a daily basis. This morning, when using Quicksilver to open NetNewsWire Lite, I accidentally opened the old expired evaluation version of NetNewsWire. For some reason, it over-wrote all my NetNewsWire Lite feeds with the default list of feeds that come with the application.

This is very much not a good thing. I'm sure I had well over 100 feeds pertaining to everything that interests me (mostly technology, but also some personal weblogs, watch weblogs, etc.). I have a backup from November that will allow me to recover many of my feeds, but my collection was constantly evolving and being refined, so the last four months of tweaks are gone.

Anyway, enough lamenting. I'm looking at this as an opportunity to start fresh with a new collection of feeds, new organization, and certainly a new aggregator. I really like(d) NetNewsWire, but I don't think I can bring myself to use it again. Additionally, I'm tired or waiting for the 2.0 version just to get Atom support (it's been in beta for a very very long time).

So my first question is what aggregators are Mac users out there using these days? I'm willing to go with either local or web-based. Once I settle on a new aggregator, I will then ask people to post some of their favorite blogs. I'm pretty sure I can have all my old feeds back with a couple of hours of searching and surfing, but I'd like to use this opportunity to find some new, more obscure feeds worth aggregating. That's a question for another time, though. First the aggregator.

Blog Topic: New Flash Book

Flash Anthology

By Dave Yang from "swfoo" (http://www.swfoo.com/)

SitePoint sent me Flash Anthology - Cool Effects & Practical ActionScript by Steven Grosvenor and asked me to write a review.

The back cover of the book says "best practice solutions to common Flash problems," which led me thinking it was a book on design patterns. However, in the "Who Should Read This Book?" section, it says the book is aimed at beginning and intermediate Flash developers and designers. The book is not for learning Flash MX 2004, ActionScript, OOP, or design patterns.

Instead, the book focuses on using ActionScript 1.0 "to achieve extensible, adaptable, and aesthetically pleasing results." There are over 60 ActionScript solutions, covering 10 chapters: Flash Essentials, Navigation Systems, Animation Effects, Text Effects, Sound Effects, Video, Flash Forms, External Data, Debugging, and Miscellaneous Effects.

I was a bit surprised to see no ActionScript 2.0, typed data, classes... etc., from a book published almost a year after Flash MX 2004 was released. Instead, I see most examples dealing with the prototype , scattering of _root , capitalization of method names in one section and lowercase elsewhere...etc.

To be fair, not all Flash projects require ActionScript 2.0, OOP or patterns. In fact, I'd think most typical Flash projects are still one-offs that have short life cycles. As Branden Hall says, sometimes pragmatic programming can be an ideal solution, especially for these quick one-off projects. Perhaps this book is for beginning Flash designers who work on small projects. SitePoint offers 30 days, risk-free, money back guarantee; so I guess it's worthwhile to give it a read if you're a beginner designer looking for ActionScript to create cool effects.

On a side note, for ActionScript 1.0 and effects, I'd suggest Robert Penner's book (http://www.robertpenner.com/) as an example of best practices.

Blog Topic: Flash

Make the World a Better Place with Flash

By Owen van Dijk from "MX Traveler" (http://ohwhen.typepad.com/)

Did you know that on your birthday 24.000 people will die of hunger and hunger-related causes? Last year I was involved with a special project that was part of the worldwide awareness-raising campaign First8. The heart of the campaign was a pocket-sized photography book with powerful portraits of people living and surviving in difficult and unacceptable circumstances throughout the world.

In September 2004 this book was delivered by mail to 25,000 people in positions of power and influence, addressed to each individual personally, by name. As the campaign was anonymous, there was no return address, only the address of this website. In the Netherlands, 1.6 million copies of this book, in mini-magazine format, were distributed along with various magazines. The book carries only this text:

Today you are one of more than 25,000 heads of state, ministers, members of parliament, monarchs, religious leaders, captains of industry, journalists and other influential people of 191 countries who hold this printed glimpse of our world.

For the first time in history we have the means to end poverty.

Today it's in your hands.

(www.first8.org)

There was no return address: the sender of the book was anonymous. The recipient was being asked to think about his or her own responsibility in the world: What can you do, using your position or status?

It also was a confrontational address to the members of the United Nations on the eve of the UN General Assembly in New York on 21 September 2004; a call to make an extra effort in the battle against poverty and to stick to agreements made in 2000, the Millennium Development Goals:

1. Eradicate extreme poverty and hunger
2. Achieve universal primary education
3. Promote gender equality and empower women
4. Reduce child mortality
5. Improve maternal health
6. Combat HIV/AIDS, malaria and other diseases
7. Ensure environmental sustainability
8. Develop a global partnership for development

I'm proud to be part of this project, and the whole team worked hard to translate the message from the book to an online experience. At the end of last year, this project won a prestigious advertising award; a Golden Epica (http://www.epica-awards.com/). The Epica awards are European prize for advertising and marketing-communication in film, print and on the web. Entries are judged on two critera: originality of the creative concept and the quality of the output.

Although it's always nice to win such a prestigious award, the real reward is the attention that comes with it for the actual website. I'd like you to take 5 minutes and have a look at the website (http://www.first8.org/first8.html) and think about your responsibility to make this world a better place.

FLASHFORWARD

& FLASH™ FILM FESTIVAL

SAN FRANCISCO 2005, APRIL 6-8
HERBST THEATRE

THE WORLD'S PREMIER FLASH EVENT

The 13th Flashforward conference and Flash™ Film Festival, the largest and longest running Flash user conference in the world, shares the latest in design, development, education and inspiration.

FLASHFORWARD 2005 FEATURES:

- 4 In-Depth Flash Workshops
- 25 One-Hour Seminars
- 20 Q&A Sessions
- 17 "Ask the Experts" Sessions
- 11 Technology Showcase Seminars
- Flash™ Film Festival
- Macromedia Keynote
- Exhibition Area
- Meet the Speakers and Flash Authors Reception
- Networking Receptions
- Exclusive Conference Workbook with Speaker Notes

GOLD SPONSOR

macromedia®

SILVER SPONSOR

VitalStream®

BRONZE SPONSORS

Curious Labs™ electricrain introNetworks macromedia® PRESS METALIQ New Riders Peachpit Press

MEDIA SPONSOR

ANIMATION MX developer's journal

REGISTRATION:

$599 through March 4
$699 through April 5
Additional $50 Macromedia User Group discount
$799 at the door

ADDITIONAL INFORMATION:

- Call toll free 1.877.4.FLASH.4
- www.flashforwardconference.com
- 1.805.640.6679 outside the US and Canada

Flash, Web Services, and Data Binding

Part 2 - Using data binding through code

by darron schall

In Part One (***MXDJ***, vol. 3 issue 2) we looked at one of three different ways to consume a Web service, here we look at two other ways and some of the pros and cons of each approach.

In this next example we'll explore using data binding in a different way from Pt. 1, by not relying on the Bindings tab of the Component Inspector panel. Instead, we'll create the bindings through ActionScript.

Example 2 - Creating Bindings Through ActionScript

To work through this example, we'll build the interface the exact same way as the previous example, leaving out the "add binding" steps. Either rebuild the interface following the previous instructions or take your complete Example 1 from Pt. 1 and simply remove the bindings on the components (by clicking the minus icon in the Binding tab).

Before we get to the ActionScript that needs to be written on the actions layer, first we'll examine the two classes that make data binding possible. Flash MX 2004 Professional includes the mx.data.binding.Binding class that does all of the heavy lifting. It uses a helper class mx.data.binding.EndPoint that is used to define the source and destination for the binding bridge.

The Binding constructor takes a source EndPoint that supplies the property to be bound, and a destination EndPoint that determines where that property is transferred to. Optionally, we can assign a formatter to transform the property from the source before it arrives at the destination. We can also optionally specify the binding as being two-way. Formatters and two way bindings are left as exploration pieces – discovery is a great learning mechanism.

An EndPoint is an object consisting of three main properties: component, property, and event. The "component" property of an EndPoint instance is a reference to the component the binding applies to, and the "property" property is a string defining what property is being bound or bound to. The "event" property is usually only necessary for "source" EndPoints, and is typically set to either "change" or "result." The "event" property can either be a single event as a string, or an array of events if multiple events are required.

Creating a new Binding instance and passing it a source and destination EndPoint is enough to get binding to work. Use the following two code blocks on the "actions" layer of the movie to hook up the data binding through script.

```
import mx.data.binding.Binding;
import mx.data.binding.EndPoint;

// bind the msg_ta text area to
   the msg parameter of the web
   service connector
var src = new EndPoint();
src.component = msg_ta;
src.property = "text";
src.event = "change"; // event is
   either a single event string,
   or an array of event strings

var dest = new EndPoint();
dest.component = msg2morse_wsc;
dest.property = "params";
dest.location = ["msg"]; // this binds
   to params.msg

// creating the binding is as simple
   as this:
new Binding(src, dest);
```

Alternatively, we can pass anonymous objects to the Binding constructor, bypassing the need for the EndPoint helper class. This makes for more compact code, although its *readability* can be argued to be either better or worse depending on your point of view.

```
// bind the result of the msg2morse_
   wsc to the text of the morse_ta
new Binding({component:msg2morse_wsc,
   property:"results", event:
   ["result"]},
  {component:morse_ta,
    property:"text"});

// bind morse_ta text to the msg
   parameter of the morse2msg_wsc
new Binding({component:morse_ta,
   property:"text", event:["change"]},
 {component:morse2msg_
   wsc, property:"params",
   location:["Morse"]});

// bind the result of the morse2msg_
   wsc to the text of the msg_ta
new Binding({component:morse2msg_wsc,
   property:"results",
   event:["result"]},
  {component:msg_ta,
    property:"text"});


// wire the buttons
msg2morse_btn.onRelease = function() {
 msg2morse_wsc.trigger();
}
morse2msg_btn.onRelease = function() {
 morse2msg_wsc.trigger();
}
```

In the first portion of the code, notice that we explicitly create an EndPoint for both the source and the destination. In the remaining code things are done a little different because using the EndPoint helper class is not a requirement. Rather, as long as we pass an object with the appropriate properties, the Binding class still functions correctly. With this in mind, we leverage anonymous objects to define the source and destination for the binding. Doing so makes the code more concise though it may or may not increase readability as that is a matter of personal opinion.

The only other interesting thing to note in the code is the additional use of the "location" property. If the "property" property of the component is complex (that is, not a simple value like 1 or "hello"), the "location" property points to the data field in the complex "property" object. In the example, the "param" property of the web service connector is an object and therefore complex, so we use "msg" as the "location" property in order for the Binding class to find the param.msg data field correctly.

Now, again, test the movie and see the magic. We've achieved the exact same results but instead of the "jungle code" approach of the Bindings tab, we've consolidated all of the binding code in one location. This makes the "magic" aspect a little bit more obvious, and should allow for easier updates in the future. Having a single location to look for certain information is a boon.

Like Example 1 in Pt 1, however, Example 2 is has a dark side as well. There's still a "magic" factor surrounding the web service connector supplied by Macromedia, and data binding adds a lot of unnecessary overhead behind the scenes. The former may not be that much of a concern, but if performance is paramount than the use of data binding needs to be reevaluated.

"But Darron, this article is about data binding, and now you're telling us not to use it?" Correct, sort of. Data binding has its place, but here's what you probably didn't know: every time you type a letter into the message area, a "change" event is generated because the "text" property has been changed; because of this, the glue that is data binding will update the parameter of the web service connector to be the text contained in the text field to keep them in sync.

While this doesn't sound like that big of a deal, consider the following scenario. Imagine typing the simple 4 letter word "test." In doing so, we've generated 4 change events, and set the "msg" parameter of the web service connector to be "t," "te," "tes," and finally "test." What a waste of time! In our case, we only care what the value of the text is when we invoke the web service because that is the only time the value is actually used. Instead of 4 calls, we really only need to make one.

Imagine, now, typing the above paragraph into the message area. Around 400 change events would have been generated, results in over 400 separate assignments, when, as previously stated, only one is required. Not just assignments, either. Don't forget about the overhead behind the scenes in dispatching and handling those change events. This wasteful aspect of data binding can be a major thorn in the side, especially when performance is on the line on slower client machines.

In reality, the wastefulness may not have a profound impact on your application. You may find that the benefits of binding outweigh the cost of using it. However, for those that want a more streamlined approach (sans data binding and web service connector "magic"), one final example needs to be gone through.

Example 3 - Not Using the Web Service Connector or Data Binding

In this last example we'll explore building the same application but without any relying on the web service connector and without the aid of data binding. This is somewhat of a code purist approach and is for those who like being closer to the underlying code.

For this example, start with a blank document and construct the interface almost exactly the same as before. Create two text areas and two buttons on an "interface" layer, give the components the appropriate instance names (msg_ta, morse_ta, msg2morse_btn, morse2msg_btn), and position and size them on the stage accordingly. Leave out the web service connectors since this example doesn't use them.

This next step is very important. By default, the classes needed to access web services are not included in all .swf files. In order to include these classes and leverage their functionality, we need to add them in the following manner.

From the menu bar, select Window -> Other Panels -> Common Libraries -> Classes. Drag the "WebServiceClasses" from the library that opens up onto the stage of your Flash document. Finally, delete the instance on the Stage that you just created. What just happened was that we added the web service classes to the library of our new movie, and by default they will now be included in our published .swf. At this point we can leverage the web service functionality through code.

"There's still a 'magic' factor surrounding the web service connector supplied by Macromedia, and data binding adds a lot of unnecessary overhead behind the scenes."

All that's left to do is put the following code on the "action" layer:

```
import mx.services.WebService;
import mx.services.PendingCall;

var morse_ws:WebService = new
    WebService("http://web221.
    area-18.server-home.net/Morse.
    asmx?WSDL");

msg2morse_btn.onRelease = function() {
 // calling a web service method
    returns a PendingCall
var pc:PendingCall = morse_
    ws.MsgtoMorse(msg_ta.text);
 pc.onResult = function(result) {
  morse_ta.text = result;
 }
 pc.onFault = function(fault) {
  trace("msg2morse fault: " + fault.
    faultString);
 }
}

morse2msg_btn.onRelease = function() {
 var pc:PendingCall = morse_
    ws.MorsetoMsg(morse_ta.text);
 pc.onResult = function(result) {
  msg_ta.text = result;
 }
 pc.onFault = function(fault) {
  trace("morse2msg fault: " + fault.
    faultString);
 }
}
```

In the code above we create a reference to the web service by passing in the WSDL location to the WebService constructor. Then, whenever a button is pressed we call the appropriate method and save a reference to the PendingCall instance it creates. Finally, the "pc" has two special methods that get invoked when either the results come back successfully (calling "onResult") or when an error occurs (calling "onFault").

Summary and Final Thoughts

As you can see, there is more than one way to accomplish a task. We've built the same application in three distinct ways, each with their pros and cons.

The first example requires the least amount of code. Data binding is leveraged through the Bindings tab in the Component Inspector panel, and two WebServiceConnectors control access to the remote web service.

The second example improves upon the first by placing all of the Data Binding code in a single location, but has the negative side effect of requiring more "hand coding."

The last example is the "down and dirty" approach that uses the WebService and PendingCall classes directly without the need for the WebServiceConnector, and rids us of the wasteful aspect of using data binding.

Each example can be the "right" way to code something, depending on the application requirements and the person doing the coding. I'm a fan of the last example, but you're welcome to disagree.

From here, I encourage you to explore the further reading links. Find a web service that interests you and try building a small application that uses it. Who knows, you might even surprise yourself!

Further Reading

- *XMethods - Web Service Directory:* http:www.xmethods.com/
- *RemoteMethods - Web Service Directory:* http://www.remotemethods.com
- *Flash Developer Center: Web Services Articles:* http://www.macromedia.com/devnet/mx/flash/webservices.html
- *Using the Flash MX 2004 web service classes:* http://www.flash-db.com/services/tutorials/mxclasses/mxwebservices.php
- *Consuming Web services in Flash MX:* http://uk.builder.com/architecture/web/0,39026570,20282917,00.htm
- *Understanding Web Services: A List Apart:* http://www.alistapart.com/articles/webservices/
- *Web Services - An Executive Summary:* http://webservices.xml.com/pub/a/ws/2002/04/12/execreport.html
- *Flash TechNote - External data not accessible outside a Macromedia Flash movie's domain:* http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_14213

Darron Schall is an application developer interested in all things programming, from ActionScript to XML and everything in between. He has a BS in Computer Science from Lehigh University, and maintains a Flash-related weblog at www.darronschall.com.

SYS-CON MEDIA
President & CEO
Fuat Kircaali, 201 802-3001
fuat@sys-con.com
Vice President, Business Development
Grisha Davida, 201 802-3004
grisha@sys-con.com
Group Publisher
Jeremy Geelan, 201 802-3040
jeremy@sys-con.com

ADVERTISING
Senior Vice President, Sales & Marketing
Carmen Gonzalez, 201 802-3021
carmen@sys-con.com
Vice President, Sales & Marketing
Miles Silverman , 201 802-3029
miles@sys-con.com
Advertising Sales Director
Robyn Forma, 201 802-3022
robyn@sys-con.com
Advertising Sales & Marketing Manager
Dennis Leavey, 201 802-3023
dennis@sys-con.com
Advertising Sales Manager
Megan Mussa, 201 802-3023
megan@sys-con.com
Associate Sales Managers
Kristin Kuhnle, 201 802-3026
kristin@sys-con.com
Dorothy Gil, 201 802-3024
dorothy@sys-con.com
Kim Hughes, 201 802-3025
kim@sys-con.com

PRODUCTION
Production Consultant
Jim Morgan, 201 802-3033
jim@sys-con.com
Lead Designer
Louis F. Cuffari, 201 802-3035
louis@sys-con.com
Art Director
Alex Botero, 201 802-3031
alex@sys-con.com
Associate Art Director
Richard Silverberg, 201 802-3036
richards@sys-con.com
Assistant Art Directors
Tami Beatty, 201 802-3038
tami@sys-con.com
Andrea Boden, 201 802-3034
andrea@sys-con.com

SYS-CON.COM
Consultant, Information Systems
Robert Diamond, 201 802-3051
robert@sys-con.com
Web Designers
Stephen Kilmurray, 201 802-3053
stephen@sys-con.com
Matthew Pollotta, 201 802-3054
matthew@sys-con.com
Online Editor
Martin Wezdecki 201 802-3045
martin@sys-con.com

ACCOUNTING-
Financial Analyst
Joan LaRose, 201 802-3081
joan@sys-con.com
Accounts Payable
Betty White, 201 802-3002
betty@sys-con.com
Accounts Receivable
Gail Naples, 201 802-3062
gailn@sys-con.com

EVENTS
President, SYS-CON Events
Grisha Davida, 201 802-3004
grisha@sys-con.com
National Sales Manager
Jim Hanchrow, 201 802-3066
jimh@sys-con.com

CUSTOMER RELATIONS
Circulation Service Coordinators
Edna Earle Russell, 201 802-3081
edna@sys-con.com
Linda Lipton, 201 802-3012
linda@sys-con.com
Monique Floyd, 201 802-3082
monique@sys-con.com
JDJ Store Manager
Brundila Staropoli, 201 802-3000
bruni@sys-con.com

SYS-CON MEDIA

Matt Liotta

Advanced CF

Deployment/Platform

Simon Horwith

Michael Dinowitz

Tim Buntel

Charles Arehart

Christian Cantrell

Jeff Peters

Accessibility / Usability

Manager

Empowered Programming

Raymond Camden

coldfusion

Steve Drucker

Michael Smith

Ben Forta

Sean Corfield

MX Integration

Lifecycle

Hal Helms

Dave Watts

Geoff Snowman

Shlomy Gantz

Boot Camp

the premier coldfusion

United

technical conference

June 29 - July 1, 2005

Washington DC Area

7th Annual ColdFusion Conference

3 full days!

New Venue

75% Bigger

www.cfunited.com

TeraTech

New Atlanta

CFDynamics

Fog Creek

Paper Thin

AboutWeb

MX developer's journal

ColdFusion Developer's Journal

Produced by
TeraTech
Programming
405 East Gude Drive
Ste 207
Rockville MD 20850
www.teratech.com
301.424.3903
info@cfunited.com

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

Ben Forta

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

Chuck Hoffman

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

Daniel Gregorio

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

Kathleen Ballard

take to the skies...

With increasing frequency, employees (called "crewmembers") at JetBlue Airways are being asked to fulfill some of their training requirements online. The aviation industry is policy-intensive, and airline employees – everyone from flight attendants, to pilots, to customer service and ground crew personnel – are required to update their training each year on certain regulatory topics. To meet these requirements, JetBlue employs a blend of classroom instruction, on-the-job training, and eLearning.

by laura sehdeva & chip moeser

JetBlue uses MX for online training of its crewmembers

with captivate & flash

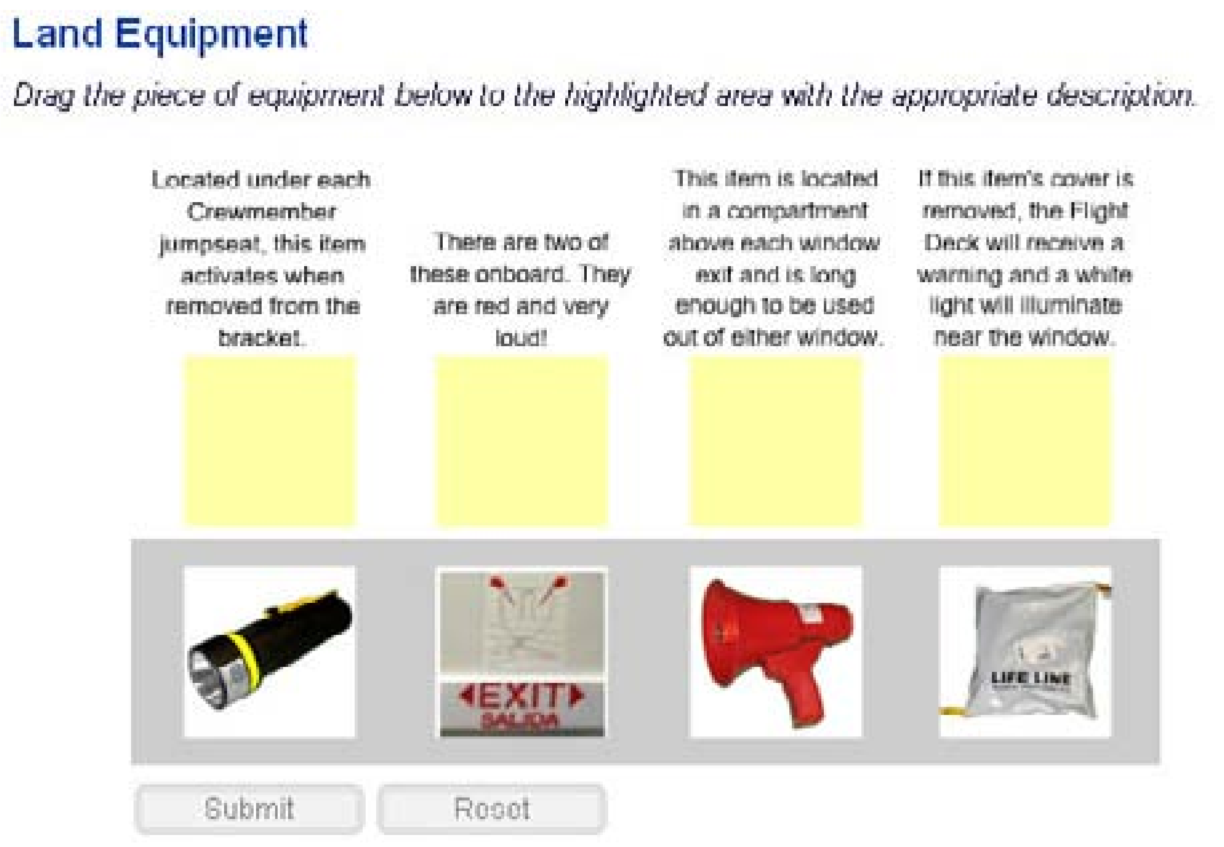
eLearning fits into the overall training strategy at JetBlue for a number of reasons. First, tracking of course completion and online exams aids in the accuracy and efficiency of training records, which are required and periodically requested by the Federal Aviation Administration (FAA). Second, as JetBlue continues to grow, eLearning will help the company scale its training to reach a rapidly expanding and geographically diverse group of crewmembers. For this low-cost, high-service airline, eLearning is a cost-effective solution that will help standardize the quality and content of training across the company. Third, the interactive potential of eLearning supports the overall training goals at JetBlue, which are not just to force the memorization of knowledge, but to prepare crewmembers to more effectively implement their knowledge in the workplace.

Flash MX, Dreamweaver, and Captivate are key tools for the development and delivery of eLearning at JetBlue. All eLearning courses are accessed via an XML/Flash-based interface that serves as a shell for course content and also tracks usage. The content of the courses themselves is multimedia, consisting of text, images, video, Flash interactions, and demos and exercises built in Captivate. Flash is an ideal tool for pulling together these various types of media as it allows learners to access course content with a single plug-in that is standard on all JetBlue computers. As an authoring tool, Flash has advantages as well. Both the skills required to use it and the content produced by it are portable, rather than being specific to a particular learning management system (LMS) or LCMS.

Flash/XML eLearning Interface

The main interface for eLearning courses at JetBlue is a Flash application that pulls data from XML files. Multiple factors contributed to the decision to use a Flash/XML-based interface. Most important, the Flash/XML interface allows interactive developers to keep the course content separate from the Flash application. Any Flash developer working in eLearning knows how often course content can change. The desire to allow instructional designers and/or subject matter experts to make changes to their own content was the driving factor behind the decision to use a Flash/XML-based solution for the interface. A carefully planned directory structure was created that keeps the content, Flash files, and ActionScript class files and their child items all separate. This allows updates to course content without affecting the Flash file itself. Since the Flash file does not need to be updated to make content changes, instructional designers and subject matter experts can make these changes in any text editor without any knowledge of Flash or involvement by the interactive developer. Another factor in our decision to use XML was the move toward SCORM standards in the eLearning community. Since SCORM uses XML, it made sense to move development in that direction. In addition, a Flash/XML-based interface was appropriate because there is a high likelihood that JetBlue will eventually use web services to have eLearning course content communicate with an LMS.

The initial look and feel of this user interface (UI) was designed with Macromedia Fireworks. Fireworks is a useful, often overlooked tool for user interface development. Multiple layouts are easily mocked up, and the design team can discuss and decide which elements work best before any Flash development begins. Fireworks enables pixel-precise


Land Equipment

Drag the piece of equipment below to the highlighted area with the appropriate description.

Located under each Crewmember jumpseat, this item activates when removed from the bracket.	There are two of these onboard. They are red and very loud!	This item is located in a compartment above each window exit and is long enough to be used out of either window.	If this item's cover is removed, the Flight Deck will receive a warning and a white light will illuminate near the window.

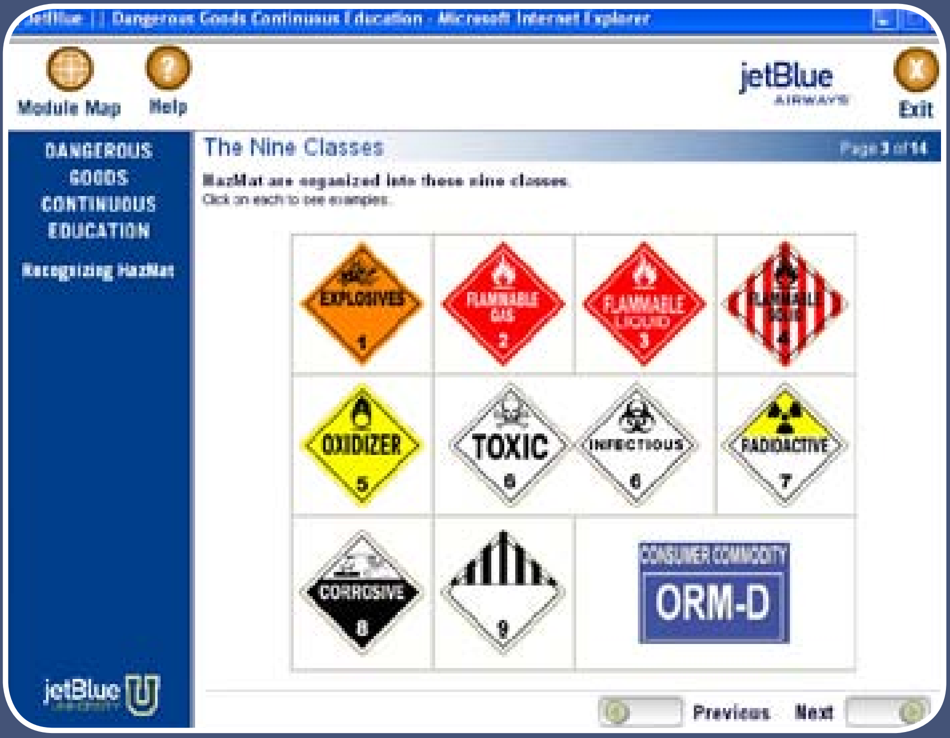
planning of all graphical elements in the interface. When drawing the vector art in Flash, the exact sizes and positions of all elements can be taken directly from the Fireworks file. Non-vector graphics such as logos can be directly exported from the initial layout. This makes Flash development much more efficient and reduces the risk of unforeseen UI issues.

The interface is made up of several separate Flash files. The main Flash file contains most of the static graphical elements and handles the loading and management of all child SWFs. Module navigation is handled by a separate Flash file. The navigation file loads an XML file with all of the module navigation information. This Flash file is comprised of two different navigation features. "First," "Next," and "Previous" buttons allow the student to move sequentially through the module one page at a time. Also included in the navigation file is a button that opens a module map. The module map allows students to navigate to any point within the module using a tree menu. Separate Flash files are also used for "Help" and "Glossary" sections, which are also XML-driven, and are loaded in by the main Flash file. The Flash files have very little, if any, ActionScript contained in the movies. Most programming is kept separate in external .as class files. This makes changes in functionality much easier by reducing the need to track code through multiple movie clips and layers.

The content for the course is currently HTML pages, which are displayed in a parent HTML page that incorporates the content and Flash interface using an inline frame. Dreamweaver MX is used to edit the HTML pages. Development is underway to do away with the content HTML pages and eventually have all content stored in XML files. Other plans include support for SCORM sequencing, enhanced assessments, and the ability to deliver performance-driven learning.

Flash as an Authoring Tool

Prior to using Flash and Dreamweaver as authoring tools, the JetBlue eLearning design team tried several proprietary authoring tools associated with particular learning management systems. Ultimately, Flash (and Dreamweaver as an intermediate solution) won out. Still in the process of testing and select-

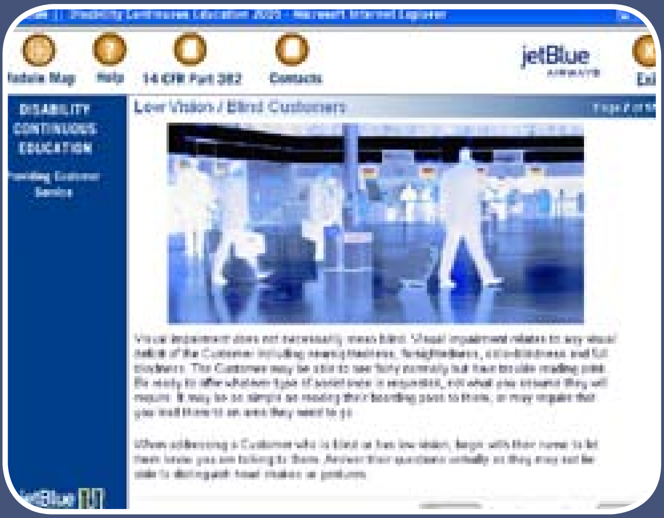

JetBlue || Dangerous Goods Continuous Education - Microsoft Internet Explorer

The Nine Classes

Page 3 of 14

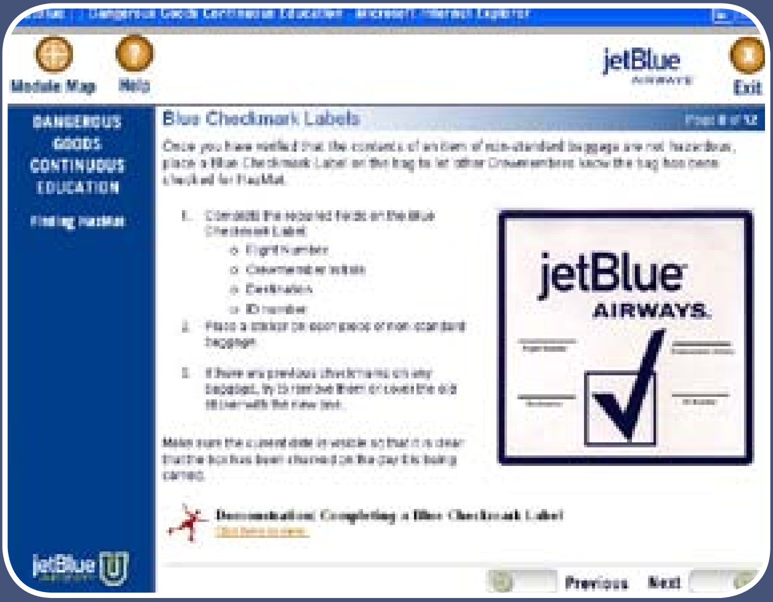
HazMat are organized into these nine classes.

Click on each to see examples.


Low Vision / Blind Customers

Visual impairment does not necessarily mean blind. Visual impairment relates to any visual deficit of the Customer including [illegible]

When addressing a Customer who is blind or has low vision, [illegible]


Dangerous Goods Continuous Education - Microsoft Internet Explorer

Blue Checkmark Labels

Once you have verified that the contents of an item of non-standard baggage are not hazardous, place a Blue Checkmark Label on the bag to let other Crewmembers know the bag has been checked for HazMat.

1. Complete the required fields on the Blue Checkmark Label
   - Flight Number
   - Crewmember Initials
   - Destination
   - ID number
2. Place a sticker on each piece of non-standard baggage.
3. If there are previous checkmarks on any baggage, try to remove them or cover the old sticker with the new one.

Make sure the current date is visible so that it is clear that the box has been checked on the day it is being carried.

Demonstration: Completing a Blue Checkmark Label

Click here to view.

ing a learning management system, JetBlue needed content that could be moved easily from one LMS to another. Furthermore, the design team found that proprietary tools tended to be imprecise in their generation of code, and to be difficult to use or have a steep learning curve. As the team expanded, the ability to hire people with skills in a particular authoring tool, such as Flash/Actionscript, was beneficial.

Although the pages of course content are still HTML-based, the integration of Flash movies has helped JetBlue to move beyond the exclusive use of static text and images to more interactive eLearning. This has enormous instructional benefit, both because online learners are infamous for ignoring most of the static text they see on a screen, and because interactivity allows us to simulate the situations in which learners may actually encounter the material. For example, we can show baggage on a belt loader and ask the learner to click on those items that would require special attention and drag them off of the belt. This simulates the real situation in which the crewmember would physically pull the bags off the belt loader.

In creating content for the eLearning courses, the design team makes frequent use of the "Quiz" template that is packaged with Flash MX Professional. JetBlue has further customized this template to create a standard look and feel for drag and drop exercises and for "hotspots" (in which the learner must click on the correct part of the screen in order to receive feedback for a correct answer). This quiz template is convenient and easily customized.

"Flash MX, Dreamweaver, and Captivate are key tools for the development and delivery of eLearning at JetBlue"

Flash Video

At JetBlue, Flash is used with video both for eLearning courses and to add transitions to longer standalone videos distributed via DVD. For eLearning courses, short video clips are often used to illustrate a procedure such as opening an aircraft door. For these we use Flash video with progressive download. The interface for our courses is built in Flash 7, and the Flash 7 player is standard on all JetBlue machines; therefore, the use of Flash video does not require an additional plug-in. This solution is limited to short video clips, as longer clips will not stay in sync – a disadvantage of Flash progressive download.

JetBlue also produces longer videos such as videos of speakers' presentations. These are currently distributed both on DVDs and via the Intranet, using Windows Media Player. The purchasing of a Flash Communication Server has been investigated; however, the cost, as well as the limited number of simultaneous users, seems prohibitive. Outsourcing the hosting of our longer Flash videos remains an option. An advantage of the Flash Communication Server for eLearning courses would be the housing of all the video on the server, rather than within the courses themselves. This would facilitate modular use (video clips could be used in multiple courses) and revisions.

Besides the costs of the Flash Communication Server, there are other challenges with using Flash for longer videos. While other formats, such as Windows Media, avi, and mov, can be played on either DVD or Web, Flash is intended for Web playback. If a video were to be distributed both online and via DVD, two versions would have to be created. An additional challenge is that Adobe Premier, which we use for video editing, does not have direct output to Flash.

In longer videos, Flash serves as a quicker, simpler substitute for Aftereffects for inserting transitions. It also enables the integration of Flash animations with the video and shows promise for advanced techniques, such as interactive video that allows the user to choose the camera angle on, for example, a piece of technical equipment. In some cases, however, it is necessary to compensate for the difference in pixel size when building Flash animations or transitions to be displayed on a TV screen.

Using Captivate for Software Training

Captivate is a promising tool for creating simulations to train crewmembers on software applications. Currently two Captivate-based training projects for JetBlue reservation agents are in the works – one for our new reservations system, and one for Blue Pumpkin, the software used by reservation agents to bid for work shifts. This training will be delivered via computer in learning labs or available to crewmembers at home. Captivate simulations will be exported as SWFs and placed into HTML pages, potentially using our existing Flash interface.

Summary

The eLearning strategy at JetBlue is still evolving. In the past year, we have upgraded from RoboDemo to Captivate, built the entire Flash/XML-based eLearning interface, and begun the transition from HTML-based authoring to XML/Flash-based authoring. We continue to strive for the best possible use of technology to meet JetBlue's learning needs and search for new technologies that will make training more effective and efficient.

Laura Sehdeva and Chip Moeser work in JetBlue University, JetBlue's corporate training department, where they use Macromedia products, specifically Captivate and Flash, for training purposes. JetBlue University recently built a Flash/XML-based eLearning interface, through which all of its eLearning courses are being delivered.
laura.sehdeva@jetblue.com

Intermedia.NET...

Now serving the hottest ColdFusion.

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

Intermedia.NET's premier hosting services include:

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

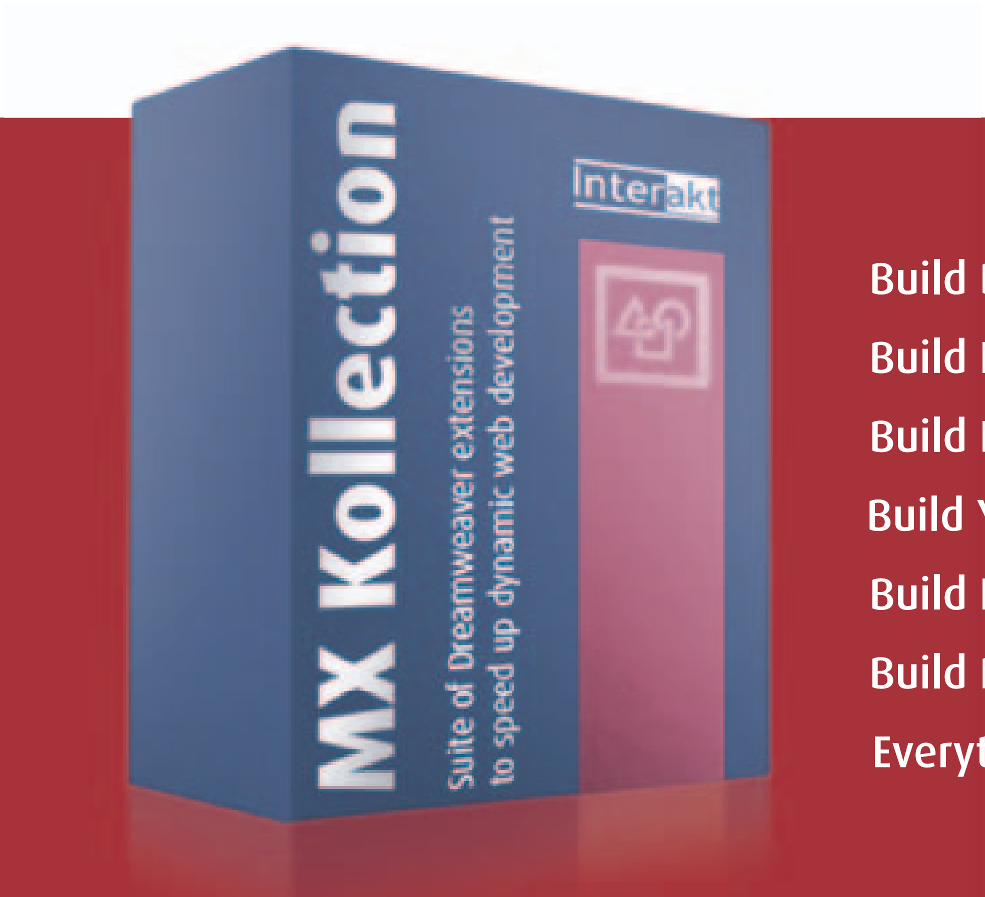
Unprecedented power, unmatched reputation... Intermedia.NET is your hosting solution.

INTERMEDIA.NET

Call us at: **1.888.379.7729**
e-mail us at: **sales@intermedia.NET**
Visit us at: **www.intermedia.NET**

GET MORE DONE

SAVE MORE TIME


MX Kollection
Suite of Dreamweaver extensions to speed up dynamic web development
Interakt

Build Dynamic Websites

Build Intranets and Reports

Build E-commerce Sites

Build Your Own CMS

Build Image Galleries

Build Registration Forms

Everything you ever dreamt of

Software that gets your job done now !

http://www.interaktonline.com/

Interakt macromedia approved macromedia® ALLIANCE PARTNER

ALL THE TIME

DREAMWEAVER PRODUCTIVITY TOOL FOR DYNAMIC WEBSITE DEVELOPMENT

Check the list bellow and see what PHP, ColdFusion and ASP tasks are taken from your shoulders.
Download the trial version from **http://www.interaktonline.com** and see for yourself how you can save more time.

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

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

Repeat Regions
Horizontal looper
Nested repeat regions
Page navigation bar
Page navigation status

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

Form Validation and Error Handling
Validate form fields
Rich formats library
Allow custom validation formats
Error handling
Preserve submitted values on error

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

Send E-mails
Send e-mail after form submit

Work with Database Information
Insert, Update Many-to-Many
Check related value into another table
Insert, Update, Delete record into table
Prevent double record insertion

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

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

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

Display Database Information
Display dynamic image
Customize image dimensions
Show/Hide info based on user login
Select UI language

Please visit http://www.interaktonline.com for more details

Between a Rock and a Soft(ware) Place

Streamlining Web-based training development with Captivate & Flash

by bryan zug

Captivate came along at a time when I was in a 'tight spot' as a web developer.

Imagine the words 'tight spot' said with all the ironic fret of George Clooney as Ulysses Everett McGill in the film 'O Brother, Where Art Thou?' and you will begin to identify with my situation.

It was April 2003. I had just been hired as a contract Web Based Training (WBT) developer by Children's Hospital Seattle. The task before me was to head up all technical development for the first robust WBT developed within the organization.

We were charged with creating an eLearning system to train all of Children's staff on a new medical orders system that was set to go live in November 2003. This new orders process was the biggest phase in a long term project of modernizing the way the hospital managed patient information.

Gone would be the clichéd days of unreadable prescriptions written in harried doctor's handwriting. Instead, all orders would be entered online into a new system where staff could interact with them collaboratively throughout the organization.

This is a case study of how Captivate has played into this WBT project at Children's and how we use this software in our ongoing eLearning products. If you are like me, how-to demos are valuable, but there's nothing like an in-the-trenches case study to help separate the deliverable wheat from the bandwagon chaff.

Now to move on to a few more details about this particular project.

Why an eLearning System? Why Custom?

As a 24/7 teaching hospital with over 3900 employees, 250 beds, and a constant influx of new staff and medical students, we had a lot of training to do.

Buying packaged eLearning software from the system vendor wasn't a viable option. Pediatric hospitals are very specialized organizations. Our medical orders system would be a very customized beast – quite different to the version our vendor implemented at regular hospitals.

Imagine the differences between a critical medication order for a two-week old baby and a 17-year old teenager and you begin to understand the complexity of a pediatric implementation of this kind of system.

Another driving force was the fact that using the medical orders system correctly was a huge patient safety gain – and simultaneously a risk. When incorrect use of a system can kill people, you tend to be very serious about certifying that everyone using it has demonstrated correct completion of the system tasks that are a part of their job.

With these requirements in mind, we began assessing what we had to build upon and what we'd need to develop ourselves.

Let's Make a WBT System!

Say 'Let's make a system!' with all the flair of a game show host saying 'Let's make a deal' and you will find yourself right alongside me during the first few weeks of the project.

Out of the gate, it was clear that there was no user tracking system in place that we could use to authenticate learners, track their progress, and report their results. In eLearning speak, these beasts are called Learning Management Systems (LMS). We put one LMS on the to-do list.

Another requirement was that learners would only see lessons that apply to the job they performed in the hospital. Doctors would see doctor lessons. Nurses would see nurse lessons, and so on. Now we were in the territory of a content management system (CMS). We added an integrated CMS to our list.

There was no user interface for the learner to navigate all of the lessons they needed to complete. We added the task of designing and programming a nice looking and easy to use web application shell to that list.

If you haven't noticed, our to-do list was getting pretty beefy. We haven't even gotten to developing the actual content yet, and the seven months between April and November 2003 is looking like an absurdly short amount of time.

Content Is Where Captivate Comes In

The last stage of assessment involved determining what process and tools we were going to use to build and deliver the content. We had to make quick, intelligent decisions on which web based eLearning delivery technology and tools to use. Would it be DHTML? Authorware? Flash? Other tools and technologies entirely?

We had 45 lessons that needed to move quickly through the storyboarding, editing, and finalization of the content phase and then on through technical production into deployment.

This is when I first came across Captivate (then known as RoboDemo) and it looked pretty good. It seemed a bit more robust than some comparative tools in the same price range (like ViewletBuilder or Camtasia) and was pretty user friendly. We definitely needed a tool that was easy for our non-technical Subject Matter Experts (SMEs) to use. They had to have something where they could rapidly capture screenshots and prototype lessons.

Our SMEs in particular were nurses who had never done any kind of multimedia, software, or WBT development. Getting an easy to use tool where they could storyboard their lessons out quickly and easily was going to be paramount to the success of the project. Whichever reasonably priced tool could offer more production advantages beyond these basics would be a serious contender.

This is when I got wind of a coming enhancement that nearly made our decision for us.

.FLAs Directly from Nurses' Storyboards, Developers Weep for Joy

In June 2003, a new feature was released that allowed the exporting of Captivate files as fully editable Flash authoring FLAs.

There haven't been many times in my professional career that I can remember my exact reaction upon hearing 'such and such' news, but I burst into tears of joy when I read that announcement.

Yes, real developers do cry.

Here I was in my 'tight spot' trying to balance all the competing tasks of the project. I was looking for as much leverage as possible because we would not make our deadline without it. So much was not yet designed and was still unknown that we absolutely needed the most robust, scalable, and flexible technology to build our system.

Flash was robust enough to handle all the requirements of the client side of the application, but, as anyone who's built a robust Flash app will tell you, it takes a lot of work. You are not a junior college 'design hipster' cranking out 'bleeding edge' splash screens. You are building software – a Rich Internet Application (as many folks refer to them), and it is hard work.

We tested the export to FLA feature the day it was released and it did exactly what we needed – exporting all of the assets of the Captivate movies in an organized FLA.

This gave us the robust path we needed to rapidly prototype and build the content for our WBT – one that could quickly adapt to emerging system needs. I will touch on some real-world, saved-my-BLEEP, examples of this in a bit.

One important thing to note that we discovered in how this feature works is that exporting an .FLA requires that the Flash authoring environment be installed on the same machine as Captivate.

In that, it is not so much an export .FLA feature as it was an import Captivate to Flash MX (and later Flash MX 2004) feature. This is an important distinction that has been confusing to many since its debut. Misunderstanding what that means is a $500 difference if you need the supplemental software the function requires.

Our Production Process

So we adopted Captivate as our content capturing and design tool of choice, with Flash as our delivery technology for the client shell and the lessons. The production process we setup for the initial 2003 project looks very much the same today as it did then.

It would be really romantic to tell you that we use all of Captivate's great features (like AICC/SCORM compliant output or easily crafted audio narration), but we don't. In a nutshell we take the raw Captivate .FLAs and hand copy the backgrounds and captions from them and into the custom .FLA lesson template. This file houses all of the code required to track and manage everything in conjunction with our homegrown LMS.

Not very sexy, I know, but it makes good business sense. We are now at the point where we've been able to automate nearly all of our technical FLA production via the creation of custom captions in Captivate and the use of Flash MX 2004 JSFL scripted commands.

In all, it takes us less than 1.25 hours of technical production to create and deploy a 20 interaction lesson – and it's getting shorter all the time. This is for a template that includes the interaction and a demonstration of the task if the learner makes two incorrect attempts.

Happy We Chose Captivate

In all likelihood, we would not have met our initial 2003 deliverables without Captivate. We needed something that was inexpensive, easy to use for

"When incorrect use of a system can kill people, you tend to be very serious about certifying that everyone using it has demonstrated correct completion of the system tasks that are a part of their job"

"We adopted Captivate as our content capturing and design tool of choice, with Flash as our delivery technology for the client shell and the lessons"

non-technical folks, and robust enough to leverage into the land of enterprise web application development. Captivate hit this sweet spot for us and continues to do so today. We've been through three versions of the software (RoboDemo 4.0 and 5.0, Captivate 1.0) and the current implementation is so much more robust than when we began using it.

Captivate 1.0's productivity enhancements continue to reduce turnaround times for our content. Notable timesavers for us have been: 1) Snap capture window to an open application, 2) recording defaults for caption styles that automatically set all captions to our design style guide during the capture process, and 3) visual timeline editing for each slide.

Captivate also continues to lead functionality wise for its price point. Cost of the next level tools is around $10,000. Knowledge Planet's Firefly (http://www.knowledgeplanet.com) or Epiance's Epiplex (http://www.epiance.com) deliver more functionality, but they do so at a steep price difference.

Captivate's Leverage Toward Flash FLAs Has Paid Off

Captivate's foot in the door toward Flash FLA development has also delivered big payoffs. Here are three of the real-world implications of Flash FLA development that have saved us from disaster:

Wider adoption means more developers...

Given the growing weight of the developer community behind Flash, it is easier to find developers for it than for many of the other less popular eLearning tools in circulation. In July 2003, when we needed an extra Flash developer to come in who was very code-centric, we had one within a few days.

Wider adoption also means more supplemental software...

In August 2004, when Microsoft quietly killed Internet Explorer's full screen chromeless window display mode with their Windows XP Service Pack 2 and effectively broke our WBT, we had third party vendors like Multidmedia (http://www.multidmedia.com) we could turn to for help.

They provided alternate fullscreen standalone projectors that would support seamless right click interactions in Flash – something that we had been mitigating with Internet Explorer dependent DHTML hacks since launching our initial WBT in 2003.

A mature IDE means more leverage...

The automation of the Flash MX 2004 IDE with JSFL is a dream come true for those of us faced with repetitive coding tasks that require just enough customization that they cannot be fully automated very easily. It's very satisfying to go to my command menu in Flash, click my "Text Entry" demo command, and watch it do 97% of the work for me.

The Last Word

Bottom-line – Flash as a technology and delivery platform is robust enough to quickly develop for and adapt to changing enterprise needs. When folks from elsewhere in the enterprise tell you they are about to make changes that will break your application, it's nice to have a robust enough technology that you can adapt in a reasonably short amount of time.

Having a tool like Captivate that strongly leverages content toward Flash FLA development for our WBTs enabled us to deliver the requested functionality. My clients are always much happier when my tools and technologies allow me to say "yes" when they ask "Can we do that?"

Bryan Zug joined Children's Hospital of Seattle in April 2003 to develop their first enterprise-wide web based training (WBT) products. The resulting web application is a combination of Flash, ASP, and SQL-Server technologies that serve as the key elements for training the hospital's 2500+ clinical staff on a new medication ordering system introduced in November 2003. Working as a web and multimedia developer since 1996, Bryan's projects have included WBT, e-commerce, content management systems, and user interface design for clients that include Bermuda Fire & Marine Insurance, Volkswagen/Audi of Latin America, Premera Blue Cross, and Perkins Coie law firm, among others. He has also performed quality assurance systems analysis for Real Networks and associated client partners such as CNN, ABCnews, CBS, FoxSports, Major League Baseball, NASCAR, and E!.
bryan.zug@seattlechildrens.org

Fireworks-Animated GIF with FreeHand Vector Art

It's fantastic, fabulous, and fun

by joanne watkins

What do "fantastic," "fabulous," and "fun" have in common? All three of these words describe the creative work you can do with Flash, Fireworks, and FreeHand. The Macromedia MX2004 Studio "Fab Three" together give you the tools to create art and animations for the web and other media. Combine these with Dreamweaver and you have a dynamic set of tools.

Did you know that all three – Flash, Fireworks, and FreeHand – can create animations? Of course, Flash is the most well known and the most powerful tool for web animations. Fireworks and FreeHand can create the graphics for simple animations to use in Flash and export in the SWF format. Fireworks is quite unique because it can create graphics, open and import vector art from FreeHand, create animated GIFs, and open animated GIFs created in other applications for enhancing.

Why would you want to create an animated GIF when it is an old technology dating back to the late 1980s? Answer: because an animated GIF might just be the perfect solution to fit a particular client's needs.

What is an animated GIF and what are the advantages or disadvantages? An animated GIF contains multiple images encoded in one single file. The advantages are: browser support without any additional plug-in or players, transparency, and a good choice for very simple animations. Some of the disadvantages are: 256 colors, a large file size, and that an animated GIF is limited to simple animations.

Why would you choose Fireworks to create your animated GIF? Fireworks' ability to create artwork, superior GIF optimization, and the use of layers and frames allow you to make a simple animation with a minimum number of steps. Add, to this, the compatibility with the vector illustration power of FreeHand and you have a winning combination.

What are the general steps to create an animated GIF with Fireworks and FreeHand?

- Create the artwork in FreeHand, Fireworks, or a combination of both.
- Use the features in both products that distribute objects to layers and frames.
- Set the frame rate and how many times the animation will play.
- Optimize to reduce file size.
- Export as an animated GIF.

Animated GIF Example

What is a specific example of an animated GIF and what are a few of the important steps to make one? The example for this article is a simple animation to promote a fictitious Flower and Garden show on a web site.

The animation is on the first page of the promotion and on each subsequent web page there would be a still graphic with one of the floral images. The animation will be a simple stylized flower with the petals dropping off one by one to reveal theme-related pictures with a hand-painted look (see Figures 1 and 2). The purpose of the example presented in this article is to serve as a starting point and to help generate ideas that you can use to create one of your own GIF animations. It is only one method to create the animation. There are many different ways of achieving the same end result.

Artwork in FreeHand

FreeHand is used in the example to develop the artwork to take advantage of the power duplicating feature to create the flower, to use the color tracing feature to create the hand-painted look for the photographs, to attach text to a spiral path, to use the release to layers to separate each object on a layer, and the compatibility with Fireworks. Note: You can also use Fireworks to create the artwork and use the distribute layers to frames feature.

1. In the FreeHand document set the unit of measurement to pixels and set the document size to custom. Type in the pixel size width and height for your animation. For example 200 x 200.
2. Create one petal shape using the Pen tool or modify a circle. Add a small circle to use as the center of rotation for the

figure 1

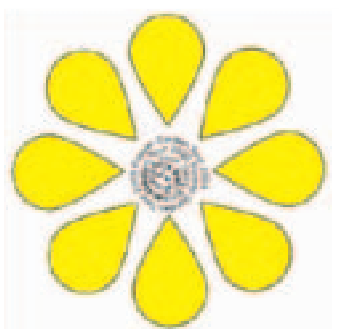


figure 2

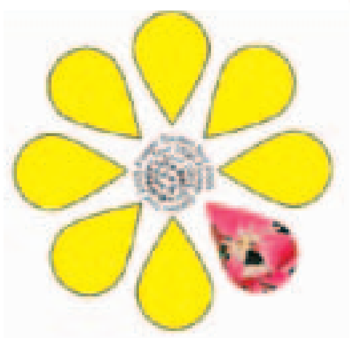
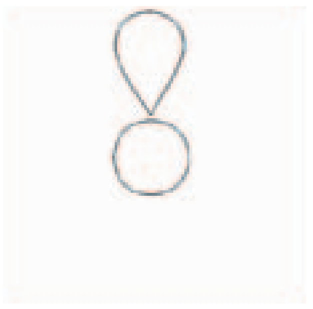


figure 3



flower petals.

3. Rotate the petal shape as necessary and place in position close to the circle. For this example, you will want to place the first petal where you want the animation to begin. Place the petal at the top for the animation to start at the top and go in a clockwise direction (see Figure 3).
4. Choose Edit > Clone to make an exact copy of the petal on top of the existing one.
5. Be sure the clone is selected. Select the Rotate tool from the Toolbox. Click the center of the circle to establish the center of rotation. Press Alt + Shift and drag the mouse to make a rotated copy constrained to and an angle of 45 degrees (see Figure 4).
6. Choose Edit > Duplicate or use the keyboard shortcut. Repeat the duplicate command as many times as necessary to complete the petals around the circle.
7. Delete the center circle used for the center of rotation. Choose Window > Toolbar > Xtra Tools and select the Spiral Tool. Make a spiral in the center of the petals.
8. Using the Type tool, type a block of text and attach it to the path.
9. View the completed steps (see Figure 5).
10. Save two copies of the file using Save As and a different file name.

One of the files is used for the animation so that the flower petals one by one disappear. The other file is used for the image you see when the flower petal drops off. You will want these two files to be an exact duplicate as the bottom one with the images will be opened in Fireworks and shared across layers. The top one will be imported into the same Fireworks file converting the layers to frames and will be positioned exactly on top of the other one for the animation.

figure 4

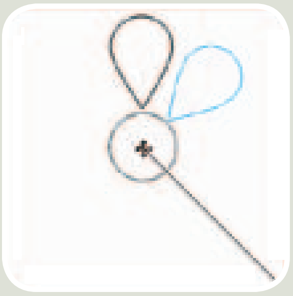


figure 5

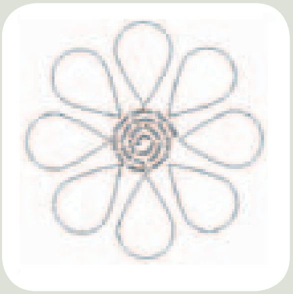


figure 6

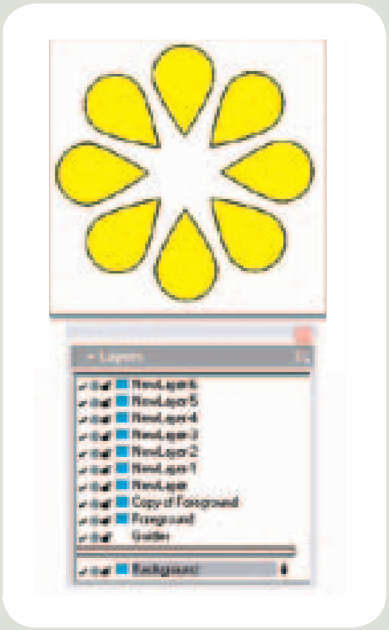
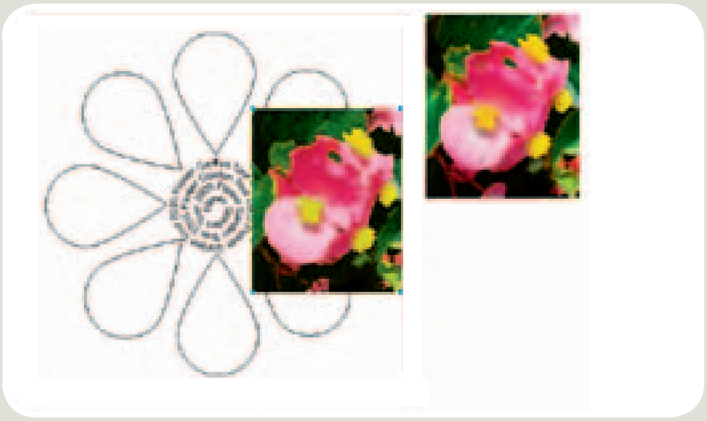


figure 7



Prepare the FreeHand Art for Animation

1. In one of the two saved FreeHand files, delete the center as it will not be animated.
2. Select all the petals and use the Fill Bucket on the Toolbox to fill them with a Hexadecimal color of your choice. The example uses FFFF00.
3. Select all the petals and group them. It is important to group the objects in order to release them to layers.
4. Deselect the group as you will need to add a duplicate layer.

In this example, the petals will all be visible before the animation starts. If you do not make an extra duplicate layer, FreeHand will drop the first object on the first layer. This would show your GIF with one petal missing before the animation starts.

5. In the Layers Panel select Duplicate.
6. Be sure you are on the duplicate layer and select the grouped petals.
7. Choose Xtras > Animate > Release to Layers. Select Drop from the pop-up menuand check Use Existing Layers.

The Drop selection for the example animation copies the objects to one layer except for one petal and continues to add as many layers as needed. In the example there will be eight layers one for each of the dropped petals and the duplicate with all the petals for a total of nine layers excluding the Guides and Background layers(see Figure 6).

Although you are going to Fireworks to make this an animated Gif, you can test the animation using the SWF test movie feature in FreeHand (Window > Movie > Test) to see if the animation is working as expected. If not, correct any steps before going to Fireworks.

Prepare the FreeHand Art for the Base

1. Open the other saved FreeHand file. This is the one with the text spiraled in the center.
2. Import each prepared photograph that is already sized to be slightly larger than the flower petal.
3. Place the photograph on an area of the pasteboard (blank area outside the

borders of the document). Previously to import the photograph at its original size.

4. Double-click the Trace tool and use settings that will be appropriate for the final GIF art from Fireworks. For example 256

figure 8

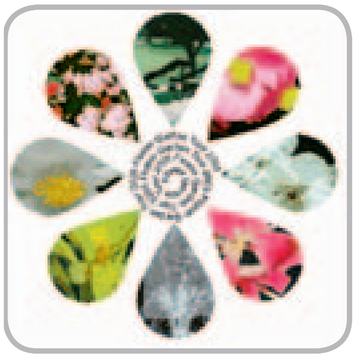
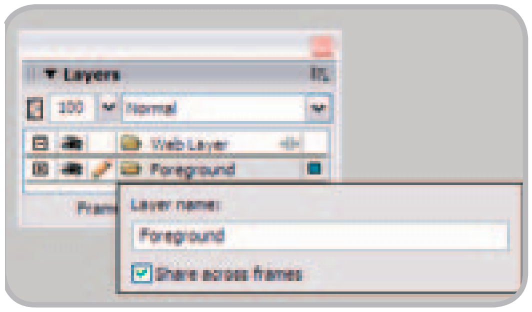


figure 9



Layer name:
Foreground
[✓] Share across frames

colors, RGB, High resolution, and Tight trace conformity.

5. Using the Trace tool, draw a rectangular marquee around the photograph and without deselecting, immediately group the traced image. This is very important as the new image contains numerous separate vector paths.
6. To decrease the number of vector paths, choose Modify > Alter Path > Simplify and type a number or use the slider. The number 2 is used for the example.
7. Move the traced art and place on top of one of the petals. Choose Modify > Arrange > Bring to Front to make sure the photo art is at the top of the stacking order (see Figure 7).
8. Cut the image.
9. Select the flower petal and choose Edit > Paste contents.

This creates a clipping path. To reposition the picture inside the petal, select the petal and in the Object properties select Contents located below Clipping path. Use the pointer tool and move the paste contents handle to reposition the picture.

10. Repeat for all the flower petals and delete the original photographs from the pasteboard area (see Figure 8).

Fireworks

Open the FreeHand File

1. Launch Fireworks and open the file with the floral images (see Figure 8). In the sample figure the canvas is set to white for illustration purposes only.
2. Leave all the default settings in the Vector File Options. Be sure that Remember Layers is selected in the second pop-up menu under File conversion.
3. In the Layers, panel collapse the Foreground layer which contains the objects from FreeHand.
4. Double-click the layer name and check Share across frames.

This will allow the flower with the art images to be on every frame and visible when a petal is dropped (see Figure 9).

Import the Second FreeHand File

1. Choose File > Import and locate the FreeHand file with the animation.
2. In the Vector File Options window, select convert layers to frames in the second pop-up menu under File conversions. Click the Fireworks canvas in the upper left corner.
3. Open the Frames panel. Set the number of loop times from the Looping pop-up menu at the bottom of the Frames panel.

If you only want the animation to play once and you want it to end showing all the yellow petals, duplicate frame one and select at the end position.

4. Double-click the column containing the frame rates and set the desired rates (see Figure 10).

You can Shift select the Frames to apply a new rate to all the selected frames at once. You can also set the looping and frame rates in the export preview window.

5. Open the Optimize panel. It is extremely important to select the Animated GIF option in this panel even though you might not change any optimization settings. Setting the format here gives you the multimage GIF in one file when you export (see Figure 11).
6. Click the 2 up or 4 up tab in the Fireworks window to preview your different optimization settings to create a smaller file size.

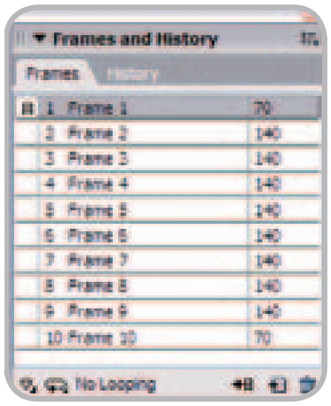
For further information on optimization, please refer to a previous article by Joyce Evans, "Fireworks Image Optimization Basics" (MXDJ vol. 2 issue 10).

7. Choose File > Export and you can either export only the animated GIF or you can export the image and the HTML page from Fireworks.
8. Test the animation in the browser.

The completed animation is now ready and you have another solution to fit a particular client's needs. It is fantastic, fabulous, and fun to turn your creative ideas into animations with the "Fab Three": Fireworks, FreeHand, and Flash.

Joanne Watkins is a featured an instructor for Hewlett-Packard's Online Learning Center. She is also an Associate Professor in the Applied Graphic Design Technology department of Collin Community College and teaches in the Senior Adult Education Program of the Business Studies Division at Brookhaven College in Dallas, Texas. Previously at Macromedia she served as a DevNet technical editor, web producer, and technical support engineer. Joanne was also the technical editor for nine Macromedia Press Books about Fireworks and FreeHand.
joannew@imagetechinfo.com

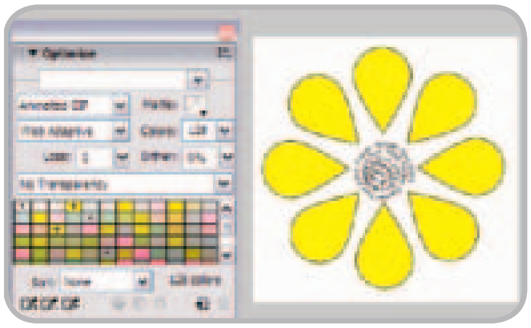
figure 10



1	Frame 1	70
2	Frame 2	140
3	Frame 3	140
4	Frame 4	140
5	Frame 5	140
6	Frame 6	140
7	Frame 7	140
8	Frame 8	140
9	Frame 9	140
10	Frame 10	70

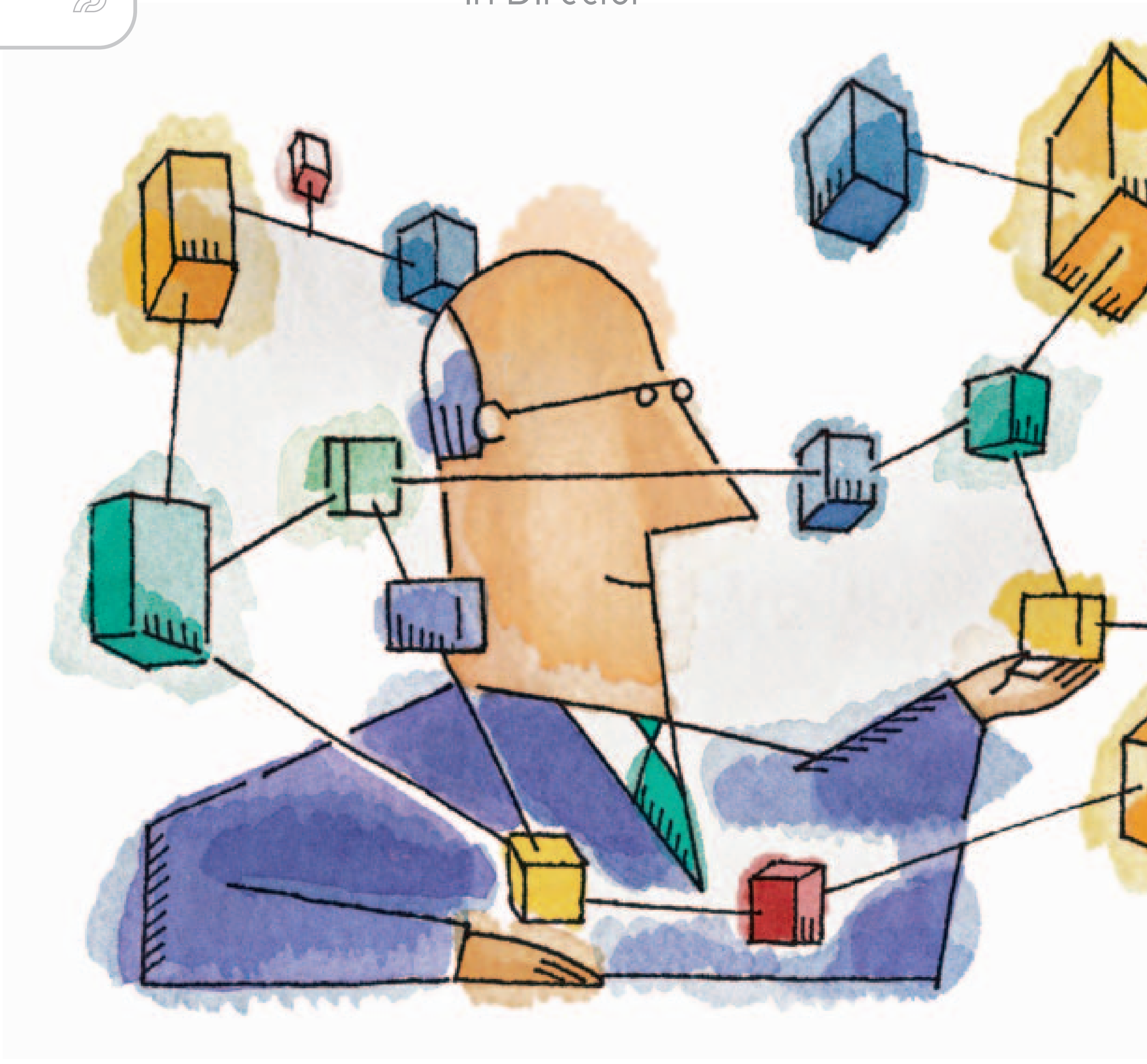
No Looping

figure 11

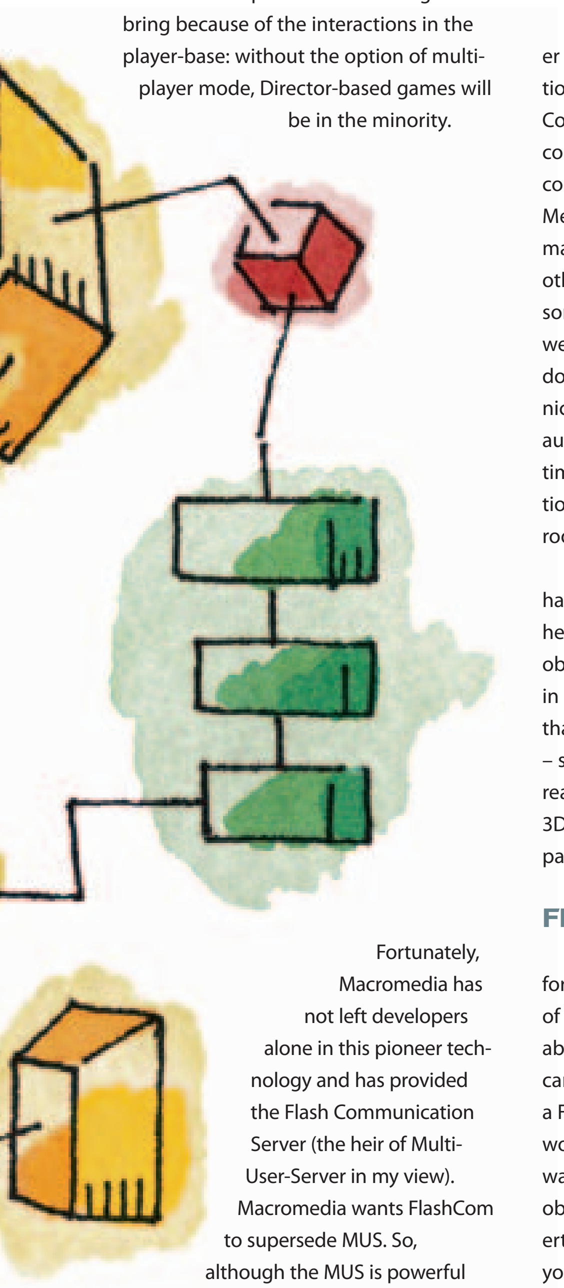


A Model for Dealing with the Flash Communication Server in Director

by nima azimi



We live in a communications world. The number of software solutions that can provide communication between users grows every day. In the future it's likely that applications unable to provide this ability will be known as "traditional applications," shunned because of their lack of openness and communication ability. A simple example of this can be seen in the game industry. Gamers are addicted to multiplayer games with others and to the pleasure that such games bring because of the interactions in the player-base: without the option of multi-player mode, Director-based games will be in the minority.



Fortunately, Macromedia has not left developers alone in this pioneer technology and has provided the Flash Communication Server (the heir of Multi-User-Server in my view). Macromedia wants FlashCom to supersede MUS. So, although the MUS is powerful enough to create amazing applications with Director such as we see on the web today, following the MX version of Director, Macromedia suggests using FlashCom Server instead. FlashCom is known as a revolution in Macromedia history and this is mainly because of its integration across several products. FlashCom is so "open" that a running joke on a few of the forums I see has Macromedia changing its name to the "Macromedia Communication Server." What follows is an example of combining Director and the FlashCom Server to provide a backbone for multi-user applications, in a more modern way than traditional MUS applications.

What Is FlashCom?

Flash Communication Server is a server technology that provides the connections between Flash 6+ players. The Flash Communication Server is a hub. Users connect to the hub using Macromedia's communication protocol called Real-Time Messaging Protocol (RTMP). FlashCom makes it possible to communicate with other users through Flash files (and, with some conversions, in Director files) in web pages or standalone applications. It does this with several types of communication streams, such as sending video, audio, and text between users in real time. End users experience these applications as video-audio conferences, chat rooms, and so on.

In addition to these types, FlashCom has a "remote shared object" that is the heart of its flexibility. It lets us make an object like a 2D sprite or 3D object exist in a way that is shared between users so that they can manipulate its properties – such as position, rotation, and so on in real time together. You can think of it as 3D ball in a football game that users can pass it to each other.

Flash Object in Director

Director has supported Flash objects for a number of years, with each ve rsion of Director supporting the latest available version of Flash. With this ability we can work with FlashCom in Director as a Flash object. We can manipulate and work with the Flash object in several ways. We can get or set all variables and objects in Flash files and access all properties of those variables and objects. If you have used FlashCom in Flash before, you're familiar with the "Developing Communication Applications" document provided by Macromedia (http://www.macromedia.com/support/flashcom/documentation.html). In that document there are six sample projects that illustrate the basic capabilities of FlashCom very smoothly. I think the best solution is to learn how to use FlashCom in Director would be to re-code those samples in Director one by one. Thanks to John Taylor and Jay Armstrong for their articles in Macromedia Developer Center, we have the torch to light the dark road of starting this kind of implementation.

But there are several situations in which the best way to implement these types of applications in Director is not immediately apparent. By starting with simpler, more contained, projects and gradually growing into more advanced applications, one can learn FlashCom more efficiently.

For this article, I have chosen two samples from the initial FlashCom documents that I think have good features and could be as a guideline for other projects, one on sharing text between users, and one on sharing a sprite object. FlashCom applications generally have a client side-script written in a .fla file (which we will either replace or encapsulate inside the .dir) and may have server-side script written in an external file in the ActionScript language with the same name as application name, or "main" with a ".acs" extension. In this article I focus on samples that don't need any server-side scripting, but if you see these files in other examples, that is likely their intended use.

Sample One - (Shared Text)

The first sample I present is a "Shared Text" project. The goal of this project is to have a textbox shared between any users that visit our Shockwave page. They could write any message to any other user. While a user is typing, other users can see the characters typed one by one.

The first step is to create the Flash object. We can do this dynamically with Lingo or by importing a Flash file into the cast window. I use this later in the sample. Since you don't want to use any built-in Flash embedded video or pre-provided FlashCom components in your project, you don't need to do anything in your Flash movie. You need only to create an empty 10 by 10 pixels Flash file and publish it to ".swf". When you import this .swf file to Director and drag it to the stage, you have all you need to work with FlashCom. Keep in mind that Shockwave file must contain the swf in its stage boundaries. You can drag it to the left down corner of the stage and set the ink property to "Background Transparent." When you publish your project, save your ".dcr" and ".htm" files to a directory named "myProjectDirectory" (or anything else you want) in your FlashCom applications directory (for example C:\inetpub\www-root\flashcom\applications\).

TIP: try to write all your FlashCom scripts and functions in one "behavior script" and assign it to your flash sprite; then call those functions from other sprite or frame "behavior scripts" if needed.

Step #1:

To provide any FlashCom capability to your applications you need to establish a connection to "Flash Communication Server". Do this by creating a new NetConnection object.

If we were programming in Flash we would use the line:

```
client_nc = new NetConnection();
```

In Director, we first assign a variable to our flash sprite and then use this variable to create flash objects and manipulate them, like this:

```
pSprie = sprite("myFlashSprieName")
pNetConn = pSprite.newObject
   ("NetConnection")
```

Step #2:

Any NetConnection object has an "onStatus" event raised when a connection is established. We use this event to test our connection status. It has a parameter defined according to the status message. The message is stored in the code property of this parameter. In some situations we can use it as a guide to alert the user if the connection was a success, closed, failed, rejected, etc. In Flash, this is implemented through a "function literal" or "anonymous function" method but we can't do it in Director.

In Flash:

```
client_nc.onStatus = function(info) {
 trace ("Level: " + info.level +
    "Code: " + info.code);
 if (info.code = "NetConnection.
    Connect.Success") {
  // Continue
 }else if (info.code = "NetConnection.
    Connect.Closed") {
 {
  // Alert Message
 }
};
```

We must implement this code in Director with the help of the setCallBack command. setCallBack is a Flash command that can be used as a sprite or a global method to define a Lingo callback handler for a particular event generated by the specified object. When ActionScript triggers the event in the object, that event is redirected to the given Lingo handler, including all arguments that are passed with the event.

In Director:

```
pSprite.setCallBack(pNetConn,
   "onStatus", #myOnStatus, me)
on myOnStatus (me, this, aInfo)
 put "Level: " && aInfo.level &&
   "Code:" && aInfo.code
 if (aInfo.code = "NetConnection.
   Connect.Success") then
  -- Continue
 else
  -- Alert Message
 end if
end
```

The parameters of setCallBack are an Actionscript object (NetConnection), the Actionscript event that occures (onStatus), the Lingo handler that handles the Actionscript event (myOnStatus) and the Lingo object that contains the corresponding handler (me). Thus any time NetConnection tries to connect, onStatus handler is triggered and myOnStatus handler triggers with all onStatus arguments in Director. Keep in mind that third argument of myOnStatus handler (aInfo) is according to the "info" parameter.

Step #3:

Now, we must connect to server. This part is very similar in Flash and Director. We must pass a RTMP url and an instance name to the Connect method. In this example room01 is an instance name for our connection. We use instance names in applications that hold several simultaneous connections (a multi-room chat for example).

In Flash:

```
client_nc.Connect("rtmp://localhost/
   myProjectDirectory/room01");
```

In Director:

```
pNetConn.Connect("rtmp://localhost/
   myProjectDirectory/room01")
```

Step #4:

Now it's getting fun! To create the remote shared object for our text sharing we use the "getRemote" method of SharedObject. The first parameter of the getRemote method is an object name. client_nc.uri is the URI of the NetConnection. The shared object will use it to connect to the server. The third parameter is a Boolean that indicates whether our shared object is persistent on the server or not. If we set it to true our remote shared object saves its value, thus other users that connect later receive the last value of the shared object from the server. Its value remains even when all users disconnect from server. In this sample we don't want our text to be persistent to the server. Next to this line of code we connect our created shared object to NetConnection that was constructed earlier.

In Flash:

```
text_so = SharedObject.
getRemote("sharedText", client_nc.
   uri , false);
text_so.connect(client_nc);
```

To create a shared object in director we must first get the SharedObject binding from Flash into a variable and then use it. The getVariable function helps us to do so. We use false for second parameter as "returnValueOrReference", because we want to get an object of Flash not its value.

In Director:

```
 mySharedObject = pSprite.getVariable
   ("SharedObject", false)
 text_so = mySharedObject.
getRemote("sharedText", pNetConn.
   uri. false);
 text_so.connect(pNetConn)
```

Step #5:

When something changes in the shared object, the server sends a synchronization message and an onSync handler change the information on client movie. This way when a user changes a property assigned to a remote shared object, with the help of this onSync handler, all connected users could see those changes.

The “list” parameter is an array that contains information about changes in our textbox. When a user changes a remote shared object by typing a character in textbox with instance name “TypingStage”, the server sends a message and onSync handler trigger on all client’s machines. “list” array has two properties, name and code. If any change occurs, the code property is set to “change” and when the changes assign to our textbox this property is set to “success”. Any change adds an entry in the list array. Thus, when onSync handler triggers, we must test the “list” array to find out any entry with “change” value and if so, update our textbox.

This handler is implemented with “function literal” method again, in Flash.

In Flash:

```
text_so.onSync = function(list){
 for (var i=0; i<list.length; i++){
  if (list[1].name == “textValue”
    && list[i].code != “success”) {
   TypingStage.text = text_so.data.
      textValue;
   Break;
  }
 }
};
```

Again, we use setCallBack to implement this code.

In Director:

```
pSprite.setCallBack(text_so,
   “onSync”, #mySyncTexts, me)
on mySyncTexts (me, this, aList)
 repeat with i=0 to (i<aList.length)
  if (aList[i].name = “textValue” AND
     aList[i].code <> “success”) then
   member(“dirTypingStage”).text =
       text_so.data.textValue
   exit repeat
  end if
 end repeat
end
```

Step #6:

What happens when you type a character in the “dirTypingStage” textbox? You change a textbox property and this change must be seen by other users, therefore, remote shared objects should update. This is very straightforward in Flash by using the “onChanged” event of the Flash textbox object, but in Director the story is a bit different.

In Flash:

```
TypingStage.onChanged = function(){
 text_so.data.textValue = TypingStage.
   text;
};
```

In Director, we have two methods to implement this equivalent of the block of code above. The first method that I prefer for this sample is to use “the keyUpScript” in the “on StartMovie” handler. We tell Director that whenever a keyUp event has occurred, trigger the “textChanged” handler that is assigned to “the keyUpScript”. Remember that the “keyUpScript” and “textChanged” handler must defined in a movie script. The second useful method when we have several textboxes or fields, is to assign an “on keyUp” handler to the textbox itself as a behavior script. In this way when a user types in textboxes other than the “dirTypingStage” textbox, there is not any assignment to remote shared. But in the former method any typing from user in any textbox in the stage causes the call of “onSync” handler from the server.

In Director:

```
on StartMovie
 the keyUpScript = “textChanged”
end
on textChanged
 text_so.data.textValue =
   member(“dirTypingStage”).text
end
```

The complete listings of this sample are shown in Listings 1 & 2. After publishing your completed project you can test it and see the result. Run your “.htm” file twice and while typing in the textbox at one of them, see the other textbox. If you built a custom textbox that does something like a text editor for a foreign language character sets, there is no need to build same custom textbox from the beginning in Flash to work with FlashCom. The solution is to use FlashCom inside Director!

Sample Two - (Shared Ball)

In this sample we want to create a shared ball (2D circle sprite) so that any users visiting our Shockwave page can change the position of the ball with their mouse, while other users see that ball moving on the stage. This is a technique that is used at a higher level in several multi-player games.

Steps 1 and 2 are the same as previous sample, step 3 and 4 are very similar, but with small exceptions. In step 3 we use a new path as RTMP url and in step 4 we use “ball_so” as the name of our shared object and use “position” as the first parameter of getRemote method.

Step #5:

We must now write an onSync handler. When a user changes the position of the ball, this changes what is stored in the x and y properties of our sharedObject. Data must be assigned to the ball sprite location properties (or ball movieClip in Flash), on the other client’s movie. When this is done, all users see the repositioning of the ball at the same time.

In Flash:

```
ball_so.onSync = function(list) {
 sharedBall_mc._x = ball_so.data.x;
 sahredBall_mc._y = ball_so.data.y;
end
```

In Director:

```
pSprite.serCallBack(ball_so,
   “onSync”, #mySyncBall, me)
on mySyncBall me, this, aList
 sprite(“Ball”).LocH = ball_so.data.x
 sprite(“Ball”).LocV = ball_so.data.y
end
```

Step #6:

When a user changes the ball position with their mouse, this change must be assigned to x and y properties of “ball_so.data”. When those properties are changed, the server sends a synchronization message and our “onSync” handler triggers the other users’ machines.
This code in Flash is implemented in the “onPress” handler of the ball movieClip but for the best performance in Director, we implement it on “exitframe” handler of a frame behavior, using traditional Director-style event handling.

In addition to setting the x and y properties of shared object, we must ensure that the user does not move ball out of the stage boundaries. Therefore, if our stage

Advertising Index

Advertiser	URL	Phone	Page
CFDynamics	www.cfdynamics.com	866-233-9626	3
CFUNITED	www.cfunited.com	301-424-3903	19
ColdFusion Developer's Journal	www.sys-con.com/cfdj/subscription.cfm	888-303-5282	35
EV1 Servers	www.ev1servers.net	800-504-SURF	6
FlashForward 2005	www.flashforwardconference.com	877-4FLASH4	15
HostMySite.com	www.hostmysite.com/mxdj	877-248-4678	58
InterAKT	www.interaktonline.com		26 & 27
Intermedia.net	www.intermedia.net	888-379-7729	25
IT Solutions Guide	www.sys-con.com	201-802-3021	49
Macromedia	www.macromedia.com/go/video5	415-252-2000	2
Macromedia WPS	www.macromedia.com/go/webupdate	415-252-2000	60
MX Developer's Journal	www.sys-con.com/mx/subscription.cfm	888-303-5282	31
Nidus Corp.	www.brainstormer.org	888-894-3840	9
PaperThin	www.paperthin.com	800-940-3087	11
Seapine Software	www.seapine.com/webdev	888-683-6456	5
SYS-CON Newsletters	www.sys-con.com	888-303-5282	57
SYS-CON Reprints	www.sys-con.com	201-802-3026	13

size is 640 by 480, we test the position of the ball and if it is 50 pixels (depending on the size of the ball) out of the stage boundaries, we move it back into the stage.

In Flash:

```
sharedBall_mc.onPress =
   function(){
 this.onMouseMove =
   function(){
  ball_so.data.x = this._x =
   _root._xmouse;
  ball_so.data.y = this._y =
   _root._ymouse;
  if (sharedBall_mc._x >=
     stage.width){
   sharedBall_mc._x = stage.
      width – 50;
  }
 if (sharedBall_mc._x <= 0){
  sharedBall_mc._x = 50;
 }
 if (sharedBall_mc._y >=
   stage.height){
  sharedBall_mc._y = stage.
     height – 50;
 }
 if (sharedBall_mc._y <=0){
  sharedBall_mc._y = 50;
 }
};
sharedBall_mc.onRelease = sharedBall_
   mc.onReleaseOutSide = function(){
 delete this.onMouseMove;
};
```

In Director we create a global variable named "isMouseDown". Assign a behavior to the ball sprite and use this variable to determine if user clicks on the ball sprite (isMouseDown = true) or not (isMouse-Down = False).

We use the "rect" property of the ball sprite to test its location and the stage boundaries. "rect" is an array that has 4 elements: the left, top, right and bottom coordinates of the sprite.

In Director:

```
on exitFrame
 if sprite(“Ball”).rect[1] <= 0 then
  sprite(“Ball”).LocH =
   sprite(“Ball”).LocH + 50
 end if
 if sprite(“Ball”).rect[2] <= 0 then
  sprite(“Ball”).LocV =
   sprite(“Ball”).LocV + 50
 end if
 if sprite(“Ball”).rect[3] >=
   640 then
  sprite(“Ball”).LocH =
   sprite(“Ball”).LocH – 50
 end if
 if sprite(“Ball”).rect[4] >=
   480 then
  sprite(“Ball”).LocV =
    sprite(“Ball”).LocV – 50
 end if
 if isMouseDown then
  ball_so.data.x = sprite(“Ball”).LocH
  ball_so.data.y = sprite(“Ball”).LocV
 end if
 go to the frame
end
```

The complete listing of this sample is shown in Listings 3 to 5.

Now, you can publish the project and run two instance of it. Drag the ball sprite and see the effect in other running instance.

What's Next?

You've just experienced some of the basic concepts of using FlashCom in Director. But all of those samples are client-side only. When you learn how to use server-side scripting with Director to receive from and send messages to it, the power of your multi-user applications will increase greatly.

For example, with server-side scripting you can manage users and verify them to accept or reject their connection request, and so on. Using authentication, you could give different users different roles in the system, assigning some users more power in the system than others. For example, think of our chat system extended to include a teacher and several students. The teacher would have the ability to correct false information and push additional material that the students might not have. In a more game-like environment, a "Game Master" might have the ability to control the positions of objects that the individual players do not have the right to move. By exploring the capabilities of the Flash Communication Server, and its use in Director, several of these types of applications can be built using the shared object model.

Nima Azimi is a software engineer, multimedia project manager, consultant and programmer on variety projects. His projects include educational "How does it works" titles for children education with real-time 3D content. He has worked with Director for over four years, and he currently teaches courses in Director programming and multimedia. In his spare time he makes highly detailed photo-realistic 3D scenes as a 3D artist and writes video game scripts and gameplay ideas that he wishes to develop into full games at within the near future.

nima.azimi@gmail.com

listing 1

```
-- (movie script)
Global text_so

on StartMovie me
 the keyUpScript = “textChanged”
 member(“DirTypingStage”).text = “ “
end

on textChanged me
 if (the key) = Return then member(“DirTypingStage”).
text = “”
 text_so.data.textValue = member(“DirTypingStage”).text
end
```

listing 2

```
-- flash sprite’s script (behavior script)
property pSprite
property pNetConn
global text_so

on beginSprite me
 -- initialize a sprite reference
 pSprite = sprite(“myFlashSpriteName”)
end

on exitFrame (me)
 -- initiate the connection if necessary
 if voidP(pNetConn) then
  me.initiateConnection()
 end if
end exitFrame

on initiateConnection (me)
 -- create a new NetConnection object
 pNetConn = pSprite.newObject(“NetConnection”)
 -- declare an onStatus callback handler
 pSprite.setCallback(pNetConn,”onStatus”,#myOnStatus,me)
 -- connect the object to the server
 pNetConn.connect(“rtmp://localhost/dir_text/Room01”)
 -- get shared object from flash object
 tSharedObject = pSprite.getVariable(“SharedObject”,
false)
 text_so = tSharedObject.getRemote(“sharedtext”, pNet-
Conn.uri, false)
 text_so.connect(pNetConn)
 -- declare an onSync callback handler
 pSprite.setCallback(text_so, “onSync”, #syncTexts, me)
end initiateConnection

on myOnStatus (me, this, aInfo)
 put “Level: “ && aInfo.level && “Code:” && aInfo.code
 if (aInfo.code = “NetConnection.Connect.Success”) then
  -- Continue
 else
  -- Alert Message
 end if
end myOnStatus

on syncTexts(me, this, aList)
 Repeat with i = 0 to (i < aList.length)
  if (alist[i].name = “textValue” AND alist[i].code <>
“success”) then
   member(“DirTypingStage”).text = text_so.data.textValue
   exit repeat
  end if
 end repeat
end

on endSprite me
 -- close the connection
 pNetConn.Close()
end
```

listing 3

```
-- flash sprite’s script (behavior script)
property pSprite
property pNetConn
global ball_so

on beginSprite me
 -- initialize a sprite reference
 pSprite = sprite(“myFlashSpriteName”)
end

on exitframe me
 -- initiate the connection if necessary
 if voidP(pNetConn) then
  me.initiateConnection()
 end if
end

on initiateConnection (me)
 -- create a new NetConnection object
 pNetConn = pSprite.newObject(“NetConnection”)
 -- declare an onStatus callback handler
 pSprite.setCallback(pNetConn,”onStatus”,#myOnStatus,me)
 -- connect the object to the server
 pNetConn.connect(“rtmp://localhost/dir_sharedball/
Room01”)
 -- get shared object from flash object
 tSharedObject = pSprite.getVariable(“SharedObject”,
FALSE)
 ball_so = tSharedObject.getRemote(“position”, pNetConn.
uri, false)
 ball_so.connect(pNetConn)
 -- declare an onSync callback handler
 pSprite.setCallback(ball_so, “onSync”, #syncball, me)
end initiateConnection


on myOnStatus me, this, aArg2
 put “Level: “ && aInfo.level && “Code:” && aInfo.code
 if (aInfo.code = “NetConnection.Connect.Success”) then
  -- Continue
 else
  -- Alert Message
 end if
end myOnStatus

on syncball(me, this, aList)
 Sprite(“Ball”).LocH = ball_so.data.x
 Sprite(“Ball”).LocV = ball_so.data.y
end

on EndSprite me
 -- close the connection
 pNetConn.Close()
end
```

listing 4

```
-- ball sprite’s script (behavior script)
global isMouseDown

on MouseDown me
 isMouseDown = True
end

on MouseUp me
 isMouseDown = False
end
```

listing 5

```
-- (frame script)
global ball_so
global isMouseDown

on exitFrame me
 if isMouseDown then
  ball_so.data.x = sprite(“Ball”).LocH
  ball_so.data.y = sprite(“Ball”).LocV
 end if

 if Sprite(“Ball”).rect[1] <= 0 then
  Sprite(“Ball”).LocH = Sprite(“Ball”).LocH + 50
 end if

 if Sprite(“Ball”).rect[2] <= 0 then
  Sprite(“Ball”).LocV = Sprite(“Ball”).LocV + 50
 end if

 if Sprite(“Ball”).rect[3] >= 640 then
  Sprite(“Ball”).LocH = Sprite(“Ball”).LocH - 50
 end if

 if Sprite(“Ball”).rect[4] >= 480 then
  Sprite(“Ball”).LocV = Sprite(“Ball”).LocV - 50
 end if

 go to the frame
end
```

Part 1

Uncovering Fireworks Masks

It is not so much what is behind the mask that matters – just as Yeats so famously describes – as is what is revealed. This is certainly true of graphical masking techniques in general, and particularly in the case of Fireworks masks where you have so many options for hiding and revealing your creations to the world.

by kim cavanaugh

"It was the mask engaged your mind, And after set your heart to beat, Not what's behind."

–William Butler Yeats, "The Mask"

A mask is simply a graphical object that is placed on top of another image and given instructions on how it should interact with its partner. Masks are always a pair of objects – the masking object and the object that is being masked. The beauty of masks is that they allow you to change a design or image without ever changing the image below the mask. You decide how much to reveal or hide based on your choice of masking object and the way you apply it.

Fireworks allows you to create masks with either a bitmap image or with a vector object that you've created. This flexibility gives you many options, but also may leave you with a question. "Which one is the right one to use?" This article and the one to follow next month will try to remove some of the mystery behind masking and give you practical examples of when to use a bitmap mask and when a vector mask is more appropriate.

Masking Defined

When one graphical object interacts with another as a mask, new transparency values are created for the underlying, or masked, object based on the color of the masking object. If a mask is composed entirely of white colors, the underlying image will be completely opaque. If the mask is made up of nothing but black, the object below will be entirely transparent. That's a simple enough concept, but it's really the interplay that you can create between the range of colors and shades between white and black that makes masks so powerful. One of the most common uses of masking is the fading of images from opaque to transparent, created by using a vector gradient fill that transitions from white to black. As the colors in the masking object become progressively darker, the underlying image is transformed and becomes progressively transparent.

When a bitmap object such as a photograph is used as a mask, you have a whole new range of creative possibilities. Bitmaps – with their more organic appearance and the millions of colors that might be in place in the image ¬– allow for a far wider range of interactions than are possible with vector objects. This allows you to create effects such as you see in Figure 1, where a simple photograph has been used to mask a text object.

Masking with Bitmaps

The uses of bitmap objects in Fireworks can be broken down into 3 broad categories based on the effects they achieve:

1. A bitmap mask can be used to remove portions of an image by erasing or changing parts of the bitmap object.
2. Painting on top of an empty bitmap object allows you to reveal portions of an image depending on the settings used for the digital paint you apply.
3. Digital photographs can be combined with an object to provide interesting new textures, tints, and patterns to a composition.

In the following mini-tutorials you'll see how all three effects are achieved.

Fireworks Interface Tools for Masks

Before moving on to the "lessons on creating bitmap masks," a quick review of the way that Fireworks allows you to apply and control bitmaps is in order. Figure 2 shows the Fireworks Layers panel with the controls you'll use for creating and controlling bitmap masks labeled.

Any time a mask is created in Fireworks, the Layers panel will display a thumbnail of the masking object side-by-side with the masked object. The chain link icon that separates the two lets you see at a glance that a mask is in place.

Remember that one of the beautiful things about masks is that, while it appears that you're modifying the masked object, in reality you're only changing the mask. No pixels are damaged or removed during the masking process, which means that if things go astray you can simply remove the mask and start over again. Your original image is never changed when masks are used.

At the bottom of the Layers paneltake note of the two small buttons that allow you to generate new bitmap masking objects. You'll see those two buttons in action in the first two tutorials in this article.

Painting Over Bitmaps: Create a Funky Image Edge

A question that I used to hear pretty often at the Fireworks forum was "How do I get an irregular edge around a photograph?" It's surprisingly easy with the tools that Fireworks gives you. In this first lesson you'll learn how to apply a bitmap mask and then paint around the perimeter to get one of those funky edges.

1. To begin, create a new Fireworks document and choose File > Import. Locate a photograph that you want to practice with and click the Open button. When the insert cursor appears (a sideways "L"), click once to paste the image onto the canvas. (For my examples here I'm using the sample images that ship with Windows. You can use any digital image of your choice.)
2. Open the Layers panel and click the Add Mask button shown here.

   Adding a bitmap mask with this button creates a bitmap object over your imported image that is completely filled with white pixels. Since white pixels allow the underlying image to remain opaque, it doesn't really appear as if much has happened. The real magic takes place by removing or painting over some of those white pixels.
3. Select the Paintbrush tool from the Bitmap group in the Tools panel as shown here.
4. Since the goal here is to hide pixels around the edges of this image, you'll need to choose a black stroke color for your Paintbrush. Painting on top of the existing white bitmap object with black will cause those areas to be hidden. What's cool about using a bitmap brush for this effect is that you can play with the different settings for stroke and edge types to get some very interesting effects. In addition, painting with bitmap pixels in Fireworks allows you to build up paint on top of the mask by repeatedly painting over the mask. This allows you to create a range of opacity with some areas remaining opaque, some completely transparent, and gives you a wide range of possibilities in-between.

In this example I used the following settings:

- Stroke type: Watercolor, Heavy
- Edge transparency: 50
- Edge texture: Confetti

Using the Property inspector match those settings or choose settings of your own and stroke around the outer perimeter of your canvas. As you stroke on more black pixels more of the underlying image will become transparent. In the end you should have a new creation similar to the one you see in Figure 3.

This technique gives you a great deal of freedom for creating irregular image edges and for cool effects that move beyond simple rectangular shapes. By experimenting with different edges, textures, and buildup of paint your options of making those funky edges are nearly limitless.

figure 1

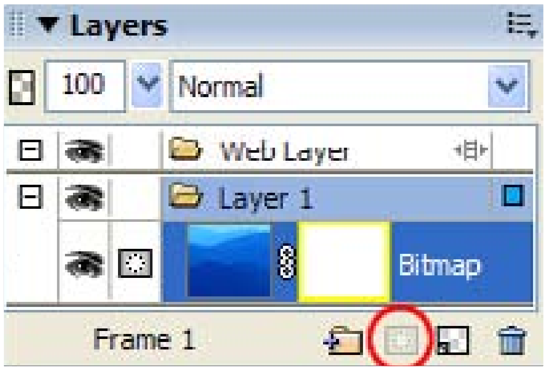
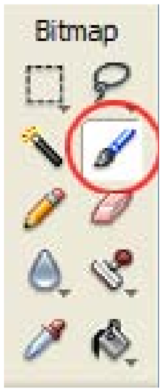


figure 2



Painting on a Blank Bitmap: Revealing an Image

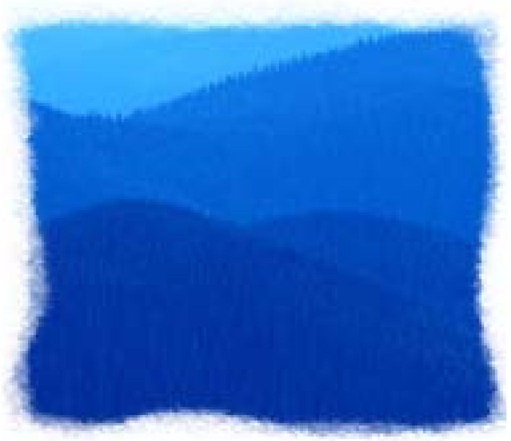
Now that you've seen how to paint on top of an existing bitmap, let's look at the second method for working with bitmap masks. In this technique you'll start with a new blank bitmap object, and by painting on top of the mask, determine what parts of the underlying image is to be revealed.

As with the previous technique, painting on top of an empty bitmap with the Paintbrush tool allows you to get some cool effects by setting the Stroke and Edge settings for the tool. Let's put this method into action.

1. Create a new Fireworks document and import the digital photo you'd like to work on by choosing File > Import. Paste the bitmap object into place by clicking on the canvas when you see the Insert cursor.
2. To add an empty bitmap object to the canvas select on the button you see highlighted in Figure 4, or choose Edit > Insert > Empty Bitmap. Either method will place an empty bitmap above the imported image on the canvas.
3. Once again the Paintbrush tool allows you to build up paint onto the canvas with a great deal of flexibility. Select the Paintbrush and set the settings in the Property inspector to the following:
   - Color: White
   - Size: 25 pixels
   - Stroke type: Unnatural > Fluid Splatter
4. Before painting, check the Layers panel to be sure that the empty bitmap object – represented by the checkerboard thumbnail in the Layers panel – is highlighted in blue to indicate that it is selected.
5. Using the Paintbrush, stroke color over your image. As you can see in Figure 5, I painted the bitmap I used in a spiral pattern so the center of the painted area is filled with color.
6. Once you have your painted mask all set, hold down the Shift key and select both the bitmap and the image that you're masking. You can do that directly on the canvas or select the thumbnails in the Layers panel.
7. Now choose Modify > Mask > Group as Mask. Your painted mask will now reveal the image below it, as you see in Figure 6.

Combining a bitmap mask with the ability that the Paintbrush tool gives you allows you to generate some fascinating images. No longer constrained to those basic rectangular boxes, masking with an empty bitmap gives you complete control over what parts of an image you'll reveal to the world.

figure 3



Masking with an Existing Image: Towards More Organic Graphics

Vector images are terrific things. Small in file weight, and incredibly easy to modify, vectors are amazingly versatile. Vectors have a drawback though in that they often appear plastic, and lack life and character. Sure, you can add textures, and even a bit of noise in Fireworks MX 2004, but the vector format itself is limited when you want to create a graphic that is a little more lifelike, or organic.

There's a solution though, and once again the secret is in the masking process. By using a bitmap object as a mask you can start with a basic vector object and then add a realistic texture that takes advantage of the subtle shades and variations in texture that only the real world provides.

In this final lesson you'll see how to apply a bitmap mask over the top of some vector-based text. As you'll see, the technique is similar to what you've done before in that you'll place the masking object over the top of the object to be masked and Voila: Organic images!

1. Create a new Fireworks document and add some text to the document. For my example I typed in the text "Wintry Days" in a large heavy font. I chose Copperplate Gothic Bold and set the font size to 58.
2. Set the font color and a stroke to your liking. My example uses a medium blue fill with a stroke set to Crayon > Basic, with a width of 2 pixels, seen here.
3. Choose File > Import and locate an image to your liking. I chose the file named winter.jpg that Windows users will find in their Sample Pictures folder.
4. Increasing the contrast before the mask is applied will improve the masking effect that you're about to apply. A simple method for getting this done is to turn to the Effects area of the Property inspector and click the Plus sign to add an effect. Choose Sharpen > Sharpen More to really make the contrast in the image pop.
5. Position your bitmap over the top of the text. This is a good time to scale the image, and even turn down the opacity of the object so you can get it just where you want it. Figure 7 shows an image prepared and positioned

over the top of the text.

6. Once the masking object is in place turn the opacity back to 100%. Hold down the Shift key and select both the text and the bitmap. Remember that this can be done either on the canvas, or by using the Layers panel and selecting the thumbnails.
7. With both objects selected choose Modify > Mask > Group as Mask. Not too bad right? Figure 8 shows the final results of this operation.
8. Once the mask is applied you can tweak things a bit by selecting the thumbnail of the bitmap in the Layers panel and use Filters to adjust the color further. Look for the yellow border to appear around the masking object in the Layers panel so you know it's selected.

Filters are a destructive process, so go slowly and undo if you don't like the effect you achieve when the mask is modified. Unlike Live Effects you won't be able to change the image again after the file is saved. In Figure 9 you see the results I achieved by choosing Filters > Adjust Color > Brightness and Contrast.

And there we have our lovely masked image. Like so many creative techniques it takes a bit of time to develop the knack for using bitmaps in this manner. You'll get the best results when images that have sharp contrast between light and dark areas are used. Take some time to play with different settings and combinations of effects and Filters until you're able to get your own unique designs looking exactly the way you want.

figure 4

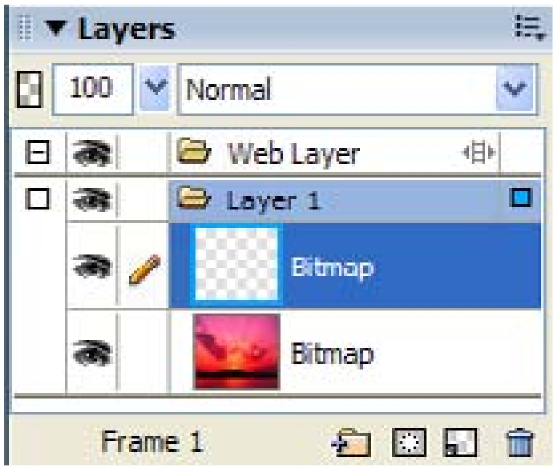


figure 5

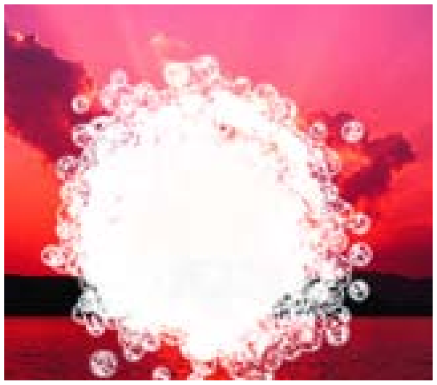
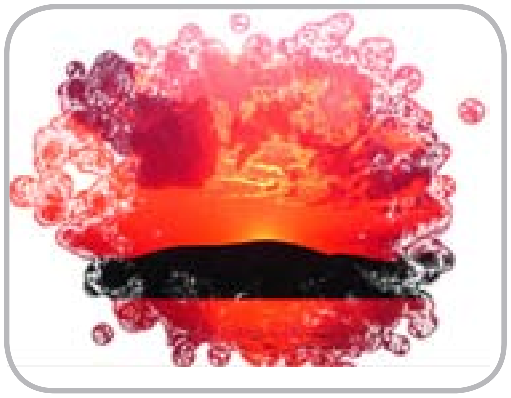


figure 6



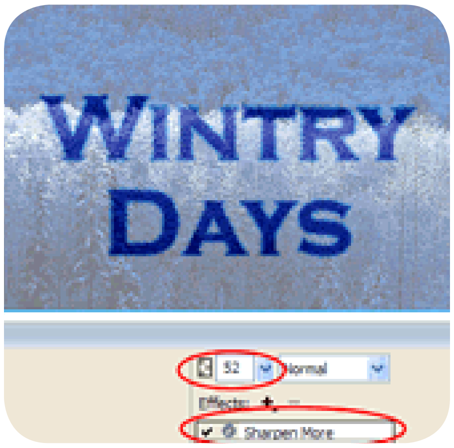
Conclusion

In this article you've seen three ways that bitmaps can be used as masks. You've learned that masks can be used to hide parts of an image, to reveal parts of an image, and for adding organic textures to objects. All of these processes are easy enough to apply. It's up to you to take things to the next level in your own designs.

In the next article in this series you'll learn how to use vector objects as masks. Using vector-based shapes and the careful application of color fills you can remove backgrounds from a photograph, fade images in a multitude of different ways, and even tint your images for some interesting creative effects. As Yeats would say, it's all about what those masks reveal, just as it is with bitmap masks.

Kim Cavanaugh has been teaching and writing about web design software from Macromedia for over 5 years. He has written two books about Dreamweaver and Fireworks, collaborated on books about Dreamweaver, Fireworks, Flash and Contribute, and continues to write extensively about Studio MX tools for CommunityMX.com. In addition to his tutorials at CommunityMX, you can find more of his tutorials at his Beginner's Guide website (www.dw-fw-beginners.com) and read about things that interest him at his BrainFrieze blog (www.brainfrieze.net).
cavanaug_l@firn.edu

figure 7



52 Normal
Effects: + −
✓ Sharpen More

figure 8

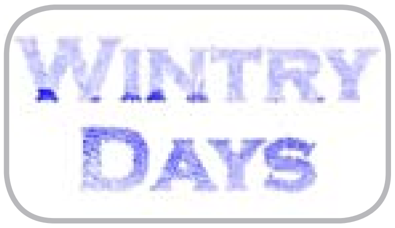
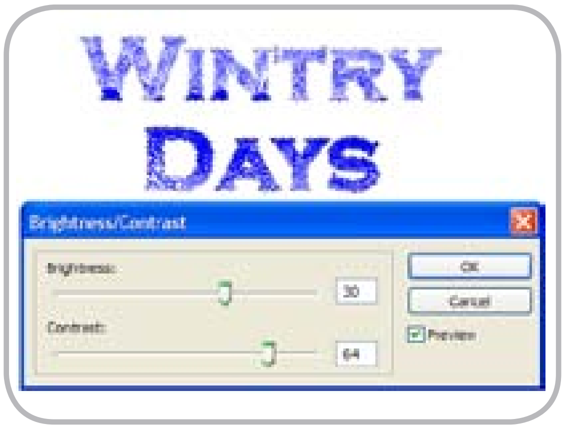


figure 9



Brightness: 30
Contrast: 64
✓ Preview

Printing Rich Document Formats with CFMX 7

How to print Web pages in FlashPaper or PDF Format

by xu chen & sherman gong

Most of us at one time or another have experienced the poor result of printing Web content from a browser. The page printout is ugly because the printer breaks the Web content into pages with borders and edges. Trying to fix the HTML code with style sheets and other layout tricks still yields an unsatisfactory outcome. You, the developer, and your end users desperately need a solution for printing rich document formats.

Likewise, if you are on the road without Internet access and want to pass your work to a client who is outside your company firewall, you need a way to distribute documents easily.

You've already invested an enormous amount of time and resources setting up and publishing Web pages and articles so they look just the way you want. You don't want to rework them just to generate a rich document – you need an easy conversion tool.

Introducing the cfdocument tag. This new ColdFusion MX 7 feature takes your current HTML/CFML pages and converts them into Macromedia FlashPaper or Adobe PDF formats in seconds. Best of all, using this tag requires no learning curve. In this article, we explain how the ColdFusion team created this new functionality and how you can use it to create printable Web documents.

Requirements

Software: To complete this tutorial you will need to install the following software and files: ColdFusion MX 7 (http://www.macromedia.com/cfusion/tdrc/index.cfm?product=coldfusion&promoid=devcenter_tutorial_product_090903).

Prerequisite knowledge: Familiarity with ColdFusion tag syntax

Your Need for the *cfdocument* Tag

When it came time to brainstorm for features in ColdFusion MX 7, many ColdFusion team members wondered what we could give developers, so that you would get even more use out of

table 1

cfdocument Attributes

Attributes	Required	Optional	Functionality
Format	yes	no	Dictates what format to generate (PDF or FlashPaper)
MarginTop	no	yes	Defaults to 0.5 inches (“Unit” attribute changes unit)
MarginBottom	no	yes	Defaults to 0.5 inches (“Unit” attribute changes unit)
MarginLeft	no	yes	Defaults to 0.5 inches (“Unit” attribute changes unit)
MarginRight	no	yes	Defaults to 0.5 inches (“Unit” attribute changes unit)
BackgroundVisible	no	yes	Allows background color or images to appear (disabling this attribute speeds document processing and limits memory usage)
Orientation	no	yes	Specifies landscape or portrait
PageType	no	yes	Specifies letter, legal, A4, A5, B4, B5, B4-JIS, B5-JIS, and custom
PageWidth	no	yes	Specifies page width when selecting “Custom” PageType
PageHeight	no	yes	Specifies page height when selecting “Custom” PageType
Encryption	no	yes	Sets 128-bit, 40-bit, or none (PDF-only feature)
OwnerPassword	no	yes	Sets owner password of the document when encryption is on
UserPassword	no	yes	Sets user password of the document when encryption is on
Permissions	no	yes	Establishes permission types
Unit	no	yes	Sets inches or centimeters
FontEmbed	no	yes	Embeds font with the document
Filename	no	yes	Chooses where to save the file
Overwrite	no	yes	Overwrites file if it exists
Name	no	yes	Saves result content to a CF variable
Scale	no	yes	Specifies zoom factor of the document

ColdFusion. One theme we heard consistently was how hard it is to create portable rich document from Web pages. We thought, wouldn't it be nice for developers to be able to add a button to any Web page, which would convert rich HTML content into easily printable PDF or FlashPaper documents?

Your request resonated with the team. We thought about a ColdFusion tag, one that consumes HTML content and converts it to rich documents. Last year, during the MAX 2003 conference, we demonstrated this idea in the sneak-peek session. We heard a resounding approval from you when we told you the possibility of providing this functionality in the next version of ColdFusion – it was that approval that helped us decide to make it part of ColdFusion 7.

The development team faced many challenges during the design and implementation phase. Foremost, HTML is a very lenient language. HTML writers can get away with broken syntax on modern browsers. This factor created an enormous challenge for parsing data and converting it to PDF/FlashPaper-specific "language." During our research, we found that there are some products out there already trying to address this issue. However, most of them either require an extra installation procedure, such as a C++/native solution that is not platform-friendly, or require an expensive add-on OEM with ColdFusion.

One technology that came close to adoption was *Apache FOP*, which converts XML to PDF by relying on XSL (style sheets) masking. The main advantage of FOP is that it deploys on top of Java, the technology that ColdFusion MX standardized upon. However, the ColdFusion team found that the Apache FOP engine only consumes parsed XML content. It applies the XSL style sheet for layout information and then translates it to native PDF content. Adopting FOP would have required ColdFusion developers to edit their HTML into XHTML, which would have increased the difficulty of their using this feature and the learning curve for developers. This was out of the question. We researched the idea of autogenerating XHTML from HTML but that was extremely error-prone and raised other issues.

Therefore, we settled on our own solution – one that allowed us to consume regular HTML 4.01 and CSS 1 and 2 without requiring any modification by the developer. This greatly enhanced the developer experience, increased the value of this feature, and – at the same time – met all our original goals.

Easy-to-Use Solution for Developers

As ColdFusion engineers, we like to think that our job is to make ColdFusion developers' lives easier. This was the exact mindset we applied while working on this task. We thought, what can we create so that all that developers need to do is wrap their HTML with a ColdFusion tag, so that conversion occurs once the browser invokes the Web request? Thus, the *cfdocument* tag syntax was born. Here's an overview of the syntax:

```
<cfdocument format="flashpaper/PDF">
<!—insert your HTML, CFML, and cfdocu-
mentitem tags->
</cfdocument>
```

This is it! No messy XML, Java *classpath* configuration, or registering native libraries. Best of all, it is a part of the ColdFusion engine. Once you install ColdFusion, this feature is available to you. You don't need to change anything in your existing HTML/CFML content.

In addition to the simplicity of the tag's functionality, we also added attributes to help you manipulate document layouts (see Table 1). The only tag requirement is the *format* attribute, which specifies to the ColdFusion engine which type of file to generate (PDF or FlashPaper); the others are optional attributes.

Besides providing basic printing functionality, the *cfdocument* tag also had to be robust and conform to standards. Our first priority was to support HTML 4.01. This standard was required before anything else because most of our users use the HTML 4.01 standard. Second, CSS versions 1 and 2 were required because style sheets are an important part of HTML formatting, making it a feature requirement.

The remaining tasks were to provide image, HTML links, security, and accessibility support so that generated rich documents act similarly to browser-rendered content. Furthermore, to support fonts that a user specifies within HTML content, the team built the *cfdocument* tag to work with the ColdFusion font manager to locate necessary fonts on the system. We took some of the PDF-specific functionality and made bidirectional language (such as Arabic, Hebrew) support possible for PDF-formatted documents.

Using the cfdocument Tag

This is the most exciting part of introducing this feature – showing you how to use it!

Sample 1: Using the cfdocument *with cfhttp tag*

This is what makes the *cfdocument* tag powerful. There is no need to edit your current HTML. You just redirect the content of the HTTP request through the *cfdocument* tag (see Figure 1):

```
<cfhttp url="http://www.w3.org/TR/REC-
html32" method="get" resolveURL="true">
<cfdocument format="flashpaper">
   <cfoutput>#cfhttp.filecontent#</
cfoutput>
</cfdocument>
```

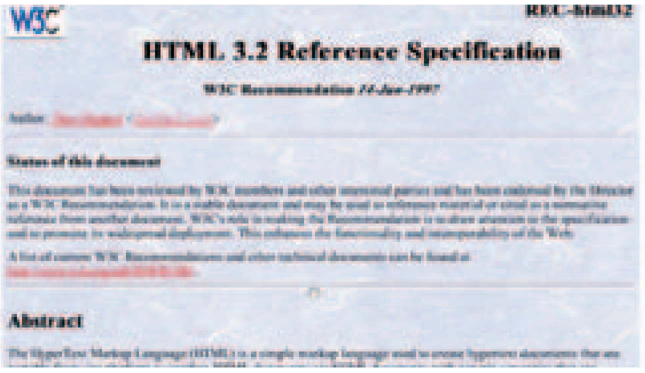
Sample 2: Using the cfdocument *tag with existing third-party report output*

If you use a third-party reporting engine

Xu Chen has been a senior software engineer with Macromedia since the days of Allaire Corporation (in 2000). He worked on ColdFusion 5 and 6, helping to migrate ColdFusion from C++ to the Java platform. He just finished working on the release of ColdFusion MX 7. He has over a decade of software development experience.
xchen@macromedia.com

Sherman Gong joined Macromedia in the midst of ColdFusion 5 development and has been with the company as a principal software engineer since 2000. He has 15 years of experience in software development, ranging from aerospace systems to financial trading systems. For the ColdFusion MX 7 release, he was responsible for font management tasks, cfreport, and FlashPaper output in the cfdocument tag.
sgong@macromedia.com

figure 1


REC-html32

HTML 3.2 Reference Specification

W3C Recommendation 14-Jan-1997

Author: [illegible]

Status of this document

[illegible]

Abstract

[illegible]

and plan to output the results in HTML, you can use the *cfdocument* tag to help create a portable document exactly as it appears. This saves much time and effort on your part because you don't need to recreate reports just for the purpose of printing to rich document formats. You simply wrap your query/report result with the tag and it gives you the desired printing result (see Figure 2).

Sample 3: Creating portable documents for easier sharing

Besides printing in a visually appealing format, you can use the *cfdocument* tag to create documents that are portable and that you can combine with the cfmail tag to send output to other users:

figure 2

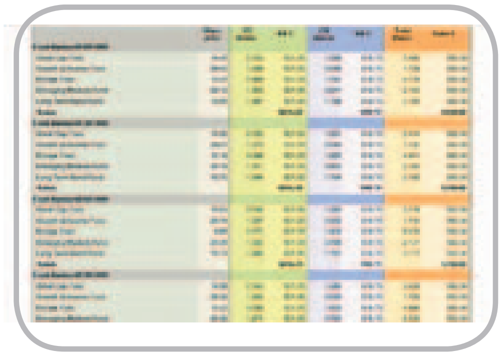


figure 3

```
<cfhttp url=”http://www.w3.org/TR/REC-
 html32” method=”get” resolveURL=”true”>
<cfdocument format=”flashpaper”
  filename=”c:\temp\w3spec.swf”>
   <cfoutput>#cfhttp.filecontent#</
     cfoutput>
</cfdocument>
<cfmail to = “recipient”
  from = “sender”
  subject = “msg_subject”
MIMEAttach = “c:\temp\w3spec.swf”>

Hi John, here is the spec you were
looking for.

Regards,

</cfmail>
```

Sample 4: Bidirectional text support

If you write text from right to left, you can now use the *cfdocument* tag to print (only available in PDF).

Code Examples for the cfdocument Tag

There are a few additional supporting subset tags for the *cfdocument* tag: *cfdocumentitem* and *cfdocumentsection*. These two tags provide additional format control.

The *cfdocumentitem* tag has three different attributes: *header, footer,* and *pagebreak*. Their attribute names indicate their functionality. The *header* and *footer* attribute values support document headers and footers, while the *pagebreak* attribute value specifies an inserted page break in the document:

```
<cfdocument format=”pdf”>
  Hello World!!!
  <cfdocumentitem type=”pagebreak”/>
  <cfdocumentitem type=”header”>
     Company Name
   </cfdocumentitem >
   <cfdocumentitem type=”footer”>
<div align=”center”>
<font color=”navy” size=”1”
  face=”Tahoma”>
page
<cfoutput>
  #cfdocument.currentpagenumber#
</cfoutput>/
     <cfoutput>
       #cfdocument.totalpagecount#
     </cfoutput>
     </font>
     </div>
  </ cfdocumentitem >
  <!-- insert your other HTML text
    here-->
</cfdocument>
```

You use the *cfdocumentsection* tag to break HTML into different segments. You can specify unique headers, footers, and other HTML style for each section without affecting other parts of the document:

```
<cfdocument format=”flashpaper/pdf”>
  <cfdocumentsection margintop=”1”>
  <cfdocumentitem type=”header”>
Company Name
  </cfdocumentitem >
  <cfdocumentitem type=”footer”>
<div align=”center”>
<font color=”navy” size=”1”
  face=”Tahoma”>
page
<cfoutput>
  #cfdocument.currentpagenumber#
</cfoutput>/
<cfoutput>
  #cfdocument.totalpagecount#
</cfoutput>
</font>
</div>
  </ cfdocumentitem >

  <body style=”background-color:
   #dddddd”>
  Hello World!!!
  </body>
</cfdocumentsection>

<cfdocumentsection margintop=”2”>
<cfdocumentitem type=”header”>
  Different Header
</cfdocumentitem >
<cfdocumentitem type=”footer”>
  different footer
</ cfdocumentitem >

<body style=”background-color:
  #eeeeee”>
 Another Section
</body>
</cfdocumentsection>
</cfdocument>
```

The user-requirements criteria – ease of use and standards support – were key factors in creating the *cfdocument* printing functionality. The *cfdocument* tag gives you a tool that you can use immediately.

We'll continue to refine this important feature in the future, and we welcome your feedback. We hope you are able to make use of this great new functionality wherever you need to print Web content.

Credits

Xu Chen and Sherman Gong developed this technology jointly. Xu designed the *cfdocument* architecture and provided the tag's implementation and PDF output format. Sherman Gong provided the FlashPaper format support and font management, and worked on the links and anchors support with Xu. Hiroshi Okugawa and Collin Tobin provided quality assurance for the *cfdocument* tag and numerous other ColdFusion features.

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...

*Offering leading software, services, and hardware vendors an opportunity to **speak to over 100,000 purchasing decision makers** about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management*

Don't Miss Your Opportunity to Be a Part of the Next Issue!

Get Listed as a Top 20* Solutions Provider

For Advertising Details Call 201 802-3021 Today!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE

SYS-CON MEDIA

The World's Leading i-Technology Publisher

Creating Better Forms Faster with ColdFusion MX 7

Macromedia makes time to give the cfform tags some love

by mike nimer

When the ColdFusion engineering team started planning ColdFusion MX 7, we finally had time to give the cfform tags some love. We talked to and heard from many customers and knew that we needed to do a cfform tag overhaul for HTML-based forms. We also knew that that if it was possible, we wanted to help ColdFusion developers harness the richness of Macromedia Flash Player and open up the cfform tag to the incredible flexibility of XForms. We knew that if we could accomplish all of this with a simple and easy cfform tag syntax, it would be a big win for you, the developer, and your application users.

Originally, ColdFusion cfform support was just a few tags that generated some basic JavaScript validation – validation that certainly had its critics. This all changed with ColdFusion MX 7. We decided it was time to throw out the tedious approach of creating forms and breathe some new life into the cfform tag.

First we had to review the enhancement requests and bug reports for the current cfform tag to make it do what you have wanted and needed it to do since we introduced it. To achieve this, we had to rewrite old tags from scratch, update validation, add new tags, and fix a few bugs (OK, fix a lot of bugs).

After making a solid foundation, we rethought the cfform tag functionality altogether. Although this was the second step in our development process, we started this step years ago. When the merger between Allaire and Macromedia occurred, there was a lot of talk about generating Flash from ColdFusion – specifically forms. During the development of ColdFusion MX 6, there were numerous "what if" discussions about the benefits of being able to use layout managers with HTML forms, which is similar to defining forms in Java Swing applications. However, it wasn't until ColdFusion MX 7 that we were able to spend the time to do this right.

This article was originally published on the Macromedia Developer Center http://macromedia.com/devnet/.

Updating the cfform Tag

As with all product development, you must start with evaluating the foundation. Thus, before we could add all sorts of cool new features to the form tags, we needed to make the foundation rock-solid. We spent the first cycle of development upgrading the cfform tags. What did this involve?

You may not know this, but some ColdFusion tags are not written in Java; they are actually written in CFML for use in your CFML pages, just like the ColdFusion custom tags you write yourself. However, the way ColdFusion custom tags associate with their parent wasn't flexible enough for the plans we had for adding XML and Flash forms support to the form tags. So, the first step was to rewrite these tags from scratch as Java tags, while at the same time adding all of the missing HTML tags that the cfform tags didn't generate. Personally I always hated that I could use cfinput to define a text field but not a submit button. One of the joys of being both a user and developer of the product: I can fix my own pet peeves.

Once we finished rewriting the new tags, we dug through the bug tracker. We knew there were numerous enhancement requests for the cfform tag that customers frequently requested through mailing lists and forums. To ensure that we didn't miss anything, we searched for all open bugs and enhancement requests submitted all the way back to ColdFusion 3 and then transferred them into the ColdFusion MX 7 queue. While reviewing all of these requests, we still decided to defer or close some of the bugs and enhancements, but we wanted to ensure that we evaluated every single open bug and enhancement. (By the way, this was a common task for most of the ColdFusion MX 7 features, not just the cfform tags.)

Evolving Form Validation

At this point, the cfform tags looked pretty good: they were a lot faster and worked a lot smoother, but something else was missing. For years, users had requested new validation types, such as e-mail and URL. There had also been many complaints about having only JavaScript validation for the client side (what if the client has JavaScript disabled?). Other complaints were about the server-side validation we supported (it required developers to use HTML hidden fields). In addition, when you used server-side validation, you couldn't validate all of the same types of data as you could with client-side JavaScript validation, which caused a frustrating discrepancy between different validations techniques.

To fix these issues, we synchronized all of the validation routines from the client-side JavaScript with the validation libraries on the server. There were a few differences we couldn't change for backwards compatibility. For instance, server-side date parsing validation allows more date formats than the client-side JavaScript validation; however, if you call e-mail validation on the client or on the server now, ColdFusion runs the same regular expression to validate it. Of course, calling these validation routines was still annoying because you still had to code an HTML hidden field to trigger each field. Although using hidden fields is still a useful way to validate data on intranets or other applications where you control who uses your application, there is a problem using hidden fields to validate data on a public site: someone could hack your site and remove the hidden fields, bypassing any validation you thought was in place. We did a few things to resolve this problem.

If you use the hidden fields, ColdFusion MX 7 automates their creation. As you may know, previously you had to code an input field and a hidden filed for server-side validation. Now you don't need to remember the cryptic syntax to trigger the validation in ColdFusion.

Before:

```
<cfinput type="text" name="fname"
   required="true">
 <input type="hidden" name="fname_
  required" value="Fname is required">
```

After: (in ColdFusion MX 7):

```
<cfinput type="text" name="fname"
  required="true" validateat="
  onServer">
```

You use the new *validateAt=""* attribute to define three different location values to validate: *onSubmit* (this is the default option), *onBlur* (a new validation location type), and *onServer*. You can mix and match these values or specify the validation locations you would like to use as a comma-separated list:

```
validateAt="onBlur,onServer"
```

On a ColdFusion page – especially a form action page – it is a common and best practice to validate all variables passed into the page. To make this easier for you, we added access to all server-side validation routines in the cfparam tag through the type attribute. Thus, you can verify whether the variable exists and, at the same time, validate the value as a specific data *type*, all in one tag (this even includes the powerful "*RegEx*" validation type).

Example:

```
<cfif isDefined("form.submit")>
     <cfparam name="form.email"
       type="email">
     <cfparam name="form.password"
       type="regex" pattern="
       [a-z]{1,3}[0-9]{6,10}">

     <!---
Your custom form submission logic
  goes here
--->
</cfif>
```

Because this may not work for all solutions, we didn't stop with the cfparam tag but, rather, added a new ColdFusion function, the *IsValid ()* function as well. Use this function to call validation routines against any variable (for instance, you might want to validate against a value you query from a database).

Example:

```
<cfif isValid("email", myQuery.
  usersEmail)>
     <cfmail from="webmaster@mysite.
       com"
          To="#myQuery.usersEmail#"
          Subject="System Update">

          This is to inform you that
            at 2pm today ...
     </cfmail>
</cfif>
```

Again, we could have stopped here, because the cfform tags were now a solid tag set with plenty of new functionality. Instead, we knew that this new foundation gave us stability and an opportunity to have some fun, reinventing the way you build forms in web applications.

New Layout Managers

When building web applications, I noticed that I spent the majority of my time building forms – creating layout and user interfaces. I don't know about you, but I prefer to code the back-end logic and functionality, not the look and feel. To make matters worse, all my forms began the same way: a two-column table with the label in the left column, setting the width to 150 pixels, and a form field in the right column. It's a nice and simple form (you've built this form haven't you?). It always seemed futile to me to keep rewriting this same table again and again. The problem was, I only cared about a dozen or so form elements, not the 200–300 lines of HTML code that wrapped them.

So we started thinking, wouldn't it be cool if developers could use layout managers like they do in other languages, such as in Java Swing applications? The problem is that the web and HTML do not have layout managers; they use tables, div tags, and span tags with absolute and relative positioning – in essence, they have no automatic layout logic.

Luckily for us, many pieces came together at the same time. First, the Macromedia Flex team was working on layout managers for Flash forms based in Flex. Second, Flex is based on XML. For years we wanted to give you the ability to generate Flash from ColdFusion and give you the power of rich forms available with Flash components. With Flex, we noticed that our prototype for XML forms – based on the XForms schema –would map very easily to the MXML (the Flex markup language) we needed for Flex. This relationship allowed us to expose the same CFML syntax for the layout managers in both XML and Flash forms.

What do these layout managers mean for you? It means you now can develop forms with layout managers, giving you more time to spend on your application's functionality instead of its layout.

Even though ColdFusion MX 7 passes the job of managing the layout to the layout managers, you still cannot create forms 100% automatically. There are just too many different ways to lay out a form.

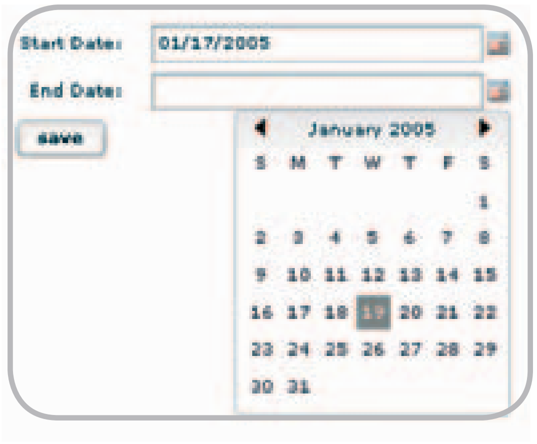
"Wouldn't it be cool if developers could use layout managers like they do in other languages, such as in Java Swing applications?"

As you know, when you leave it to an engine to guess what you want, things never work the way you might like and you may never use it.

The key was to find a middle ground that everyone could work with. This middle ground had to remove the bulk of code you would write to lay out a form, yet still provide a way for you to define some layout. To do this, we added a new ColdFusion tag called cfformgroup. You use this new tag to give formatting "hints" to the layout managers.

The layout managers use a number of rules to define the layout – such as defined widths, minimum widths of components, overall form size, available room left in the form, size of labels, and so forth. Because of these rules, the groups only provide hints, not hard-and-fast

figure 1



rules. The layout managers try to do what the group specifies, but if the fields don't fit, the managers lay out the fields in an alternative format, different from what you defined.

For instance, if you create a group using the type="horizontal" attribute, but you have 10 form elements in the group, odds are that 10 form fields side by side will appear wider then the form's space limit. In this case, the layout manager does what it can. Most likely, it will lay out three or four fields side by side, then output another three or four fields on the next line, and then output a new line with the last three fields.

This is different than when you lay out your form with the old HTML table methods, where each field lies in its own table cell. In this old scenario, the form renders off the side of the page to fit. When you find the layout managers rendering an unexpected output in ColdFusion MX 7, adjust the width of the form elements; usually reducing the width of the form fields a little can make the difference. If that doesn't work, increase the overall form width or remove the individual form element labels and use group labels instead. The following example demonstrates how you can use the cfformgroup tag to place the FirstName and LastName fields next to each other, with one shared label:

```
<cfformgroup type="horizontal"
   label="Enter Name:">
     <cfinput type="text"
        name="FirstName" value="">
<cfinput type="text" name="LastName"
   value="">
 </cfformgroup>
```

Better User Experience with Flash Forms

Layout managers were not the only reason we added the support for Flash forms. The desire for richer form controls and lack of cross-browser issues were contributing factors as well.
To the cfformgroup tag , we also added some groups for Flash forms that did more than just lay out alignment – groups could now serve as layout containers as well. Examples of these containers are the *tabnavigator, accordion*, or *panel* layout containers.

In the following example, you can create tabs in your form using the *tabnavigator* container:

```
<cfform format="flash">
 <cfformgroup type="TabNavigator">
     <cfformgroup type="page"
       label="Tab 1">
         <!-- your form elements go
           here -->
     </cfformgroup>

<cfformgroup type="page" label=
  "Tab 2">
         <!-- your form elements go
            here -->
     </cfformgroup>

<cfformgroup type="page" label=
  "Tab 3">
         <!-- your form elements go
            here -->
     </cfformgroup>
 </cfformgroup>
</cfform>
```

Perhaps the greatest reason for adding Flash forms support – a reason that was by far the most requested enhancement in ColdFusion since the merger of Allaire and Macromedia – was to replace the Java applets, specifically the cfgrid and cftree controls. For those of you who like the Java applets, don't worry, we kept them. But we added new Flash versions of the cfgrid and cftree controls (there are other new versions of the cfgrid and cftree tags as well: XML and object). These two controls are similar, but not identical, to the original versions. There are still some things that the Java applets do that the Flash controls cannot. Likewise, there are new things that the Flash cfgrid and cftree controls can do that the Java applets cannot. Use them; try them both. There is a time and place for both controls in your applications. For instance, the Java applets can handle a lot more data than the Flash controls. So if you need to output thousands of rows of data in a grid, use the Java applet controls.

Along with the new cfgrid and cftree controls, we added two new rich Flash controls to ColdFusion for dates and selecting dates. One of these controls is the cfcalendar tag you can use inside Flash, XML, and HTML forms (see Figure 1), just like the cfgrid and cftree controls. Another new control is the *dateField* type value for the cfinput tag. Use this control to create a text field with a pop-up calendar.

Note: You can only use the *dateField* type inside a Flash form.

Example:

```
<cfform format="flash">
 <cfinput type="dateField"
   name="startDate" label="Start
   Date:">
</cfform>
```

Of course, Flash forms may not be for everyone or for every form in your application. For instance, your application may not need the richness of Flash, or you may be unable to run Flash in your corporate environment. You may also need to use non-Flash controls, or need complete control of the whole form. This is where you would use the XML Forms feature.

If you want to take Flash forms to the next level and be able extend and customize the forms further than Flash forms allow, check out Macromedia Flex (http://

www.macromedia.com/devnet/flex/).

Extending Forms with XML Forms

Since we introduced the cfform tag, there have been enhancement requests to improve it or complaints about the way we implemented it. With the introduction of the XML forms, these requests and complaints are no longer an issue. While Flash forms look and work great, XML forms give you the power to extend form functionality and layouts as you see fit, especially as you dig a little deeper and see what they have to offer.

We started to mull over the idea of XML forms years ago during the ColdFusion 6 development cycle. Conveniently, at the same time, the W3C was creating the XForms specification, which was finalized in the early days of ColdFusion MX 7 development. The XForms schema is designed as a way to describe a form in XML – including the model, elements, and validation – and with a way to extend the schema for custom implementations.

The idea behind the XML Forms in ColdFusion is similar to the Flash forms – to provide a way for you to define the form elements and layout manager hints. However, instead of having one predefined set of rules about the layout managers, you have endless possibilities.

The main benefit of XML Forms is the separation of presentation and logic. You can apply presentation "skins" to the forms in your application. As the developer, you define the fields and logic to manage the form, including default values, validation, and submitting to a database. Meanwhile, those in charge of the look and feel can pick the skin they want to apply and change it as much as they like. Six months from now, for instance, they could decide to change the look of their site completely. After making a quick change to the skin attribute or the XSL skin file, the form would have a new look and feel without the risk of accidentally changing the form logic.

If you want to write applications that change appearance for each user, XML forms give you an easy way to create a skin for each customer, reusing most of the same XSL code with only a few modifications – usually only to modify the CSS your form skin uses. In short, you can create a customized look for each application user.

The XML Forms use XSL files to skin your form. These style sheets give you complete control over how a browser displays your form. Besides, you were looking for an excuse to learn XML and XSL, right?

For instance, you could write a skin that generates the layout or the rendered HTML in browser-specific code. Doing this allows you to define the form functionality once and then use a variable, such as *CGI.USER_AGENT*, to switch between skins using browser-specific controls: a Firefox skin that uses CSS to render the layout, an Internet Explorer skin that uses tables, or a WAP skin to render the same form for those browsing your site from their phone.

I have found over the years that all my forms look the same. No matter what I do, my forms repeat the same HTML over and over again. With a custom XML skin, however, I can spend a little time creating my corporate skin, complete with the same table I always end up creating, a company logo, and any Section 508-compliant HTML requirements. By doing this, each form that my team or I create will have the same consistent look and feel. This brings "engineer art" to a whole new level!

To avoid confusion, many of the XForm examples available on the Internet require a browser plug-in. These are client-side implementations of XForms, where you must embed XForm XML elements inside of your XHTML document. With these examples the browser plug-in must then determine how to display the different elements in the browser. However, you still must use regular HTML to define the layout of form elements – one problem that ColdFusion MX 7 eliminates.

With ColdFusion, we implemented a server-side version that uses the XForms schema definitions for form elements, binding, and the model – with custom extensions instead of recreating the wheel and creating custom XML schema that defines form elements. However, instead of relying on a plug-in to render the form, ColdFusion MX 7 uses standard XSL scripts. If you wanted to use an XForms plug-in, you could use an XSL script to output plug-in friendly XForm XHTML and use one of the many XForm plug-ins to do the work on the client.

By letting the server-side scripts decide this, you can change the element definition on a skin-by-skin basis. For instance, perhaps you don't want to use a regular HTML *textarea* control. Instead, you would prefer to use a rich *textarea* control automatically, such as the Ektron text editor. Using a skin, you can automatically output the ActiveX code for the text editor every time a *textarea* form element is used.

With Flash forms we introduced new controls for the cfgrid and cftree tags; we have also done the same for XML Forms. In the XML forms, you can use the Java applet or Flash version of these controls, but you can also use the new "XML" format. This way, the XML describes the tree or grid with the tag attributes and generates the data in the XML form definition. In short, with this you can render controls as you like with your XSL scripts. For instance, you might want to output a DHTML grid instead, or perhaps you load an ActiveX control while still using the same simple cfgrid or cftree syntax in ColdFusion. As an example, I have used this XML format to turn the cftree into a DHTML menu.

Watch for additional articles in MXDJ about how to write your own skins. One easy way to learn about skins is to take one of the sample skins we shipped with ColdFusion MX 7. Save it as a new filename. At this point you now have a new skin. Edit the HTML that the skin outputs, the rules it includes (if you'd like to add support for some custom widget), or the CSS file that is associated with the skin. You have customized your first skin!

The Wait Is Over

Many of you at Macromedia MAX 2003 saw the early work we did on the cfform tag. Many of you wanted us to ship it then. Trust me, it was worth the wait to allow us to do the job right and polish up the form features. Thank you for your patience and feedback. I hope you find the new cfform tag to be a much more powerful and enjoyable experience to use.

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

Rolling Your Own Event Gateway

Writing and using event gateways in CFMX 7

by tom jordahl & jim schley

Event gateways are an exciting, new feature in Macromedia ColdFusion MX 7 that arose from one simple idea: that there are many applications out there that aren't part of the Web and don't communicate through the HTTP protocol.

These applications are on all types of devices. They run the gamut from the ubiquitous instant messaging clients to SMS on mobile phones to new things that haven't even been invented yet. ColdFusion does a great job powering applications that run on the Web –why not power non-Web applications too? Not only should it be possible for ColdFusion developers to write applications for non-Web applications and devices, but it should be easier to write them in ColdFusion than any other way.

ColdFusion MX 7 ships with several exciting types of event gateways that make it easy for you to get off the ground running with things like SMS. The server contains a simple Java API so that you can write your event gateway to connect to just about anything you want. With this extensibility, third-party software vendors can easily provide event gateways so that CFMX can talk to their non-CFMX applications.

Requirements

To complete this tutorial you will need to install the following software and files: ColdFusion MX 7 (http://www.macromedia.com/cfusion/tdrc/index.cfm?product=coldfusion&promoid=devcenter_tutorial_product_090903)

This article was originally published on the Macromedia Developer Center http://macromedia.com/devnet/.

Overview of Event Gateways in ColdFusion 7

ColdFusion 7 contains a new subsystem to support event gateways. The gateway types are Java classes that implement an application programming interface (API) provided by ColdFusion. Figure 1 shows the process for event gateway communication. The gateways communicate with the CF Event Gateway Services subsystem through CFEvent objects. The subsystem, in turn, queues the CFEvent object requests and passes them to the ColdFusion runtime engine for processing, as input to ColdFusion components (Listener CFCs). The Listener CFC might return output to the Event Gateway Services subsystem and then back to the event gateway.

Among the event gateways provided with ColdFusion MX 7 are: SMS for mobile text messaging; XMPP for open-standard instant messenger networks such as Jabber; Lotus Sametime for enterprise instant messenger communications; and asynchronous CFML for sending requests from your CFML to a CFC for processing in a separate thread. ColdFusion 7 also provides some sample event gateways types (with source code), including JMS for messaging applications that support the J2EE standard; TCP/IP Socket for use with a telnet client to interact with your applications; and Directory Watcher for watching a file system directory and to run your CFC when a user or application creates, edits, or deletes a file in that directory.

You create gateway instances from a gateway type. Instances correspond to individual copies of a gateway that are running. This is an object that is started/stopped (through the Administrator). Each gateway instance specifies a CFC to handle incoming messages. You can have more than one instance of an event gateway type, and each instance will have its own configuration. For example, you can have multiple instances of a given gateway type, each with different logins, phone numbers, buddy names, directories to watch, and so forth.

New Application Development Paradigm

Getting into the mindset of writing event gateway applications requires a change in thinking from writing traditional Web applications. The paradigm is different because you're no longer tied to the request/response nature of HTTP. Event gateways are asynchronous and can receive and send messages without depending on each other.

There are two basic types of event gateway applications: initiator applications and responder applications:

1. ***An initiator application*** is a ColdFusion application that generates an event message from CFML code and sends the message using a configured event gateway instance. An example of an initiator application is an e-commerce application that sends an SMS notification when a user's order has shipped. The application uses the SMS gateway to send a message, responding to some logic in the CFML code. You can enhance any CFML application to be the initiator. Use the *sendGatewayMessage* function to send outgoing messages, such as SMS text messages through an event gateway. In this case, you use the function just like the CFMAIL tag.
2. ***A responder application*** is a ColdFusion application that receives an event message that comes from

CF Event Structure

table 1

gatewayID	Instance ID
data	ColdFusion structure containing message data
originatorId	Identifier of event sender
gatewayType	Type of gateway (SMs, IM, etc.)

external source through the listening event gateway instance. The event gateway service forwards that event to the listener CFC, and the CFC method returns a response back to the event gateway instance. An example of a responder application is an IM bot that responds to questions. Responder applications are written in CFML and use a CFC as the listener for a gateway. The CFC listens for an event from a given gateway instance and processes the event structure passed to it to return a response. The event structure contains the message, along with some detail about its origin (see Table 1).

Writing an Event Gateway Application

The best way to learn about event gateway applications is to dive into the examples that ship with ColdFusion MX 7. Look for these examples under the "gateway" subdirectory in your install directory.

The DirectoryWatcherGateway event gateway comes preconfigured on ColdFusion MX 7 as an example to help you build your first CFML gateway application. The Java source code is also available for you to learn how to write your own Java gateway.

This event gateway watches a directory for changes. Based on your configuration file, which specifies the directory you want to watch and file system events you want to watch for (such as file creation, deletion, and changes), you can also send event objects to your CFC with the information when any event occurs.

In the following example, a CFC processes events using the Directory WatcherGateway event gateway:

```
<cffunction name="onAdd" output="no">
  <cfargument name="CFEvent"
    type="struct">
  <!--- get event data --->
  <cfset data=CFEvent.data>
  <!--- log a message --->
  <cflog file="watch"
    text="a file was #data.type#
     and " &
         "the name was #data.file
           name#">
  <!--- watcher will ignore outgoing
    messages --->
</cffunction>
```

The code is pretty simple – the function takes the event object as an argument and treats it like a simple CFML *struct* datatype. The *data* key of the struct contains a message that describes the file system event. The two entries in this struct are as follows:

- ***filename*** of the file that changed
- ***type*** of change that happened: add, delete, or change

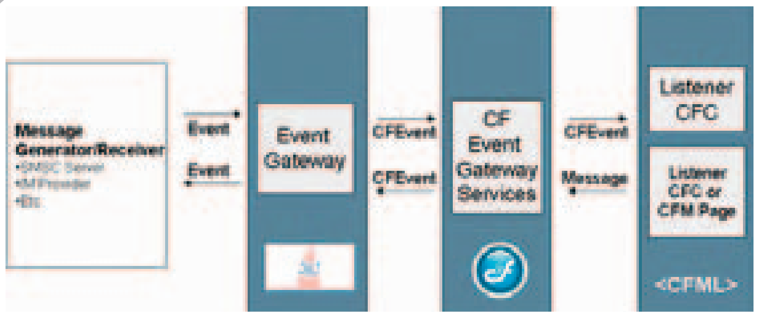
The event gateway also contains an entry, *lastModified*, which specifies the time an added or changed file was last modified.

Detailed Look at the ColdFusion Event Structure

Understanding what an event object contains and passes to your CFC as an argument through the event gateway is essential to writing your application. The event object is a Java object that maps to a CFML structure. Each event gateway type puts different data into an event object, so it's important to look at the gateway author's documentation to know what to expect. The CFEvent object includes whatever it is that the event gateway is trying communicate to your application. The event gateways provided with ColdFusion are well documented. The SMS event gateway event data structure contains the following information:

- ***shortMessage*** is the text message data
- ***sourceAddress*** is the phone number of the message sender
- ***destAddress*** is the phone number of the message recipient

figure 1



The SMS event gateway event data contains additional fields that you may fill, depending on the network carrier. You can read more about the SMS event gateway in the detailed chapter from ColdFusion MX Developer's Guide (http://livedocs.macromedia.com/coldfusion/7/htmldocs/wwhelp/wwhimpl/common/html/wwhelp.htm?context=ColdFusion_Documentation&file=00001666.htm).

The instant messenger (IM) event gateway event data structure contains the following information:

- ***message*** is the instant message text
- ***sender*** is the user ID of the message sender
- ***recipient*** is the user ID of the message recipient
- ***timestamp*** includes date/time data when the message was sent

You can read more about the IM event gateway in the chapter from ColdFusion MX Developer's Guide (http://livedocs.macromedia.com/coldfusion/7/htmldocs/wwhelp/wwhimpl/common/html/wwhelp.htm?context=ColdFusion_Documentation&file=00001651.htm)

In the event object, use the *gatewayType* key to identify the event gateway type sending the message to the CFC. This is useful for a CFC that handles input from multiple event gateway types. For instance, you might have an application that commu-

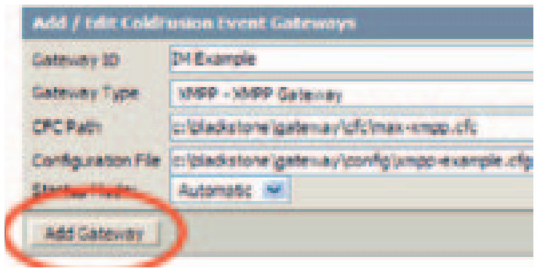
"ColdFusion MX 7 is extensible. Anyone can write an event gateway using the ColdFusion API" nicates with mobile phones using SMS and Lotus Sametime IM clients. Because there are two different event gateway types sending input to your application, each with its own version of the event object, it's best to evaluate the value of the *gatewayType* key before processing the incoming data.

The sendGatewayMessage Function
Use the new *sendGatewayMessage* CFML function to send event objects to an event gateway directly from your CFML code—from any CFC or CFML page.
Use the following code syntax:

```
endGatewayMessage (gwid, data)
```

- ***Argument 1:*** Gateway ID – Name of the gateway instance configured in the ColdFusion Administrator
- ***Argument 2:*** Data – CFML struct with name/value pairs that corresponds to an event object for the gateway

figure 2

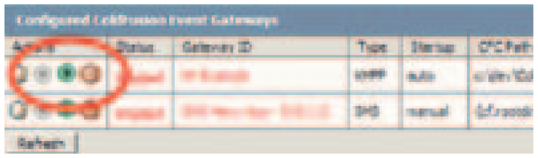


Add / Edit ColdFusion Event Gateways

Gateway ID	IM Example
Gateway Type	XMPP - XMPP Gateway
CFC Path	c:\blackstone\gateway\cfc\max-xmpp.cfc
Configuration File	c:\blackstone\gateway\config\xmpp-example.cfg
Startup Mode	Automatic

Add Gateway

figure 3



Configured ColdFusion Event Gateways

Actions	Status	Gateway ID	Type	Startup	CFC Path
[illegible]	[illegible]	IM Example	XMPP	auto	[illegible]
[illegible]	[illegible]	[illegible]	SMS	manual	[illegible]

Refresh

When creating a structure for an outgoing IM event object, use the following syntax:

```
<cfset data = structNew()>
<cfset data.message = “hello”>
<cfset data.buddyId = “cfguy”>
```

When sending the message through the XMPP event gateway, use the following syntax:

```
<cfscript>
sendGatewayMessage(“XMPP
  server1”,data);
</cfscript>
```

What to Specify in Outgoing Data

Just as each gateway type has different event object data to pass to the CFC, each gateway type requires different data in the event object that it will process for outgoing messages. The event gateway's documentation specifies what it requires for data. Many gateways have a few required data items and many optional items. The design is flexible and allows for an unlimited number of data items.
For example, the IM event gateways requires the *buddyID* and *message* data items, but optionally accepts the command data item, with values of *accept, decline,* or *submit*. If the value is *accept* or *decline* (for instance, to respond to a request to add the gateway user to a sender's buddy list), you can optionally set the *reason* data item, which specifies explanation text.

The SMS event gateway has a long list of optional data items that an outgoing event object can contain. The optional data items correspond to specific options in the SMPP protocol, which communicates SMS messages on mobile carrier networks. The ColdFusion team did our best to include every single option in the SMPP specification, even though different carriers implement the specification differently. For instance, you can optionally set a *registeredDelivery* data item in your outgoing event object to request delivery receipt for your SMS.

The getGatewayHelpers Function

Use the *getGatewayHelper* function to provide additional functionality to ColdFusion applications according to the event gateway's purpose. The IM event gateways, for example, provide buddy-list management functionality using an event gateway helper.

Use the following code syntax:

```
getGatewayHelper (gwid)
```

- ***Argument:*** Gateway ID – Name of the event gateway instance configured in the ColdFusion Administrator

This function returns an object on which helper functions can be called. For the IM event gateways, this returned object has *addBuddy (), remveBuddy (),* and *getBuddyList ()* functions available for buddy-list management. These are just some of the functions that the IM gateway helper makes available.

The following code example adds a new buddy with the user ID, cfuser2, to the server user's buddy list:

```
<cfscript>
imhelper=getGatewayHelper(“XMPP_
  server1”);
imhelper.addBuddy(“cfuser2”);
</cfscript>
```

Instant Message Example

Getting started with an instant messaging application requires first setting up an event gateway instance in the ColdFusion Administrator. To set up the instance, specify a configuration file with the connection details for the IM network.

The following is an example of what your configuration file for the XMPP event gateway might look like:

```
# XMPP Instant Message Configuration
  file
userid=yourid@jabber.org
password=yourpassword

# everything else is OK to default
serverip=jabber.org
serverport=5222
retries=5
retryinterval=10
onInComingMessageFunction=
  onIncomingMessage
onAddBuddyRequestFunction=
  onAddBuddyRequest
onAddBuddyResponseFunction=
  onAddBuddyResponse
onBuddyStatusFunction=onBuddyStatus
onIMServerMessageFunction=
  onIMServerMessage
```

You also specify the listener CFC. The following is an example of a simple CFC that echoes an instant message sent to it with the original text:

```
<cfcomponent>
<cffunction name=”onIncomingMessage”
   output=”no”>
<cfargument name=”CFEvent”
   type=”struct” required=”yes”>
  <!--- Get the message --->
  <cfset data=cfevent.DATA>
  <cfset message=”#data.message#”>
  <!--- where did it come from? --->
  <cfset orig=”#CFEvent.
    originatorID#”>
  <!--- make a struct to return --->
```

```
    <cfset retValue = structNew()>
    <cfset retValue.message
         = "Your message: " & message>
    <cfset retValue.buddyId = orig>
    <!--- send the return message
      back --->
    <cfreturn retValue>
  </cffunction>
  </cfcomponent>
```

Once you have your configuration file and CFC set up (see Figure 2), you're ready to create the event gateway instance in the ColdFusion Administrator. After you start the event gateway instance (see Figure 3), you're ready to start using the IM application.

Rolling Your Own Event Gateway

If you need to create an event gateway for an application, protocol, or purpose that you can't accomplish with the built-in, sample, or third-party event gateway types, remember that ColdFusion MX 7 is extensible. Anyone can write an event gateway using the ColdFusion API.

"ColdFusion is no longer confined to the Web"

You must know Java to write an event gateway, but you can start with the documented sample code. Also check out of the chapter on *ColdFusion MX Developer's Guide: Creating Custom Event Gateways* in the documentation; it walks you through all the necessary details to create and run your own event gateway type.

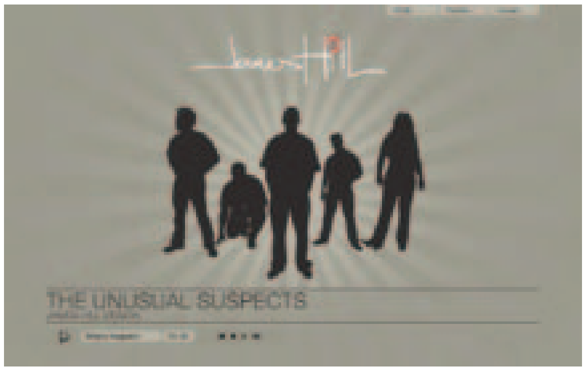
Event gateway applications are easy to write once you understand the programming model available to you. Once you realize that your ColdFusion applications do not depend on an HTTP request/response paradigm, you'll see how event gateways open up a whole new world of possibilities for you. ColdFusion is no longer confined to the Web.

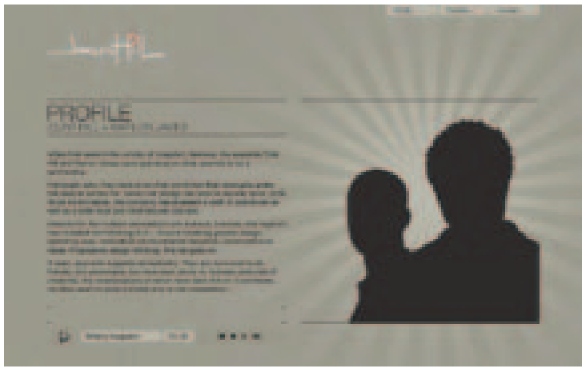
Tom Jordahl is the principal engineer and technical lead for the ColdFusion Blackstone project. As part of the original ColdFusion team, he has implemented a wide variety of tags, functions, and features in both the original (C++) and MX (Java) releases of ColdFusion. Tom thinks CFMX 7 is going to rock, particularly the event gateway feature. He is one of the primary implementers of the Apache Axis SOAP engine and is the Macromedia representative on the W3C WSDL 2.0 Working Group. tjordahl@macromedia.com

Jim Schley has seven years' experience building Web applications and has been programming with ColdFusion since version 2.0. He has built e-commerce sites and enterprise Web applications for such customers as the US Mint, the FAA, Pfizer, Pharmacia, The BOC Group, and Schering-Plough. For Macromedia, he works as Principal QA Engineer for ColdFusion, concentrating most of his efforts on product performance. jschley@macromedia.com

The Unusual Suspects

James Hill Design Limited is a seven-year-old design agency borne out of equal parts inspiration and frustration – frustration at working for other people whose vision didn't match ours, and inspiration to do our own thing and make our own mark on this already marked-up world. However, due to the shifting nature of the business as well as the shifting locations of the principals (Jamaica, Canada, or New York depending on the mood), an Internet presence became essential. In order to keep up with the global way of doing things, this Web site was done in collaboration with the Canadian firm Sumo who did the lion's share of the programming, posting, and a good bit of the designing from the original print and multimedia materials that we had already created for our corporate identity. As James Hill becomes more and more of an entity rather than a place, our Web site has become our most crucial marketing and communication tool. It has also freed us from country boundaries. Now clients see us as a world class design firm first, and a Jamaican company second. The Web site has also opened us up to a new kind of client, as boundary-free as we are, with only one concern: getting damn good creative. www.jameshilldesign.com





HostMySite.com

TM

Built for ColdFusion Pros by ColdFusion Pros

plans from **$8.95** / mo.

FREE Domain Name*

FREE Setup

FREE 2 Months

- 24 / 7 / 365 Phone Support
- 99.9+% Uptime
- Macromedia Alliance Partner
- "Full Control" Panel
- CFMX 6.1 or CF 5.0
- SQL Server 2000 or 7.0
- Custom Tags Welcome

Visit www.HostMySite.com/mxdj for:

2 Months Free

and FREE Setup on Any Hosting Plan*

"When it comes to ColdFusion hosting, HostMySite.com rules them all!"

James Kennedy
mbateam.com

*offer applies to any annual shared hosting plan

call today! **877•248•HOST** (4678)

macromedia

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

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

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