SYS-CON

GREAT MAPPING MASHUPS USING ADOBE FLEX

FORMERLY MX DEVELOPER'S JOURNAL

Web Developer's & Designer's Journal

WWW.WEBDDJ.COM

VOLUME 4 ISSUE 9 **2006**

BUILDING ENGAGING APPLICATIONS AND CONTENT WITH ADOBE TECHNOLOGIES

The Web Is Exciting Again

There has never been a better time to be building Web applications than now

Plus:

- Fireworks Text Optimizing
- Do You Speak Web?
- Flex Tree Control

Presorted
Standard
US Postage
PAID
St. Croix Press

Michael Uman and Luis Blanco,
INTERspectacular

Adobe® Creative Suite 2

Say hello to the next generation.

It's found in the world's most inspirational places. It's pushing the boundaries of art direction and design. Adobe® Creative Suite 2. Today's most talented designers and art directors are working smarter and faster because of it. Just ask the masterminds behind INTERspectacular—they rely on Adobe CS2 to make their ideas tangible reality. See what Adobe Creative Suite 2 can do for you. It's everything but the idea. Better by Adobe.™

adobe.com/go/designer

Adobe®

Freedom of Expression

Automatically Convert User-generated Video Into the Leading Streaming Media Format Using VitalStream Transcoding

Introducing VitalStream Transcoding, an online solution enabling businesses to easily translate user-generated video into Flash™ video, the most widely available streaming media format. With transcoding you can accept most digital video formats from customers and automatically convert those files into Flash video.

To learn more about VitalStream transcoding services, visit www.vitalstream.com/go/transcoding or call 800 254 7554

VitalStream®
PROFESSIONAL SERVICES

Web Developer's & Designer's Journal
September 2006

```
function optimizeRIA() {
        if (omniture.actionsource == true) {
                businessSuccess();
        } else {
                if (javascript.futile == true) {
                        businessFail();
                }
        }
}
```

businessSuccess();

OMNITURE®
ActionSource™

A new way to measure the impact of your **Rich Internet Applications.**
No JavaScript required. **No** hassles. **More** accurate. **More** success.

WANT TO LEARN MORE ABOUT OMNITURE ACTIONSOURCE™?

→ **www.omniture.com/actionsource**

1.877.722.7088

OMNITURE®

Web Developer's & Designer's Journal

Group Publisher Roger Strukhoff
Art Director Louis F. Cuffari

Editorial Board
Aral Balkan
Erik Bianchi
Craig Goodman
Jim Phelan
Andrew Phelps
Darron J. Schall
Stephanie Sullivan
Jeff Tapper
Jesse Randall Warden

Editorial
Editor
Nancy Valentine, 201 802-3044
nancy@sys-con.com

Associate Editor
Lauren Genovesi, 201 802-3041
laureng@sys-con.com

To submit a proposal for an article, go to
http://grids.sys-con.com/proposal.

Subscriptions
E-mail: subscribe@sys-con.com
U.S. Toll Free: 888 303-5282
International: 201 802-3012
Fax: 201 782-9600
Cover Price U.S. $5.99
U.S. $29.99 (12 issues/1 year)
Canada/Mexico: $49.99/year
International: $59.99/year
Credit Card, U.S. Banks or Money Orders
Back Issues: $12/each

Editorial and Advertising Offices
Postmaster: Send all address changes to:
SYS-CON Media
577 Chestnut Ridge Rd.
Woodcliff Lake, NJ 07677

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ

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

Promotional Reprints
Megan Mussa, 201 802-3024
megan@sys-con.com

Web Developer's & Designer's Journal (ISSN#1546-2242) is published monthly (12 times a year) by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

The Perfect Storm of Web 2.0 Disruption?

by Jeremy Geelan

The winds of change in the Web world have reached hurricane force right now, and nowhere are they blowing more fiercely than around that epicenter of weather activity that's been labeled "Web 2.0." There, a perfect storm is brewing.

I choose the term "perfect storm" advisedly since no other phrase that I am aware of encapsulates as succinctly the crucial insight that it's a confluence of simultaneous yet quite disparate events and changes that, in my view, is driving the current turmoil. Few technology - and fewer still business - commentators seems to have lifted up their heads for long enough recently to realize that the reason they need to take "Web 2.0" very seriously indeed is that its marketeering connotations belie the reality that we are experiencing the simultaneous occurrence of events which, taken individually, would be far less powerful than the result of their chance combination. Together, however, they have developed an awe-inspiring power.

Such occurrences are, by their very nature, rare. But that doesn't mean they aren't real and that they don't happen. Ask anyone in the 1,000-mile radius of the epicenter of the Asian tsumani of 2004.

Paul Graham, well-known essayist and hardly a superficial "hypester," did an interview with TechCrunch recently in which he said as follows:

"To me "Web 2.0" translates to "Web." And Web technologies don't appeal only to a small niche. Web-based email services have hundreds of millions of users. The network (in the broader sense of the Internet plus the phone networks) pervades everything now.

We're pretty open about what we think makes a technology stick. We print it on T-Shirts: 'Make something people want.' If you had to reduce the recipe for a successful startup to four words, those would probably be the four."

Graham has also written, in a November 2005 essay:

"Does 'Web 2.0' mean anything more than the name of a conference yet? I don't like to admit it, but it's starting to. When people say 'Web 2.0' now, I have some idea what they mean. And the fact that I both despise the phrase and understand it is the surest proof that it has started to mean something."

Tim Berners-Lee on the other hand, has been dismissing Web 2.0 as "a piece of jargon" and pointing out - most recently in a podcast he did with IBM's developerWorks - that Web 2.0 relies on Web 1.0 technologies such as the DOM, HTML, http, SVG, web standards, and JavaScript. Which, with the deepest respect for one of the great geniuses of the late 20th century, suggests that TBL is not wholly up to speed yet with the 21st: to say that "Web 2.0" is meaningless because it relies on pre-existing technologies is surely a little like saying that "supersonic flight" is meaningless because the Wright Brothers got there first.

Berners-Lee argues that "Web 1.0" was already totally about connecting people. "It was an interactive space," he notes. "The idea of the Web as interaction between people is really what the Web is. That was what it was designed to be - as a collaborative space where people can intreract."

Now one doesn't take issue lightly with the Father of the World Wide Web, but my concern is that, in his perhaps understandable desire to do whatever he can to stop what he possibly perceives as a New Bubble, Berners-Lee is inadvertently falling into the (for him, of all people) surprising role of what Virginia Postrel calls a "stasist" - as in one who favors the static over the dynamic.

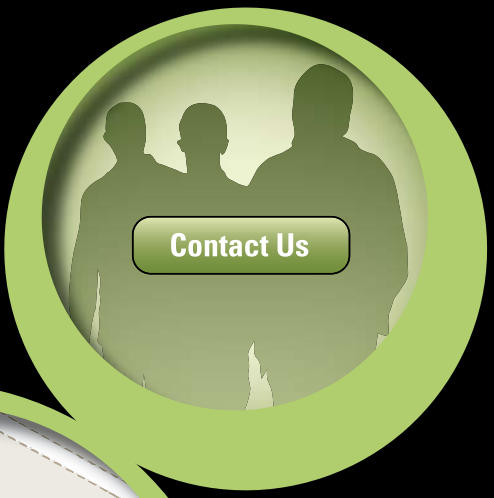
Because the real essence of "Web 2.0" - pace Berners-Lee - is not its technology

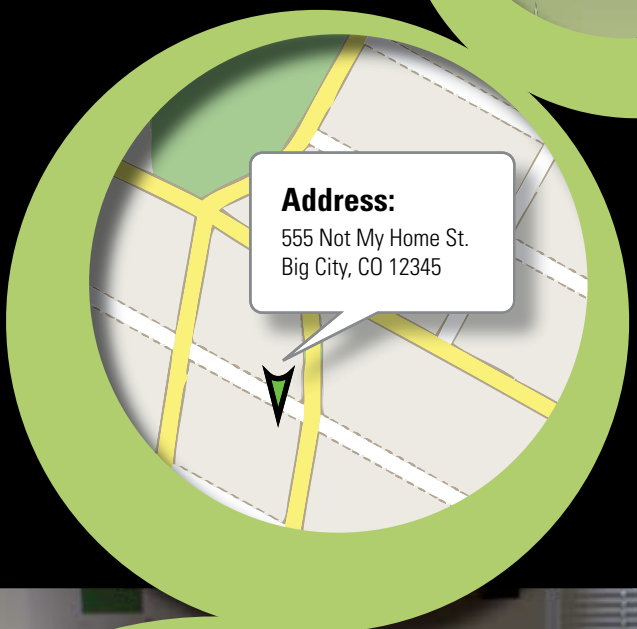
– continued on page 50

Jeremy Geelan is Sr. Vice-President, Editorial & Events, of SYS-CON Media. He is Conference Chair of the AJAXWorld Conference & Expo series and of the "Real-World Flex" One-Day Seminar series. From 2000-6, as first editorial director and then group publisher of SYS-CON Media, he was responsible for the development of all new titles and i-Technology portals for the firm, and regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas.

Scary Question.

Exactly who is developing your next app?





Your App Starts Here.

We are the leaders in RIA development services.

INCREDIBLE APPLICATIONS

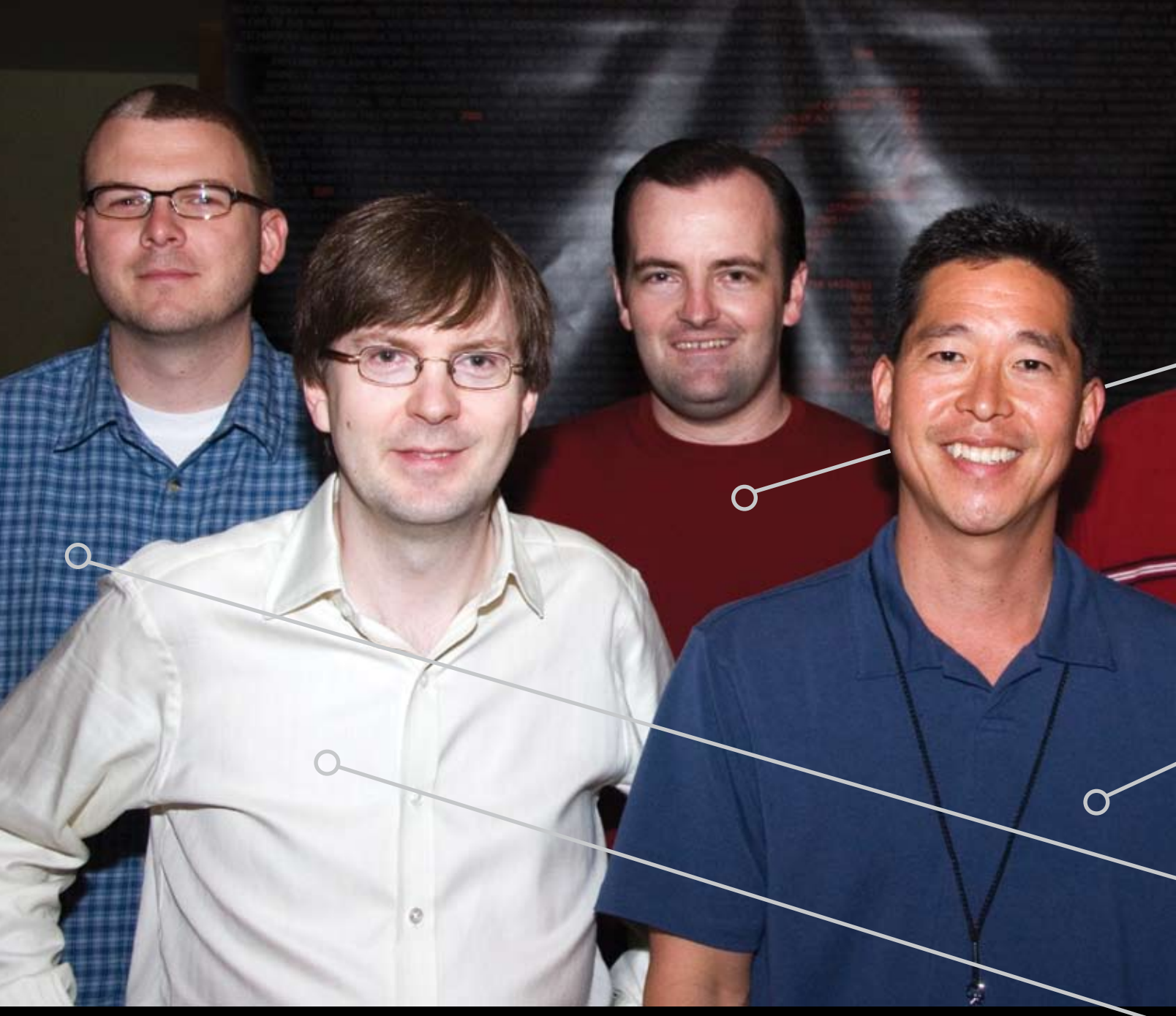
PASSIONATE USERS

PROVEN SUCCESS

Unlock your potential
with the help of industry leaders in
Rich Internet Application development.
10 Years. 1400+ Customers.
Your app starts here.

CYNERGYSYSTEMS.COM

Adobe | Solution PARTNER



Letter to the Editors

Identity Crisis?

I enjoyed reading the feature in the August issue about "10 Glorious Years of Flash" [WebDDJ vol. 4 iss. 8], but was just wondering two things.

1. You included biographies of all the people in the photo on p. 10 of the team behind Flash…but you didn't specify who was who!! Since I live overseas and don't attend events like MAX, I don't have any way of knowing who is who, except for Kevin Lynch whose face of course everyone recognizes. Could I ask that WebDDJ maybe reprints the photo, but this time with the bios linked to the faces?
2. I wondered why Jonathan Gay, the "Father" of Flash, was not one of those in the photo. Is he still with Adobe?

John Gibbons
Bargememon, Nice
France

Mike Downey

- Senior product manager, Flash
- Passionately involved with web technologies since the late 90s.
- Joined Macromedia in 2000
- Experienced instructor, presenter, and developer.
- Has met with thousands of designers, developers, and technologists through numerous live and online events.
- Has also been a featured presenter at conferences such as FlashForward, FiTC, Macworld, Spark, WebDU, Oracle OpenWorld, JavaOne, Macromedia MAX and several others.
- Maintains a weblog devoted to Flash and other Internet technologies at http://weblogs.macromedia.com/md.

Robert Tatsumi

- One of the original Flash engineers – worked with Jon Gay
- Co-Creator (with Jonathan Gay) of SmartSketch (later FutureSplash Animator) the precursor of Flash.

Justin Everett-church

- Sr. Product Manager for Flash Designer/Developer relations
- Co-author of four books on game development in Flash.
- A frequent speaker at conferences on the topics of Flash for mobile and devices and Flash outside the browser.
- Formerly Rich Media Evangelist for Yahoo!
- First saw FutureSplash on MSN as a spinning globe animation in 1996.

Slavik Lozben

- Flash engineer #3
- Principal engineer on the Flash Media Server team.
- Involved in the design and implementation of many of the features in FMS.
- Joined Macromedia in January of 1995.
- Originally worked on the Director product line, but when Macromedia bought what is now known as Flash, saw a tremendous opportunity and joined Robert Tatsumi and Jon Gay, the creators of Flash.
- Has been on the Flash team ever since.

Kevin Lynch

- SVP and Chief Software Architect, Platform Business Unit
- Named one of CRN's "Top 25 Innovators" in 2003 and honored as one of the "First Annual Web Innovators" by CNET in 1998.
- Joined Adobe through the company's 2005 acquisition of Macromedia, Inc., where he served as chief software architect and president of product development.
- Responsible for the company's ubiquitous Portable Document Format (PDF), Adobe® Reader®, and Macromedia® Flash® Player, as well as alignment of Adobe's servers and tools with the company's technology platform.
- Also oversees Adobe's developer relations program.

Mike Chambers

- Senior Product Manager for Developer Relations
- Has spent the last 8 years building applications that target the Flash runtime.
- Has worked during those 8 years with numerous technologies including Flash, Generator, Central, Flex, and Ajax.
- Currently in charge of developer relations for Apollo, a project which aims to bring the power of Rich Internet Applications to the desktop.
- Has co-authored a number of books and articles, spoken at numerous conferences, and maintains a Flash Platform development weblog at http://weblogs.macromedia.com/mesh/.

Aral Balkan

Aral Balkan is founder and managing director of Ariaware, a London-based company offering products like Ariaware Optimizer and the open-source Ariaware RIA Platform (ARP 2.0) for Flash developers. Ariaware also offers RIA development process and usability consulting and development services. Aral holds an MA in Film and Electronic Media, is a Macromedia Certified Instructor and is celebrating his 20th year as a programmer (he's only 27!). His passions include software architecture and Human-Computer Interaction - in other words, building solid, usable applications. He's co-author of "Flash MX Most Wanted Components" and "Flash 3D Cheats Most Wanted," as well as author and editor of numerous articles for Adobe Developer Center and Ultrashock.com.

Erik Bianchi

Erik Bianchi is a software engineer with more than five years of experience developing Flash-based RIAs and enterprise-wide applications for Fortune 50 and 500 companies. In his spare time he enjoys building Flash-based games, writing or tech editing Flash-related books, and when burned out on code, playing video games on his PC/console systems. You can get more info about Erik and his latest projects on his blog at www.erikbianchi.com.

Craig Goodman

Craig Goodman is the managing editor of Adobe's Developer Center. He and his team publish the tutorials and articles in the area. Craig joined Macromedia in 1995 and his past roles include managing web support and supervising product technical support for Macromedia Flash.

Jim Phelan

Jim Phelan is vice president of development for Stream57, a New York City based firm specializing in communication solution development for the enterprise. Jim's expertise in creating solutions for consolidation and collateralization of business communications has allowed his team to create applications for the management and delivery of live and on demand rich media content. Jim is a strong proponent of the Adobe Flash Platform and is a member of the editorial board of MX Developer's Journal.

Andrew Phelps

Andrew M. Phelps is in the Information Technology Department at the Rochester Institute of Technology in Rochester, NY (http://andysgi.rit.edu/).

Darron J. Schall

Darron J. Schall has been programming long before he could drive. In school he studied programming languages, ranging from Basic to Pascal to C++ and eventually moving into Java and C# throughout college. Somewhere in the middle he got hooked on Flash 5 and it's been a crazy love affair ever since. Darron is an independent consultant specializing in RIA development. He maintains a Flash Platform related weblog (www.darronschall.com) and is an active voice in the Flash and Flex communities.

Stephanie Sullivan

Stephanie Sullivan is a Web developer, partner at CommunityMX (www.communitymx.com), owner of VioletSky Design (www.violetsky.net), and contributing author of Dreamweaver MX 2004 Magic.

Jeff Tapper

Jeff Tapper, co-founder of Tapper, Nimer and Associates, has been developing Internet-based applications since 1995, for a myriad of clients including Toys R Us, IBM, Allaire, Dow Jones, American Express, M&T Bank, Verizon, Allied Office Supplies, and many others. As an Instructor, he is certified to teach all of Adobe's courses on Flex, ColdFusion and Flash development. He has worked as author and technical editor for several books on technologies including Flex, Flash and ColdFusion, such as "Object Oriented Programming with ActionScript 2.0", and "Flex 2 Training from the Source."

Jesse Randall Warden

Jesse R. Warden is a senior Flash developer at Surgical Information Systems, an operating room software company, where he currently uses Flash MX, Flash Remoting, .NET, and Oracle to create next-generation rich Internet applications for the OR. He contributed four chapters to the Flash Communication Server MX Bible and has written articles for various publications, including one for Macromedia for a DRK.

CF_UNDERGROUND VIII PRE-MAX EVENT

DATE: SUNDAY, OCTOBER 22, 2006

LOCATION: THE BEACH, LAS VEGAS

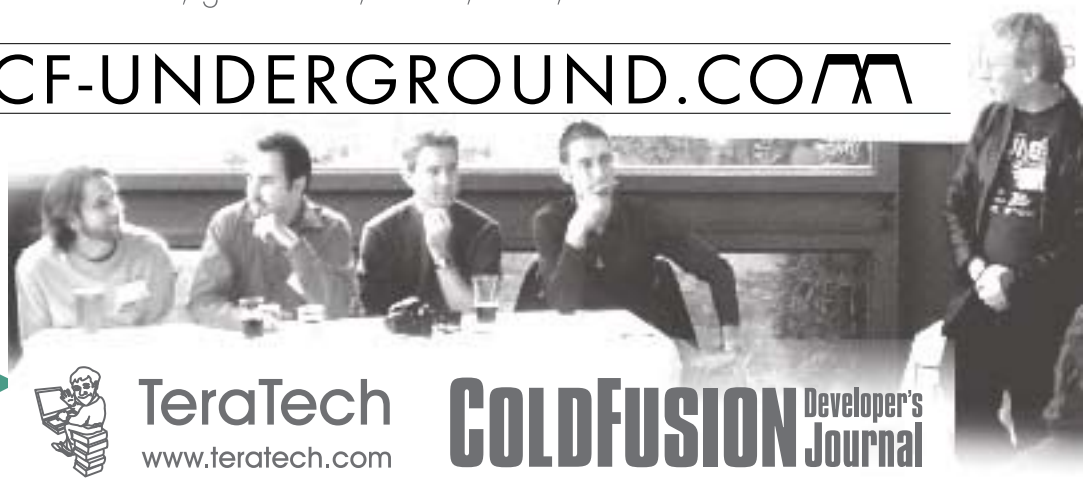
365 Convention Center Drive, Las Vegas, Nevada

Speakers include:
Simon Horwith
Michael Smith
Glenda Vigoreaux
and more...

Meet with old friends and new before the MAX event. A full day of networking, ColdFusion, great talks, drinks, food, and fun.

WWW.CF-UNDERGROUND.COM

only $99



TeraTech
www.teratech.com

ColdFusion Developer's Journal

Un Momento, Por Favor

A brief glimpse into software design patterns and their usefulness

by James O'Reilly

When developing applications, it's very common to find situations where you'd like to remember the state of an application, either in whole or in part, and later return your application to a previous state.

Let's take a look at a real-world example…

Your footsteps echo down the unmarked path. Gravel shuffles everywhere as you slow and strafe around the corner of a generic concrete bunker. You reach for the double- barreled shotgun but it's too late. A loud bang rips through the air but it's the soft thud as you hit the ground that confirms your worst fear, you're dead…again. A quick click of the mouse and poof…like magic you're all the way back to your last reached checkpoint.

If you're like me, you're all too familiar with this scenario. When you reach the checkpoint for the first time, the application quickly stores essential information about your character's current state. Your state might consist of things like health percentage, the existence and condition of body armor, a list of carried items and your checkpoint location. This information is bundled into a "game state" object and stored quietly behind the scenes. Later on, when you fail miserably to reach the next checkpoint, the game state object is loaded and the application is restored to a previously defined state.

In addition to the application state as a whole, you may find the need to record the state of smaller parts. We can also find examples of this in our favorite first-perso n shooter. For example, the state of a room you enter might be stored in a "room state" object when you leave the room. This object may store a list of items and their locations, existence of any damage, etc. This way, the guard you fragged will still be lying on the floor and the health pack you picked up will still be missing when you re-enter the room later. A "caretaker" object would be responsible for holding each of a level's room state objects in a randomly accessible list so that the application can retrieve the state of any room it needs to. By utilizing a room state object, when you leave a room the game is able to unload the overhead the room might consume with its textures and objects yet maintain the ability to easily recreate the room by loading it in its default state then modifying it with information found in a room state object.

While these examples are game-specific, the same concept can be seen in almost any application. In fact, the undo feature is the quintessential example of restoring an application to a previous state. Multiple undos are implemented by storing command objects in a stack. The last command added to the stack is the first to be retrieved by CTRL+Z, creating a last-in, first-out scenario.

From a mile-high view, the undo feature, room states, and checkpoint spawning are all basically the same thing. Therefore, it would be reasonable for us to assume that each of these scenarios can be implemented in a similar way if we were to describe their solutions from a similar distance.

Software Design Patterns

Software design patterns are practical solutions to the common software design problems we face when developing applications. While software design patterns were a concept first adapted from architectural design patterns in 1987 by Kent Beck and Ward Cunningham, they didn't really become popular until about seven years later. In 1994, the book Design Patterns: Elements of Reusable Object-Oriented Code by the "Gang of Four" (Gamma et al) was released and object-oriented programming practices were changed as design patterns became very popular.

Critics of design patterns argue that the abstract solution a design pattern provides often increases the amount of code needed to implement the solution. They also state that, by adding code not specific to the project, you go against the principles of "best practice" programming. Many advocates of design patterns, like me, recognize these cons but believe the pros outweigh them by far and do, in fact, promote "best practice" solutions in a way of their own. Yes, with design patterns you're implementing abstract solutions, but so what? After all, if it's considered

James O'Reilly lives on Long Island, New York where he has been developing Web-based applications for roughly 11 years. Currently, James focuses on Flash, ASP.NET, and SQL Server to create RIAs for industry-leading clients. james@jamesor.com.

"best practice" to design abstract classes, why can't solutions be abstract? To me, patterns end up providing a more scalable solution than I might have implemented had I chosen to write something specific to the project. Another major advantage is the ability to have a common language with other developers, especially ones I've never met.

Without design patterns, developers working in teams need to do more planning and collaboration to implement the same solution, since each developer's prior experience may have led them to different solutions to the same problem.

"Hey Mike, I think we need to take a snapshot of the game at each checkpoint so that we can restore the game to a point in time when the player dies."

"Yeah, you're right, John. I remember on the Acme project we wrote some classes that stored small pieces of information we later used to restore the application."

"I dunno, I didn't work on the Acme project so I'm not at all familiar with how they handled it. But what you describe sounds more like what we did on the Foo video player."

"No, not really. With the Foo player we had that hack Joe work on the project and we ended up changing it a bit afterwards. I was thinking more like…"

Anyway, you get the idea. Wouldn't something like the following be easier?

"Hey Mike, I think we should implement the Memento design pattern for the room states."

"Yeah, no doubt. I'll go ahead and write the memento class. Do you want to write the caretaker class?"

By implementing a design pattern, the developers can reuse the amount of time spent on this feature.

The Memento Pattern

As you've likely guessed from the blatant hints so far, I was leading up to discussing the Memento pattern. The Memento pattern is easy to understand, since it's frequently used and easily identifiable.

The Memento pattern is typically made up of three objects: the originator, the memento, and the caretaker.

- ***Originator*** – The object of focus, the state of which we want to save and restore. This object would have two public methods:
- ***getState():object*** – Returns a memento object to the caller, which will be the caretaker object
- ***restoreState(object):Boolean*** – Called by the caretaker passing in the memento object so that the originator can restore its state
- ***Memento*** – A container object that will encapsulate the essential information required to restore the originator
- ***Caretaker*** – An object that will store the memento object for the originator. It will provide the memento object to the originator when the originator needs to be restored.

In a nutshell, the Memento object is created by the originator object, and then stored in a caretaker object. The caretaker supplies the originator with a memento when the originator needs to restore its state.

Let's examine a simple version the Memento design pattern by creating a Flash SWF player you might typically see on the homepage of a site. The main SWF is a parent SWF that loads children SWFs, each of which play as animations and have some level of interactivity. In this example, the main SWF would be the caretaker, the currently loaded child SWF would be the originator and the memento would be an object created by the child and stored in the parent. The reason we need to store the memento in the caretaker and not the originator is that we'll unload the originator when we load the next SWF in the playlist. By using a caretaker, the memento can persist after the originator is gone.

For a simple illustration of the concept, I show the implementation below using more of a procedural method of programming. An object-oriented approach would be better to enforce encapsulation and prevent non-originator access to the memento content. On my Web site at http://www.jamesor.com/downloads/wddj/mementos.zip you can find the full source code to both procedural and object-oriented implementations.

In our parent movie, we create a list to hold our memento objects:

```
// list to hold memento objects
var aryMementos:Array = new Array(5);
```


SYS-CON **Media**

CEO
Fuat Kircaali, 201 802-3001
fuat@sys-con.com

President & COO
Carmen Gonzalez, 201 802-3021
carmen@sys-con.com

Sr. Vice-President, Editorial & Events
Jeremy Geelan, 201 802-3051
jeremy@sys-con.com

Advertising
Vice President, Sales & Marketing
Miles Silverman , 201 802-3029
miles@sys-con.com
Robyn Forma, 201 802-3022
robyn@sys-con.com

Advertising Sales Manager
Megan Mussa, 201 802-3023
megan@sys-con.com

Associate Sales Managers
Kerry Mealia, 201 802-3026
kerry@sys-con.com
Lauren Orsi, 201 802-3024
lauren@sys-con.com

Production
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 Directors
Abraham Addo, 201 802-3037
abraham@sys-con.com
Tami Beatty, 201 802-3038
tami@sys-con.com

Assistant Art Director
Mandy Eckman, 201 802-3032
mandy@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
Paula Zagari, 201 802-3042
paula@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

Customer Relations
Circulation Service Coordinator
Edna Earle Russell, 201 802 -3081
edna@sys-con.com

> "Without design patterns, developers working in teams need to do more **planning and collaboration to implement the same solution**"

```
var nLastIndex:Number = 0;
// when you change the child SWF,
// the first thing you do is save
// a memento of the currently loaded
// slide then changed to the new
// slide and if we have a memento
// for the new slide, then restore it
function changeSlide (i:Number) {
  // store memento of current slide
  aryMementos[nLastIndex] =
          child_mc.getCurrState();
  // load in the new child movie
  ... code to load child SWF i ...
  // pass child back it's memento
  if (aryMementos[i] != undefined) {
    child_mc.restoreState(
                    aryMementos[i]);
  }
  nLastSlideIndex = i;
}
```

On frame one in our each of our child movies, we include getter and setter like functions for creating and restoring mementos:

```
function getCurrState ():Object {
  // store all of the relevant data
  // into a memento object
  var memento:Object = new Object();
  memento.frame = _currentframe;
  memento.name = name_txt.text;
  memento.foo = "bar";
  ... whatever else you need ...
  return memento;
}
function restoreState (
            memento:Object):Void {
  // restore any saved data
  name_txt.text = memento.name;
  ... plus the remaining data ...
  gotoAndPlay(memento.frame);
}
```

Since we defined our memento object inside the child SWF, we can have unique memento definitions for each child we load. The caretaker doesn't care what's inside the memento objects; it just stores them in an array and returns them as objects when needed.

Occasional Connectivity

If you connect to the Internet with a laptop, you're probably familiar with being online only some of the time. You may lose your wireless signal going through a tunnel on the train or just have no access at all until you're back in the office plugged into a LAN line.

One way to improve a user's experience and meet their ever-increasing expectations of RIAs might be to implement the Memento pattern on a complex form. This way, if a connection is lost before the user gets to submit a complex form, you could create a memento object that contains all of the form field data on the screen. This memento object can be stored on the user's hard drive and later retrieved so it can be submitted the next time an Internet connection is available.

Using the SharedObject, we can easily store and retrieve a memento object.

```
// Get the flash cookie named
// "memento" or create one if
// it does not already exist
// and get that.
var my_so = SharedObject.getLocal(
                              "memento");

if (weRealizeThereIsNoConnection) {
  // store our memento object inside
  // the data object of our
  // SharedObject
  my_so.data.memento =
          oOriginator.getCurrState();
  my_so.flush();
} else {
  if (my_so.data.memento!=undefined){
    // restore our originator
    oOriginator.restoreState(
                  my_so.data.memento);
    // now that the data is restored
    // and a connection is available,
    // try sending the data again
    ...try submitting data again...
  }
}
```

The new MX CSS Menus2 for Dreamweaver

Create and customize CSS menus in Dreamweaver !

The new MX CSS Menus 2 from InterAKT is a Dreamweaver extension that offers you the choice to create your own CSS menu, implement your own ideas in a customized skin and perfectly blend your menu into your site's design. With visual step by step implementation and no CSS knowledge required this new version includes:

- CSS Skin Editor to create your own design
- Support for HTML and database driven sites
- Faster loading in the browser
- Vertical, tabbed, horizontal or expandable menu types
- 23 pre-built CSS skins
- PNG and PSD files included
- Animation effects on sub-levels
- New wizards and interfaces

Buy at only $99

Free trial available at: **http://www.interaktonline.com/mxcss**

Simplifying Web Development

Interakt

AJAX, JSON, PHP, and Flex Together

Building a Google-finance type site

by Mike Potter

This article previously appeared in WebDDJ vol 4 iss 6, but due to a transmission error it was not complete. We apologize and have the pleasure of reproducing it in full here.

We've all seen Google finance, and the great job that it does at mixing HTML content with Flash content, as seen on company stock quotes. Google has done a great job at using Flash where it makes the most sense, in the graphs for company quotes, with HTML where it makes sense (linking to news items). Today, I'll show you how to build your own Google-finance type site, using a combination of Ajax, JSON, PHP and Flex / Flash. And, best of all, we'll do it all for free.

In this example, we'll grab data from a Feedburner feed, and then show the hits to that feed in a graph, over time. We'll bring in the RSS information from our blog feed, and when we click on the items in the graph, they'll highlight the RSS feed items for that day.

To complete this tutorial, you'll need the software / accounts shown on page 20.

Note: You can perform this same tutorial without Flex Charting. You would need to create your own Flex component, and you could do that with the SDK. So, technically this tutorial could be accomplished for $0. In my case, to reduce development time, I'm using the Flex Charting components, which cost $249.

The first part to this tutorial is to install all the above software. I won't go through that here, if you have problems please comment in my blog.

Believe it or not, there actually are pieces of Adobe software that I don't use, feel free to use Photoshop to design the appearance of the webpage, Lightroom to hold photos of pepople you will put on the HTML page, Premiere to do video editing, ColdFusion can replace the PHP part fairly easily etc... I leave it up to you to extend this tutorial to make use of every piece of Adobe software.

Once we've got everything installed, we'll start by hooking up our PHP backend to Spry. The reason that we need the PHP backend is because Ajax applications cannot load data from outside sources: we need to load data from the same source as the Ajax application. So, we've created a small PHP file that will go to Feedburner, get the stats and then print the output. That PHP file is very small and very simple:

```
<?php
$fpURL = 'http://api.feedburner.com/
awareness/1.0/GetFeedData?uri=adobe/
mpotter&dates=2006-07-01,2006-07-17';
$handle = fopen($fpURL, "r");
while (!feof($handle)) {
$strOutData .= fread($handle, 8192);
}
fclose($handle);
header('Content-type: text/xml');
echo $strOutData;
?>
```

Any server side language could do something similar.

And here's the JavaScript code for Spry to call and load that PHP file:

```
var dsFeedburner = new Spry.Data.
XMLDataSet("getdata.php", "/rsp/feed/
entry");
```

When the HTML page loads, it will call getdata.php, and populate the Spry dsFeedburner data source with data from that file. Here's a sample of the XML content that gets output from that PHP file.

```
<?xml version="1.0" encoding="UTF-8"?>
<rsp stat="ok">
<!--This information is part of the
FeedBurner Awareness API. If you want
to hide this information, you may do
so via your FeedBurner Account.-->
<feed id="412263" uri="adobe/mpotter">
<entry date="2006-07-01" circula-
tion="0" hits="0"/>
<entry date="2006-07-02" circula-
tion="0" hits="0"/>
...
```

The text "/rsp/feed/entry" is simply an xpath expression to get to each <entry... item in the XML file.

So, now we've got the data. To populate a table with data, we do the following:

```
<table border="1" spry:
region="dsFeedburner">
<tr>
<th onClick="dsFeedburner.sort('@
date');">Date</th>
<th onClick="dsFeedburner.sort('@
hits');">Hits</th>
</tr>
<tr onclick="alert({@date});" spry:
repeat="dsFeedburner" spry:select="Sel
ectedFeedburnerItem" spry:hover="Hover
FeedburnerItem"> <td>{@date}</td>
<td>{@hits}</td>
</tr>
</table>
```

We repeat each table row (<tr>) with the spry:repeat="dsFeedburner" attribute. We set the selected item's CSS class to "SelectedFeedburnerItem", and the hover state to "HoverFeedburnerItem". You can edit the style for those items with simple CSS, in the HTML file (or an external CSS file, whichever you prefer.)

OK, so if we run that, then we should see a repeatable table showing dates and hits to our items. Pretty good, now let's hook that up to a graph component that we made in Flex.

Its important to note here that Flex can be used to create a number of components. Its possible, with a little work, that you could create a component that exactly matches the graphing component that Google uses, with a date slider / selector at the top, above the graph. I won't do that here, for the sake of simplicity, but it is possible. Or, you could create an Ajax media browser that plays videos in Flash. The possibilities are limited only by your imagination.

So, let's go ahead and create our Flex graph... Here's the MXML code:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.
adobe.com/2006/mxml" layout="absolute"
width="400" height="400" backgroundGra
dientColors="[#ffffff, #ffffff]">
<fab:FABridge xmlns:fab="bridge.*" />
<mx:Script>
<![CDATA[
import mx.charts.events.
ChartItemEvent;
import mx.collections.ArrayCollection;
import com.adobe.serialization.json.
JSON;
public function bindJSONToChart(
JSONString:String ):void {
var arr:Array = (JSON.
decode(JSONString) as Array);
linechart.dataProvider = new
ArrayCollection(arr);
}
public function
chartClicked(clickEvent:
ChartItemEvent):void
{
if( ExternalInterface.available )
ExternalInterface.call("chartClicked",
clickEvent.hitData.item);
}
]]>
</mx:Script>
<mx:LineChart id="linechart"
paddingLeft="5" paddingRight="5"
showDataTips="true" left="10"
right="10" top="10" bottom="10" itemCl
ick="callEI(event)">
<mx:horizontalAxis>
<mx:CategoryAxis categoryField="@date"
displayName="Date"/>
</mx:horizontalAxis>
<mx:series>
<mx:LineSeries yField="@hits"
displayName="Hits"/>
</mx:series>
</mx:LineChart>
</mx:Application>
```

Everything should be fairly self-explanatory. The only lines that may be unfamiliar are:

```
<fab:FABridge xmlns:fab="bridge.*" />
which is needed for the Flex / Ajax
bridge to function properly and:
public function callEI(func:String,
clickEvent:ChartItemEvent):void
{
if( ExternalInterface.available )
ExternalInterface.call("chartClicked",
clickEvent.hitData.item);
}
which is needed to call back to the
page that contains the Flash file, in
our case the Ajaxed HTML page.
OK, so we've got our Flex application
built, we've got our Ajax file, now we
need to pass data from the Ajax page
to the Flex application. Here's how
you do that in JavaScript:
var obs = new Object;
obs.onPostLoad = function(notifier,
data)
{
Spry.Debug.trace("obs.onPostLoad
called!");
};
var notifierData;
obs.onDataChanged = function(notifier,
data)
{
notifierData = notifier.data;
try {
with( FABridge.example.root()) {
bindJSONToChart( jsmin( toJsonString(
notifier.data ) ) );
}
}
catch( e )
{
var initCallback = function( )
{
with( FABridge.example.root() ) {
bindJSONToChart( jsmin( toJsonString(
notifierData ) ) );
}
}
FABridge.addInitializationCallback("ex
ample",initCallback);
//alert( "Error in onDataChanged"+e );
}
Spry.Debug.trace(toJsonString( noti-
fier.data ));
};
dsFeedburner.addObserver(obs);
```

There's lots going on here. First, we create an observer on the data's "onDataChanged" method, so every time the data is changed, we send new data to the Flex application. Although we only use this once, when the page is loaded, you could use this if you had say a drop down list of a number of your feeds, and wanted the data source to change when the drop down changed. When you changed your Spry data source, the Flex graph would refresh because of this observer.

Then, we hack around a bit. First of all, we need to convert the data to JSON format:

```
toJsonString( notifier.data )
```

We also need to minimize it, because Flex's JSON Library doesn't like new lines or carriage returns in that data (this took me a long time to figure out!). So, we run:

```
jsmin( toJsonString( notifier.data ) )
Finally, we pas the result of that
function to the method in our MXML
file, bindJSONToChart():
bindJSONToChart( jsmin( toJsonString(
notifier.data ) ) );
I'm lazy and I've made that all one
line in the code.
```

Now, it will do that fine if the .swf file has loaded. However, sometimes the data gets returned before the .swf file is actually loaded. I found this out when running the application locally. So, I've wrapped

Software / Account	Why do we need it?	Cost?	Where do I get it?
Feedburner with API access turned on	*Data to populate an Ajax page and Flash graph.*	*$0*	*www.feedburner.com*
PHP	*Acts as a proxy for our Ajax application.*	*$0 / Open Source*	*www.php.net*
	Connects to Feedburner and prints out XML.	*Spry Framework for Ajax*	*Create the Ajax page*
$0 / Open Source	*labs.adobe.com*		
Flex and Flex charting	*Create the .swf file that will show the graph and call back to JavaScript*	*Flex SDK: Free*	
Flex Charting: Free trial.	*www.adobe.com/go/flex*		
Flex / Ajax bridge	*Call ActionScript (Flash) from JavaScript*	*$0 / Open Source*	*labs.adobe.com*
ActionScript 3 Library for JSON	*Parse the JSON that we're passing to Flex*	*$0 / Open Source*	*labs.adobe.com*
JSON and JSMin Library	*Remove new lines and carriage returns in JSON*	*$0 / Open Source*	

"Its important to note here that Flex can be **used to create a number of components**"

all this in a try / catch statement, and added an initialization function on the swf file, so that if its not loaded, when it loads it will run this function and populate the graph with data from the Ajax call.

There you go. When you load the HTML page, Spry will load that PHP file using an Ajax call, the PHP file will connect to Feedburner, get the XML data, print it out. Spry will read that in, bind it to the HTML elements on the page, then call the Flex application, passing it the data in JSON format. The Flex application will read in the data and display it in a Flex chart. Here are a few tips and tricks that I've found when building this out:

1. I found it easiest to modify the HTML that gets output by Flex Builder, rather than to reference the built swf file. If you do that, be sure to modify index.template.html in the html-template folder of your Flex project, rather than the .html files in the bin/ directory of your Flex project. The HTML files in bin/ get overwritten when you save and re-build your Flex application, and if you modify those, rather than index.template.html, you'll lose your changes.
2. I started building out the Flex graph using JavaScript and the Flex / Ajax bridge. I don't recommend that. Build out your entire Flex component in Flex Builder, then simply write functions to pass data to it from HTML. I think that's easier than trying to build Flex components using JavaScript.
3. Similarly, write functions that closely couple your Flex application to your HTML page. For instance, you'll notice that in my MXML file, I call the chartClicked JS function using:if(ExternalInterface.available)
4. The ActionScript 3 JSON library doesn't like new lines or carriage returns in the JSON data. You need to strip those out before sending data from JS to ActionScript over the Flex / Ajax bridge.
5. I don't know of any other technologies that I could use in this tutorial to get a higher click through on MXNA. Well, maybe Web 2.0.

```
ExternalInterface.call("chartClicked", clickEvent.hitData.
item);
```

I could have attached an observer to the lineChart instead, using the Flex / Ajax bridge, but I find it easier to get the data items and debug the application in Flex Builder, rather than trying to do that in JavaScript on the HTML page.

For answers call us at 1-866-EDGEWEB
3343932

When calling your web host for support you want answers, not an annoying song stuck in your head from spending all day on hold. At Edgewebhosting.net, we'll answer your call in two rings or less. There's no annoying on-hold music, no recorded messages or confusing menu merry-go-rounds. And when you call, one of our qualified experts will have the answers you're looking for. Not that you'll need to call us often since our self-healing servers virtually eliminate the potential for problems and automatically resolve most CF, ASP, .NET, SQL, IIS and Linux problems in 60 seconds or less with no human interaction.

Sleep soundly, take a vacation, and be confident knowing your server will be housed in one of the most redundant self-owned datacenters in the world alongside some of the largest sites on the Internet today and kept online and operational by one of the most advanced teams of skilled Gurus, DBAs, Network and Systems Engineers.

By the Numbers:

- 2 Rings or less, live support
- 100% Guarantee
- 99.999% Uptime
- 2.6 GBPS Redundant Internet Fiber Connectivity
- 1st Tier Carrier Neutral Facility
- 24 x 7 Emergency support
- 24 Hour Backup
- Type IV Redundant Datacenter

EDGE WEB HOSTING

For a new kind of easy listening, talk to EdgeWebHosting.net

http://edgewebhosting.net

CFDJ READERS' CHOICE AWARD · MXDJ READERS' CHOICE AWARD
2003 - 2006

◦ Shared Hosting ◦ Managed Dedicated Servers ◦ Managed Colocation ◦ Semi-Private Servers
◦ ColdFusion ◦ BlueDragon ◦ ASP ◦ .NET ◦ .Linux ◦ .Java ◦ SQL Server ◦ .MySQL ◦ Self-Healing Servers

Flex and LiveCycle – Platform for Enterprise Success

http://weblogs.macromedia.com/swebster/archives/2006/09/flex_and_livecy.cfm

by Steven Webster

When I joined Macromedia/ Adobe Consulting, my role as Practice Leader was to grow a practice in EMEA around Rich Internet Application technologies. In recent months, we have taken the decision to bring together the Rich Internet Application practice with our LiveCycle practice, reflecting the recurrence in opportunity for us to fuse these client and server technologies together to solve a number of enterprise business problems. As I assume the responsibility to grow this combined practice in EMEA, I'll extend my blogging to share some of the excitement and thinking we have around how the technologies from our Enterprise Developer Business Unit – including Flex, Flex Data Services and the numerous LiveCycle technologies – converge as an enterprise business platform.

I'd like for this blog post to initiate some thinking on what this might mean.

I expect that people who track this blog – by the nature of where it is generally aggregated – are already familiar with Rich Internet Application technologies. I expect these readers already have a strong understanding of the importance and effectiveness of a user-experience that enables consumers, customers, citizens, workers or all manner of other individuals (user is such a non-personal description of the various persona of people who interact with business through a user-interface) to perform their tasks with applications that are not only useful, but usable and desirable.

Steve Webster is the practice director for Rich Internet Applications at Adobe.

What is LiveCycle ?

These same readers may not be as familiar however with the LiveCycle suite of technologies. I'm not going to go into exhaustive detail on this suite of technologies, but would highly recommend a browse through the Adobe LiveCycle section of our corporate website. Nor am I going to attempt here to be exhaustive of the business benefits of LiveCycle, but will instead aim to call out some concrete examples of where the fusion of Flex and LiveCycle allows us to tackle even more of the business challenges that our richer and more effective user-experiences have set out to address.

The Business Benefits of RIA…

I'm a great believer in education through example, and understanding through analogy, so let me try and paint a picture of the current landscape.

The ineffectiveness of online application forms not delivering on the promises of 24/7 at-your-leisure access when delivered in a browser, is well understood. Take a mortgage or a loan application from your online bank – these are characterised as frustrating processes that typically employ between 8 and 11 different HTML pages, where each page refresh results in 5% of applicants making an abandonment decision. The problems of page-refresh are well understood - technological frustrations associated with navigating back through an already completed process or session time-outs, user-experience frustrations such as being forced to complete the form in a sequential manner (what I often compare to "reading a book through a straw") rather than more familiar "complete the easy stuff and come back to the hard stuff", or the situation where you need to go and find some supporting information, like your social security number. I can pitch this "failed promise of the Internet" for an hour, so I'll spare you here.

Rich Internet Applications replace these multi-step, complex processes with simpler, easier, more effective single-screen or guided-completion processes, often employing UI metaphors such as the accordion to deliver a user-experience that more effectively matches the interaction patterns of real users.

An analogous problem to financial service application forms, is the same problems manifesting themselves through retail checkout experiences – addressing these problems with RIA have been shown in blind A/B tests to improve the appalling abandonment rates of 75% with increased conversion rates of checkout customers by 50%. So with RIA, we're well on our way to delivering efficiencies through user-experience and design-led solutions that attack the problem from

the user's perspective rather than a technology perspective.

I'll talk more of such design-led approaches to RIA solutions in future blog entries.

…fuse with the Business Benefits of LiveCycle

But typically, the rich user-experiences that we deliver with RIA are often the beginning of a more complex document-based process, that inevitably crosses the digital world to a world where people interact through paper.

Let's consider some of these inefficiencies, by returning to our loan application.

Immediately you complete your application (and by replacing that application process with a design-led RIA, there are many more of you that have completed that application compared to the old days of the several HTML forms) you enter the world of workflow, and the world of paper.

In fact, many organisations will insist that you become the protagonist in their paper play. The final step of your mortgage application will often require that you print your application as a form, sign that form, and post that form to the bank.

If you print the form, if you sign it, if you remember to put it in your bag, if you buy the envelope and the stamp and if you remember to post it, then some days later, once your piece of paper has survived the postal system and made it's way to the desk of a loan processing officer, your application is ready to be processed. Electronically. But it's a bit paper. So what are we going to have to do … that's right, having taken you from the state-of-the-art RIA-enhanced electronic world back to a piece of paper, the very next step is to capture that form information digitally again.

So how will LiveCycle help us here? Well first and foremost, moving seamlessly between the RIA and the PDF world is a simple and obvious fusion of Flex and LiveCycle. One of the key aspects to LiveCycle, is the process of document generation – leveraging technologies such as LiveCycle PDF Generator to automate the creation of PDF files or LiveCycle Assembler to "form-stitch" together document fragments to create customised PDF documents.

More effective however, is to remove the paper-trail altogether, and this is where LiveCycle can really begin to optimise the behind-the-curtain processes in an enterprise RIA solution. In addition to creating a dynamic PDF representation of our application form data, LiveCycle Document Security will enable the customer to digitally sign and encrypt a document before it is electronically submitted to the bank for processing, removing the need for manual rekeying when the document is received.

However, if the bank does require receipt of a paper document with an ink signature, the manual rekeying can still be eliminated. LiveCycle Barcoded Forms automate the capture of user-provided data from printed or faxed forms, by representing that data as a 2-dimensional barcode that can be easily and rapidly scanned, removing the processing and errors associated with manual rekeying of user-supplied data.

Once your loan application is born again as an electronic application form in back-office systems it must be processed; typically this processing is a complex hand-off of your application between different departments, or different people within a department. Perhaps you have to be credit-scored, before your application is then allowed to proceed to the next department, who are responsible for checking your employer references. A gap in your employment perhaps triggers the need for a letter to be sent to you requesting further information – otherwise, your application progresses to a decisioning process that determines the maximum amount of loan you will be offered. You are then sent a letter either approving or declining the loan, and this kicks off another step in the workflow, where your approved loan exists for 30-days. After 30-days, your loan is no longer valid, until another credit-check has been performed. And all the while, you have no insight as to where you are in the loan approval process, or indeed whether your loan has even been received at all.

What we have here, is an example of a complex workflow associated with a transient document – your loan application – as it moves between a variety of systems and people. LiveCycle Workflow Server allows us to streamline these human-centric processes, allowing either the visual or programmatic assembly of "workflows", which allow your electronic application to move quickly and securely through an organisation.

Meanwhile LiveCycle Policy Server could be employed, ensuring that information remains confidential, specifying not only who has access to the document, but what they can do to the document, when they can do that

Meet Robert

A business executive at a popular social media site

Challenges

- Encountering poor video quality due to exponential growth in traffic.
- Increasing bandwidth costs due to growing audience size.
- Experiencing compatibility issues due to user-generated video arriving in multiple formats.

Solutions

- VitalStream Streaming Services – Improved quality of end-user video experience using a scalable and global content delivery network.
- VitalStream Advertising Services – Transformed the delivery of digital media from a cost center into a profit center.
- VitalStream Transcoding Services – Automatically converted all user-generated content into the leading streaming media format.

Providing End-to-End Solutions for Your Business

VitalStream is more than a rich media delivery company. We are your professional partner providing solutions that meet your unique business challenges. To learn more about VitalStream solutions, call 800-254-7554 or visit www.vitalstream.com/go/solutions/

VitalStream®

Digital Media Solutions for Your Business

to the document (for instance, once the customer receives their loan approval as a PDF, can they update their mobile telephone number in-site, but not change their address), and for how long they can make these changes. Furthermore, Policy Server will allow all these security policies to be updated even once a document has been issued – revoking access to a loan approval after 30 days, or even confirming that the user has printed the loan application approval, removing the need to send them a paper-copy in the post).

And what of keeping you, the customer engaged ? A Flex application upon a LiveCycle workflow, leveraging Flex Data Services, can offer real-time notification to a user as to the receipt, the progress and the success of their loan application. The enterprise technology stack enables a two-way communication between customer and bank, not a one-way flow of information into a black hole.

And the list goes on…

As I said earlier, I have no desire to be exhaustive in my overview of LiveCycle technologies in this post, but I hope that the above real-world scenario outlines how – effective as a Rich Internet Application can be – we can offer significant additional return on investment when the RIA on the front-end is supplemented with the process automation, document generation and information assurance services that LiveCycle can offer in the back-end.

And it's not just in financial services; think of all the online processes that you have considered that can be improved and innovated upon with RIA technology – not just financial services and online banking, but travel booking, online check-in for flights, retail commerce, making a movie reservation, booking seats at the theatre, hiring a car, submitting your tax return, filing your expenses for approval, etc, etc.

As limited as you are by your imagination, think how many of these opportunities for insanely great user-experiences also result in something physical and tangible in your hand as the result of a process of automations, approvals, workflows or generation of documents.

Like account statements. Like travel itineraries. Like boarding passes. Like movie or theatre tickets, with barcodes that can be scanned at point of sale or point of entry.

Think of how a richer user-experience can not only benefit business with inc reases in conversion rates, increases in the effectiveness of cross-selling and up-selling, increases in repeat-business or increases in customer acquisition through customer evangelism. Think of how a richer user-experience can benefit from the efficiencies of automatic and electronic processing.

Think of the value you can begin to offer your clients, and their customers. Think of how Flex and LiveCycle might work together for you.

Summary

A couple of years ago, I was using the phrase "Enterprise RIA" to define the kind of online applications that could benefit from a richer user-experience while performing complex integration with new and existing systems and infrastructure. The fusion of RIA and LiveCycle dramatically lowers the barrier to delivery of these enterprise RIA, upon a recurring solution platform for enterprise business applications.

I'm tremendously excited about the solution opportunities and business problems our consultants are going to be delivering with the combination of Flex and LiveCycle.

In future blog posts, I'll speak more of how some of these integrations can be achieved technically, whether you are a J2EE architect that wants API-level access to these services, or someone who would rather orchestrate these solutions in a visual development environment, and integrate them with your Rich Internet Applications.

Additionally, I'll be aiming to share some of the leadership our User Experience practice, and RIA and LiveCycle practice consultants are creating around Flex and LiveCycle.

I hope this blog post lifts the lid on a whole new world of enterprise RIA development, while kick starting your thinking about the solutions you could deliver, and the skills you could leverage or the partnerships you could create in order that you are more suitably positioned to deliver these solutions.

"So how will LiveCycle help us here? Well first and foremost, moving seamlessly **between the RIA and the PDF world is a simple and obvious fusion of Flex and LiveCycle.**"

Making Great Mapping
Mashups Using Adobe Flex

Geotagging + Spatial & Temporal Widgets = Mapping Magic

by Mansour Raad

This article is based on a presentation that I made at the Adobe Flex seminar in August 2006 (http://www.flexseminar.com/), after which the master of ceremonies Jeremy Geelan asked me to explain how to make great mapping mashups using Adobe Flex.

My name is Mansour Raad. I'm the senior software architect of ESRI's ArcWeb Services. ESRI (www.esri.com) is a provider of Geographic Information Systems (GIS). On any given day, more than one million people around the world use ESRI's GIS to improve the way their organizations conduct business.

A geographic information system is a system for management, analysis, and display of geographic knowledge, which is represented using a series of information sets such as maps and globes, geographic data sets, processing and work flow models, data models, and metadata. A GIS can produce information that answers specific questions and allows you to share that information with others. By visualizing relationships, connections, and patterns in data, you can make informed decisions and increase efficiency throughout your organization.

ArcWeb Services are GIS Web services that are hosted by ESRI and uses ESRI's software and best-of-breed geographic data. Basically, we have created Web services out of our geographic processing engines and made these available for commercial consumption.

Mansour Raad is the senior software architect of ESRI's ArcWeb Services, a provider of Geographic Information Systems (GIS). mraad@esri.com

In other words, there is no software to install and no data to maintain. We take care of that for you by providing a 7 by 24, 99.9% uptime in two geographically separate data centers. The system currently produces over six million maps per day. And, ArcWeb Services offers much more than just street maps. We can give you aerial photos, satellite pictures, shaded reliefs, topographic, demographic, thematic, business listings. Some of the data such as satellite pictures are static, others are dynamic such as traffic, weather and current earthquake locations and magnitudes. Using our portal application at www.arcwebservices.com, you can create custom map styles by layering data using your own cartographic criteria. That is part of the beauty of our system: we not only host our data, but we can host your data too. You can upload your own address points, lines and areas of interest.

When it comes to geo-processing mashups, ArcWeb Services also provides capa-bilities to locate places such as "London", points of interest such a "Disneyland", dot domain names such as www.esri.com, or Internet Protocol (IP) addresses such as 63.241.153.171. The latter is perfect for creating mashups of locations derived from firewall attack logs, or finding the locations of prospective buyers that are surfing the net to purchase a home.

In addition, if you're a Sprint or a Canada Bell Mobility user and have granted access to the location of your cell phone, we have Web services that enable applications to locate you via GPS or tower triangulation. As you can see, these services are international in their scope. We also provide more than "A-to-B" routing. Our enhanced Web services give you directions and path geometries for a multi-stop route, and they can solve routes for the least distance or time while avoiding specific barriers and traffic locations.

With ArcWeb Services, you can mash up live data (such as traffic) and static data (points) and inject time constraints. For example, if you need to be at this location between 10 and 11 a.m. and at this other location between 1 and 2 p,m., you will get the best route that considers all stops based on current traffic conditions and where you are right now!

The services that we provide are more than just a simple "where is the nearest pizza shop"; for example. ArcWeb Services provides drive time and drive distance solutions. Here is a perfect, if dramatic, example. An Amber Alert is reported at a specific location and you need to rapidly find the half-hour, one-hour and a two-hour drive-time destinations from that location that will intersect major and minor highways so officers can be dispatched for road blocks (see Figure 1). I hope that none of us have to do this search, but you can see that there is a lot of expressiveness in our services and data and it has to come out somehow,

and thus enters Adobe Flex. "Finally" you say after this soap-box speech.

To best experience what I have just described, just open a browser and go to http://www.arcwebservices.com/awx/index.jsp. There, you will see a Flex-based application that is hosted by ESRI. We have been using Flex since the first beta. One of the things that attracted us to Flex is the fact that it's a Java-based Web application. You can run it in Apache Tomcat; just take the .war file and drop it in the webapps directory and voila.

The second impressive point about Flex is the abundance and range of built-in, rich client-side widgets such as trees and tables. As a developer, I don't have to build these items from scratch, and instead put my energy into creating Rich Internet Applications like the ArcWeb Explorer application that you opened.

Another of Flex's strong points is the integrated support of Web services. Since ArcWeb Services is a Web-based service, this feature is imperative for our development initiatives. Let me walk you through an example. If you still have the ArcWeb Explorer open, you have a map of the world in front of you. Let's switch to a satellite view. Click on the "Map Types" top menu element and a widget will show up. Click on the satellite icon, and the map should switch to a tiled satellite view. Dispose of the widget by click-ing on the top left X. I love the dispose effect, which is one of the many cool effects of Flex. Now, you can hold down the left-mouse button and pan the map around. You can quickly zoom to an area of interest by holding down the shift key and dragging the mouse while holding down the left button. Hold down the right-mouse button, and select "World Map" to see the full world again. Select the "Find" menu item. A single line input widget should show up.

Click on the computer icon to invoke the IP search Web service to locate yourself (most likely, your ISP provider) on the map. Mouse over the map marker and click on the globe icon to get more information about your location. Cool, or what?

Next, click the "Clear" button on the Find widget and type 909 793 2853 (ESRI phone number) and click Go. A phone marker should appear on the map. Next, erase the 909 number and type "Disneyland" (no quotes) and hit enter. The map should center onto Disneyland. Now, let's say that you want to get the directions from 909 793 2853 to Disneyland. Click the "Directions" menu item to get the directions widget. From the Find widget, select the row with the telephone icon by clicking on it; while holding down the left-mouse button, drag it on top of the Directions widget and let go (Drag and Drop is just one of the many built-in behaviors in Flex). Do the same for the Disneyland row and click Go in Directions. Isn't this how it is supposed to be? The directions should appear and the map is updated with the route. Since this is a Rich Internet Application, the data is transferred and resides on the client side. This can be seen by moving the mouse over any section of the route where a tooltip will appear and give information about that section without a round-trip to the server.

Clear the directions and dispose of the Directions widget and let's get some census information about the area around ESRI. Drag-and-drop that 909 telephone number from the Find widget onto the map. This will automatically center the map. Click the Reports menu element and the Reports widget will show up. Click "Use Map Center" as the location and a two-mile radius for a report area definition. Click on "Get Report". This will invoke the report Web service, which will send back a PDF document summarizing the 2000 census information. Again, this is cool.

Now that you have learned how to manually do all these cool things in Flex, I will show you how to do them programmatically. In addition to the SOAP interface, we provided a REST and a JavaScript-based API for mashup purposes and for people who think that SOAP is too "heavy." The documentation of the REST API can be found at http://www.arcwebservices.com/v2006/help/index.htm#rest/samples.htm. URLs are cre-ated with parameters to create maps in Flash format. You can create apps with very little coding that pan as well as zoom in and out. You can acetate lines, points, and polygons. In addition, legends and north arrows can be specified for authentic cartographic representation. And, because the geographic data is vector based and generated on the fly, a user can specify a map projection for distance or area fidelity based on the center of the view in latitude and longitude. An example of what I have just described can be found at: http://www.arcwebservices.com/services/v2006/restmap?actn=getMap&tkn=mytoken&ds=ArcWeb:TA.Streets.NA&c=-117.1817|34.0556&sf=52500.

Figure 1

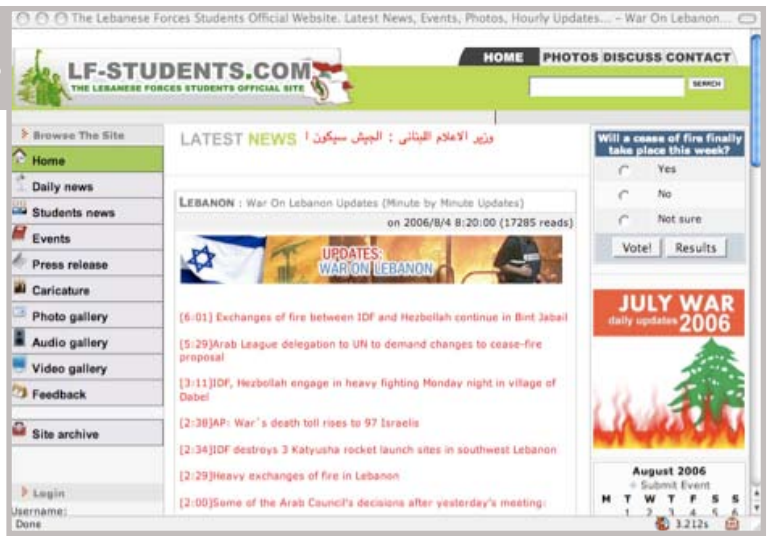
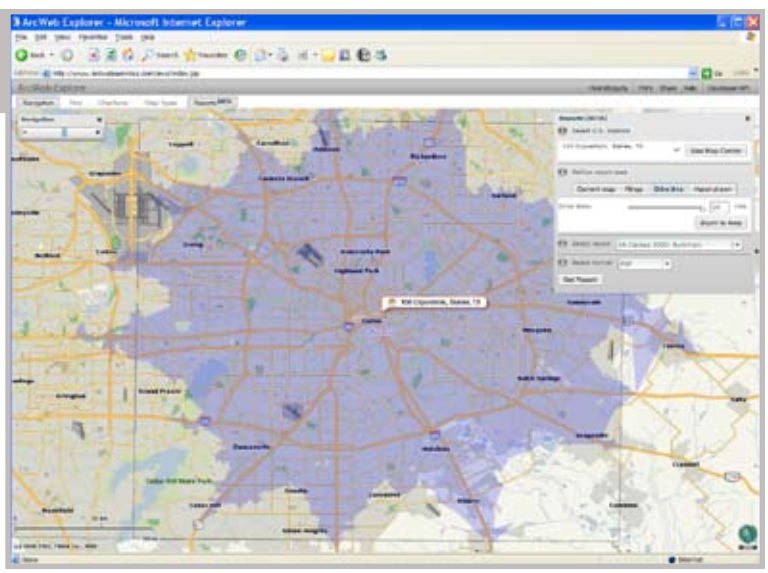


Figure 2



If you are interested in experimenting with different parameters, I included a refer-ence to the source code of a Flex 2–based application that you can download in the re-sources section. Compile the application using the Flex2 SDK (http://www.adobe.com/products/flex). This will produce a swf file that you can load in a browser with the ubiquitous Flash player.

Now for great mapping mashups. From Wikipedia: "A mashup is a Website or a Web 2.0 application that uses content from more than one source to create a completely new service." What does that mean in terms of Flex, ArcWeb Services and you? Your data (residing on your server) published in XML format from an HTTP/GET request can be placed, or mashed up on top of map generated from ArcWeb Services in an expressive Flash/Flex-based client application. Your data can be persistent in a flat file or an RDBMS, and it can be extracted using a .NET ASP, ColdFusion, or JSP application that is called from the browser client application and converts it to markers on the map.

Let me illustrate the power of mashup in a somewhat dramatic personal account. Back in July and August 2006, the news from the Middle East was dominated by the war in Lebanon between Israel and Hezbollah. I followed the war

from a text-only blog that detailed the events minute-by-minute and where they occurred (see Figure 2). I thought it would be more expressive if I could screen-scrape this text information and send it to ArcWeb Services for geo-tagging and totally mutate the page to display a new page with a spatial widget and temporal widget. The spatial widget (the map) can show me where events are happening, and the temporal widget will give me their sequence. Then I wondered: What if I could click on one widget and get info from the other. Would I see patterns? Now this type of application is really expressive and much more informative! Mashup really change your perspective.

Using the Aardvak Firefox extension (http://karmatics.com/aardvark/), I drilled down to the HTML section on the page that contained the to-be screen-scraped info. It con-sisted of an unorder list element (ul) with list item (li) children tags. The text in the li tag consisted of a time-stamp followed by a description of the event and sometimes where the event occurred (for example, 6:01 Exchanges of fire between Israel Defense Force (IDF) and Hezbollah continue in Bint Jabail).

To screen scrape the desired info from the page, I used greasemonkey located at http://greasemonkey.mozdev.org/. I always wanted to write a grease-monkey script and here was the perfect opportunity. greasemonkey is a Firefox extension that enables you to install a JavaScript document on the fly. The script is executed when the user navigates to a specific page. The script has access to the document object model and can mutate the document and communicate with a server-side component.

```
function getXml()
{
var xml = "<data>";
var unorderedList = document.
getElementById("news");
var childNodes = unorderedList.child-
Nodes;
for( var j = 0; j < childNodes.length;
j++ )
{
  if( childNodes[j].nodeType == 1 )
  {
    xml += "<event>";
```

Figure 3


```
    xml += childNodes[j].textContent;
    xml += "</event>";
  }
}
xml += "</data>";
return xml;
}
```

The above is a script snippet locates the unordered list element (ul) and iterates through its children (li), extracting the text content to compose an XML document. This function is called when the user clicks on the AW Mashup icon injected by the script at load time. The XML document is HTTP posted to the server for geo-tagging. The latter is a Web service that scans a document and locates in that document places such as "Beirut" or even more powerful places such as "20 miles north of Haifa, Israel". Each identified placed is returned as part of the Web service response with a latitude and longitude value. The collection of the geo-tagged information is forwarded to a server-side viewer (in my case a JSP page) to produce a page with a spa-tial widget and a temporal widget. The geo-tagger information is then placed on each widget to express its dimension in a visual way. The spatial (map) widget is based on the ArcWeb Explorer (http://www1.arcwebservices.com/v2006/solutions/awx.jsp). This is a Flash 9 AJAX widget that enables the user to view vector and raster geographic features. It can be controlled using a JavaScript API and enables a developer to add "intelligent" markers with mouse-over effects and mouse-click callbacks. The temporal widget is based on the SIMILE Timeline project (http://simile.mit.edu/timeline/). This is another DHTML-based AJAX widget that enables the user to visualize time-based events. A user can pan the timeline by dragging the timeline horizontally and can represent the event in different bands such as hours and days.

Now, greasemonkey script is executed whenever I navigate to the text-based blog URL. The script screen scrapes the relevant information (ul and li text content) and sends it to the server to be geo-tagged. The geo-tagged information is forwarded to a JSP page that renders the information using the ArcWeb Explorer and the Simile Time-line widgets (see Figure 3).

I hope you enjoyed reading this article. I have included URL references throughout that will give you more in-depth information. In addition, the source code for the items that I have developed are referenced in the resources section.

I would like to dedicate this article to all who died in the aforementioned war.

Resources:

- Flex2 Rest Sample : http://arcscripts.esri.com/details.asp?dbid=14736
- News AJAX Mashup : http://arcscripts.esri.com/details.asp?dbid=14737

The Web Is Exciting Again

There has never been a better time to be building Web applications than now

By Ryan Stewart

ColdFusion developers have known for years how powerful rapid development can be and how much of a difference that makes when building dynamic Web applications. Over the course of a little more than a year we've watched as the Web model was turned on its head in favor of something that feels much more intuitive and is much more user-friendly. Part of this change has come about because of what most people call "Web 2.0." Web 2.0 has brought about more buzzwords than a marketing convention. A lot of normal users have stopped trying to figure out what tagging is, what AJAX means, or what the blogosphere really is. They have to be wondering why most of the Web sites they're using suddenly have "beta" on them.

"Web 2.0 has brought about more buzzwords than a marketing convention; a lot **of normal users have stopped trying to figure out what tagging is, what AJAX means, or what the blogosphere really is"**

Despite all of this, the thing that matters most to Web users is experience. No matter what you think about Web 2.0 and the obscene amounts of money venture capitalists have decided to throw at all of these "beta" Web applications, it's difficult to deny that people are looking at the Web in a whole new way. The Web is exciting again, and more people are using it every day. The potential for developers is almost limitless. Whether your goal is to sell your company to Google or just make a few people's lives more comfortable, the technologies at our disposal are making that easier than ever. But that's part of the problem. We have so many technologies that it gets confusing to sort out what we should start looking at let alone what we should ultimately use as we build our Web applications. Luckily, there are some best practices that we can use to sort out the mess.

What Is Your Target Audience?

In the past, developers didn't have to worry about what people were going to be using to view their Web applications. The language they developed in, whether it was ColdFusion, PHP, ASP.NET, or any other variety, didn't really matter. For one, most of the Web was using Internet Explorer, which made it difficult to implement a standards-compliant CSS page, but it also meant that everyone was basically seeing the same thing. Any small differences caused by the browser as a result of the back-end language (of which there were few) didn't come up. Web developers lived in a fairly homogenous world.

This has changed. A variety of different Web browsers have sprung up, and all seem to handle Web standards differently. More people are also using Macs, which means yet another browser, Safari, that handles things very differently from most of the major Windows-based browsers. But most importantly, people have come to rely on browser technologies to power their Web applications. To provide a rich experience on the Web, developers are leveraging technologies like JavaScript, CSS, and DHTML, all of which look very different depending on the browser you're using, and when combined, becomes an even bigger headache.

A technology like Flash solves this problem because it runs on top of the browser via the plug-in and it's on 98% of the world's computers. If you're trying to reach as wide a base as possible, and you're looking for a powerful, rapid solution for developing Web applications, Flex 2 is the way to go. Web developers will be right at home in MXML, the XML-based language that powers Flex 2. If you're used to JavaScript, you'll also be able to pick up ActionScript 3, the language behind the scenes of Flex 2, quickly. ActionScript 3 is based on the new ECMA standard, and JavaScript developers will find it very easy to use, but also quite a bit more powerful.

So when thinking about your audience, take into account what browser and operating system they're using. If you're writing a Web application for people who are all going to be using Firefox and IE then AJAX is a possibility. But if you're planning to deploy to a wide audience, or even an enterprise-level audience, Flex 2 is the best way to go. It gives you the wide reach of the Flash Platform, and all the richness of a desktop application.

How Expandable Does This Application Have to Be?

There are a couple of angles to this. One, how does your application have to adapt over time? I've seen some powerful things done with AJAX, but one thing that a lot of people don't seem to consider is how JavaScript is adapting as a language and how support for JavaScript may change over time. While there are a lot of frameworks that help users get over the issues of cross-browser compatibility, the core functionality of JavaScript is five-years-old. There are radical changes ahead for the ECMA standard, and as a result the next version of JavaScript. When you combine that with the different ways that browsers handle JavaScript, building an AJAX application that will be compatible with tomorrow's browsers becomes a dicey proposition.

Flash doesn't have this problem. In fact, the newest version of the Flash Player is compatible with the very first version of Future Splash from way back in 1996. This is an incredibly important consideration. It means you can develop your application, and when a new version

STREAM57

Customized Flash-based media solutions

Software and services for collaboration, video conferencing, and e-learning

visit us
stream57.com

e-mail us
streamline@stream57.com

call us
212.909.2550 x1012

comes along, take your time in deciding what features to embrace and what features to leave out. AJAX has the benefit of being standards-based, but the standards body isn't really beholden to anyone. First, Macromedia and now Adobe have a history of making sure that they don't alienate their developers by rendering the time and energy they've spent on development useless.

The second thing to consider when it comes to expandability is obviously being able to add to your code base and quickly build new features. This means being able to find developers as well as being able to enhance your application easily. When it comes to developers, JavaScript is an easy language to find them for. Because it's been around so long, there are a lot of people that know the ins and outs very well. Flash engineers are harder to come by, but they're growing every day. AJAX applications win out when it comes to developer numbers, but when it comes to enhancing the application, Flex wins. AJAX frameworks like Dojo, Prototype, and even Adobe's Spry framework make getting into AJAX easy. They even solve a number of the compatibility issues. But when it comes time to make enhancements, they fall short. You're essentially trying to build out your application on two separate code bases: yours and the framework you're using. Flex follows a component-based model that allows for quick and easy expandability. Flex 2 components are one of the more underrated features of the framework. Developers can be working on what seem like two separate applications and very easily bring them together when it's time for a final deployment. That flexibility is a dream come true for developers trying to keep up in the ever-shifting world of Web applications.

What Kind of Infrastructure Do I Have? What Does My Budget Look Like?

Nothing beats free, and if you looked at Flex 1.5 while it was out, you know that there were some enormous costs associated with it. For an enterprise-level shop, it wasn't a big deal, and a lot of enterprises have been using Flex for a long time. But for a start-up or smaller Web application companies, when comparing Flex and AJAX, the relative cost of the Open Source AJAX made it a no brainer. But Adobe listened to the complaints and Flex 2 is entirely free. You can download the Flex 2 SDK from Adobe right now and start building Flex applications. Buying Flex Builder 2, which is built on top of the Eclipse platform, gives you powerful debugging tools and all of the trimmings you'd expect from a world-class IDE. But even Flex Builder 2 isn't expensive, starting at $499 without the charting components. A steal compared to Flex 1.5 and still not much when you compare the benefits it provides versus other IDEs.

You also want to think about how all of your technologies are going to work together. You may be using ColdFusion to drive your Web development, or you may have chosen to go Open Source with PHP. You could be using all Microsoft technologies and your Web applications are built on top of .NET. AJAX is universal – it doesn't matter what your code looks like on the middle tier because it's only concerned with the client. This is a pretty compelling reason to use AJAX, because you want your client technology to be as agnostic as possible. However Flex 2 has also made some great strides in this area and Adobe has worked hard to make sure that Flex 2 works with almost any technology you put behind it. There are some great resources out there for building a Flex 2 application on top of PHP or .NET. For the ultimate rich Internet experience, Adobe's ColdFusion works seamlessly with Flex 2. You can also take advantage of Action Messaging Format (AMF) with either ColdFusion or PHP. AMF is a binary format used to transfer data between the client and the server. It results in smaller data sizes and more security. ColdFusion has been engineered to use AMF and there are some fantastic PHP solutions including AMFPHP. Recently, the team over at Midnight coders released a version of their WebORB software that allows for AMF between Flash and .NET. There are a lot of options when it comes to Flex and your infrastructure.

What Are My Time Constraints?

For a long time, people associated Flash with a very tedious, time-consuming development process. The nature of Flash didn't lend itself to rapid development in any way shape or form – it was a tool made for designers. When looking for rapid development, there are a plethora of AJAX frameworks out there to get you started, and JavaScript is an easy language for people to pick up and start building. Now, the Flex 2 framework has abstracted a lot of the tedious parts of Flash in the same way that AJAX frameworks do for JavaScript. Jumping into Flash development has never been easier, and you have never been able to crank out great looking applications in less time than you can now.

JavaScript has served developers very well over the years, and its resurgence in the form of AJAX has helped foster a lot of the creativity that we're seeing on the Web now. But many of the good things about AJAX are matched by Flex 2. Flash has come a very long way since its days as a tool for Animation, and more developers are realizing the benefits of a rich desktop-like experience, without the headaches of worrying about browsers and operating systems. However, if you've invested a lot in AJAX, it may be difficult to give up on the technology entirely. Luckily, AJAX and Flex do some great things together. Adobe released the Flex AJAX Bridge on Adobe Labs, which lets AJAX developers get their feet wet in Flex 2 without making a big commitment. It's a great way to try out the richness of Flex 2 and the improvements without rewriting your entire code base or trashing your AJAX investment.

It's still always about choosing the right tool for the job, and AJAX is a fantastic technology when it comes to enhancing Web sites and making the entire Web experience better. But when you're setting out to develop Web applications, and you want them to have the richness that people associate with their desktops, what at first may seem like big advantages for AJAX turn out to be covered very well by Flex 2. Flex is a young technology, and many developers are just now starting to see what it can do, let alone try to push the boundaries. AJAX, on the other hand, is built on technologies that have been through a lot, but have very little left in them when it comes to pushing the envelope. As you make your development decisions going forward, give Flex 2 a look and see how it can help you. I guarantee you'll be impressed by how far Flash has come.

REGISTER TODAY AND SAVE!

Rich Internet Applications: AJAX, Flash, Web 2.0 and Beyond...

www.AjaxWorldExpo.com

AJAXWORLD™ EAST
CONFERENCE & EXPO

NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

SYS-CON Events is proud to announce the AjaxWorld East Conference 2007!

The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this unique, timely conference – especially the web designers and developers building those experiences, and those who manage them.

BEING HELD APRIL 2-3 2007!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested in learning about a wide range of RIA topics that can help them achieve business value.

A Fireworks Quickie - Optimizing Text

Getting good quality

by Jim Babbage

Getting good quality text in a JPEG file has long been a concern for many designers. Fortunately for us, Fireworks MX and higher versions have a very useful feature to help with this called the Selective Quality. In Fireworks MX 2004 and higher, we also have the ability to adjust and customize the aliasing of text. In this short tutorial find out how to utilize these features to keep your text crisp and snappy when exporting a file as a JPEG image (see Figure 1).

We'll begin by looking at the custom anti-aliasing attributes in the Properties inspector. Fireworks MX 2004 and later give you the ability to alter the strength of text aliasing through five preset options, or by creating your own custom anti-aliasing setting. The enlarged screen captures to the right give you an idea of what the different presets do and how they blend pixels when anti-aliasing.

Anti-Aliasing is used to smooth out the "jaggies" most noticeable in areas where a shape travels in a curve or diagonally. Anti-Aliasing adds additional edge pixels of varying tones to help blend the edges of text (or other vector shapes/strokes) with the image behind it. The most obvious difference is with no anti-aliasing. There is no blending of the text edges and the stair-stepping (aka jaggies) is very evident. The system Anti-Aliasing on my XP system comes next for crispness, followed by Crisp, Strong and Smooth.

The other option available is Custom Anti-Alias, where you control the Oversampling, Sharpness and Strength (see Figure 2). Oversampling controls the amount of detail used in creating the blend between the text and the background. Sharpness controls the smoothness of the blend. Strength controls how much the text edges will blend with the background. The higher the values, the crisper and sharper your text will be. Changing the anti-alias settings does affect the file size in a minor way; when previewing my image, the file size fluctu-

Article courtesy of Community MX. For more quality articles such as this one, go to www.communitymx.com

ated by about 1 kb or less when testing out the various presets and custom settings. In my opinion for this file that was a pretty acceptable range.

Once you have determined a suitable crispness for your text (if you have that option in your version of Fireworks, that is), it's time to make some optimization considerations. Because the main part of this image is a photograph, JPEG format is still the best option for exporting the file. However, depending on how much you compress the file in the Optimize Panel, your text quality can suffer greatly. JPEG format is not very kind to areas of solid color, like text.

Use the Preview, 2-up or 4-up tab to determine the best overall compression/quality for your image. Don't worry about the text at this point; our first step is to find that fine line between quality and file size. This setting will vary depending on the amount of detail and range of color in your image.

Once you are happy with the main quality setting, click on the Edit Selective Quality Options icon, as seen in Figure 3.

Choose Enable Selective Quality and you should notice that Preserve text quality option is checked (see Figure 4). If it isn't, select it now.

The general idea with Selective Quality is that you create bitmap selections around areas where you want to maintain more detail (less compression). These selections get converted to JPEG masks and will be optimized at whatever quality setting is within the selective quality input field. This way you can compress other areas of your JPEG file at higher levels (lower quality=higher compression), while still maintaining detail in important regions using the selective quality masks.

Now, one thing that you may not know, is that you don't even need to make a bitmap selection if your only concern is text quality. Once the option is selected, changing your JPEG compression for either of the two quality settings won't have a big impact on your text. I do recommend though that you keep both settings the same. I have found that if you have a higher value in the Selective Quality field than the main quality field, you actually end up with worse results.

Other Optimization Tips

- JPEG compression tends to soften an image. You can set a small to moderate amount of Unsharp Masking via the Live Effects (Live Filters) to help counteract this softening. You will gain a bit of file size, but the overall result may be more to your liking.
- GIF format is best for images that are solid color, such as logos or illustrations that don't utilize gradients. While you can use GIF format for true grayscale images, you will get better compression in JPEG format.
- Make sure to preview your optimization settings in one of the preview tabs, so you know what your image will look like when compressed.
- Do not use Sharpen JPEG Edges and Smoothing options at the same time; image quality will be reduced.
- For large images, JPEG is still the best option. PNG format preserves more detail, but it doesn't compress nearly as much.

Figure 1

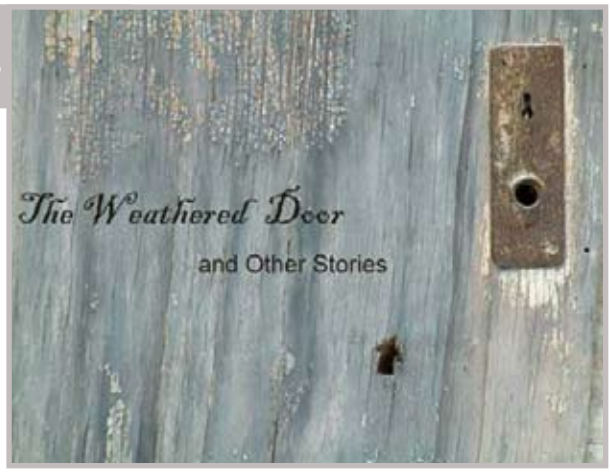


Figure 2

Figure 3

Figure 4

Flex at AJAXWorld Conference & Expo 2006

An entire track devoted to Flex

by Jeremy Geelan

As the biggest-ever i-Technology event devoted specifically to AJAX-compatible Rich Internet Application technologies, it is hardly surprising that AJAXWorld Conference 2006 had an entire track devoted to Flex.

It seems worth recording here in WebDDJ the details of the various sessions devoted to Adobe's powerful application development solution for fast, end-to-end development of cross-platform RIAs within the enterprise and across the web.

The sessions, which last an entire day, aim to show in full technical detail, from a variety of hands-on perspectives, how Flex 2 developers can utilize Flex's library of rich built-in controls and services and its robust data services model for developing richer enterprise and web apps using a standards-based programming framework and a powerful Eclipse based IDE.

Flex with Java | James Ward

Is Flash the ubiquitous, cross platform, client side virtual machine we always wanted Java to be? Flex applications are compiled to Flash bytecode and execute in the JIT-enabled Flash virtual machine. With Flex you can rapidly build web applications which can make RPC calls to backend Java objects (POJO, EJB, Spring, etc), send and receive JMS messages, and more. Voila! Write once, debug once, run anywhere actualized for web applications! Come experience why Java developers everywhere are building next generation web experiences with Java and Flex.

Speaker Bio: James Ward is a Technical Evangelist for Flex at Adobe. Much like his love for climbing mountains he enjoys programming because it provides endless new discoveries, elegant workarounds, summits and valleys. His adventures in climbing have taken him many places. Likewise, technology has brought him many adventures, including: Pascal and Assembly back in the early 90's; Perl, HTML, and JavaScript in the mid 90's; then Java and many of it's frameworks beginning in the late 90's. Today he primarily uses Flex to build beautiful front ends for Java based back ends. Prior to Adobe, James built a rich marketing and customer service portal for Pillar Data Systems.

Flex: Delivering ROI to Enterprise Apps | Mike Nimer

There are many options today for building Web 2.0 applications - also known as Rich Internet Applications. Adobe's Flex 2.0 offers one of the richest, most powerful options in this field, which also allows for incredibly rapid development, increasing the total Return on Investment (ROI) for the application.

Speaker Bio: Mike Nimer has recently joined forces with fellow guru Jeff Tapper to form Tapper, Nimer and Associates to provide expert guidance on Rich Internet Application (RIA) development and mentoring. Mike is a formerly a member of the ColdFusion engineering team, responsible for the Flex2 integration with ColdFusion and a number of other features in ColdFusion, such as Flash Forms. Before joining the engineering team he spent three years working as a senior consultant with the Macromedia consulting group, where he provided on-site assistance to customers around the world with their architecture planning, code reviews, performance tuning, and general code issues.

Taking RIAs to the Desktop with Apollo, Adobe's Next-Generation Client | Luis Polanco

Learn about Adobe's next-generation client that will extend the reach and capabilities of today's Rich Internet Applications, freeing them to run outside the browser, on desktops. Join Apollo product management and experience compelling RIAs for the desktop built using familiar web development technologies including HTML, AJAX, Flash, and Flex.

Speaker Bio: Luis Polanco is the senior product manager for Apollo at Adobe. Luis has managed a variety of next-generation client and server software, including desktop applications and enterprise software products. Prior to the Apollo team, Luis worked on Adobe's Rich Client Initiative. Most recently, Luis has led product development for an Operation Intelligence product suite targeted to Fortune 100 companies, which leveraged Flash and Java technologies, and a Business Process Management platform for large enterprises.

Creating Great Mapping Mashups Using Flex | Mansour Raad

In this session, you will be exposed to ESRI SOAP/REST ArcWeb Services, and how Flex can access geoprocessing and mapping capabilities. In addition, ArcWeb Explorer, a Flex-based application, will be demonstrated and integrated into an AJAX-based mashup to visualize real-world events.

Speaker Bio: Mansour Raad has over 20 years of experience in the IT field. As the senior software architect of ArcWeb Services at Environmental System Research Institute (ESRI), he is using his command of Internet technologies to design the next-generation Internet solutions. ESRI is a leader in Geographic Information Systems (GIS). Currently, Mansour is the Flex evangelist within ESRI and is leading the Flex-based ArcWeb Explorer solution. He has been using Flex since its Royal days. In addition, while at ESRI, he was a team lead in architecting and implementing ArcIMS, the premier mapping solution on the internet. Mansour assisted several large companies in implementing and integrate mapping solution in their enterprise. He graduated from Boston University with a masters degree in Aerospace Engineering. With the combination of his IT experience, he designed and implemented an airport noise and operation monitoring system (ANOMS) that is currently used in over 30 airports world wide.

SAP's Use of Flex to Change the Face of Enterprise Applications | Boris Kibisher

First SAP built an AJAX application to build model-driven applications, then augmented SAP NetWeaver Visual Composer's capabilities with the ability to render the resulting models into Flex. In this session, SAP product management and development will discuss the rationale for architecting and deploying such an approach. SAP will also share early customer feedback.

Speaker Bio: Boris Kabisher is an engineer in SAP Visual Composer development team, where he works on the User Interface Modeling framework. Prior to joining the Visual Composer team, Boris was a founder and chief architect of number of technology startups, focusing on state-of-the-art GUI and user experience.

Case Study: The JBoss Flex-based Mail Client | Andrew C. Oliver

When the developers of JBoss Collaboration Server determined their product needed a Web mail client, they quickly realized that Flex was the best option to build it with. Flex allowed them to rapidly build a superior user experience, which works the same on any OS and browser, and is easy to maintain and improve. Since the Flex SDK is free like the JDK, they could easily incorporate it. Come learn more about why the developers of JBoss Collaboration Server choose Flex and how they built the Flex-based mail client.

Speaker Bio: Andrew C. Oliver is a professional cat herder who moonlights as a software developer. He's been developing in Java since 1998, primarily as a consultant for large companies with the inevitable dot-com thrown in between. He is a former member of the Apache Software Foundation, former member of the Apache Jakarta PMC, and founder of the Jakarta POI project. When he's not off globetrotting to provide training, consulting and support to JBoss, Inc. customers, you can find him bit-twiddling with some obscure file format or protocol, or JBoss Collaboration Server.

Do You Speak Web?

A look at a young, new programming language - haXe

by Nicolas Cannasse

The recent years of Web development have been marked by several important events. In the Flash world, we have seen a constant evolution of the technology. From a developer's point of view, Flash started to be really usable with the introduction of ActionScript 1.0 in Flash 5 and the improvements made in Flash 6. It turned out that Flash was no longer only an animation tool for designers but also a framework for developers. This led to many Flash projects that would not have been possible before, ranging from Web games to Rich Internet Applications.

Flash

The usage of Flash by developers has been greatly increased over the years. With the introduction of ActionScript 2, people familiar with existing object-oriented language such as Java could start developing Flash content more easily. With the new graphical effects of Flash 8, game visuals could be greatly improved. And even more recently with Flex2 components, one can now develop a working RIA in a matter of hours.

However, with all these additions, a lot of early Flash adopters might have been left behind, lacking time to learn all these new technologies and at the same time, feeling pressured by all the new exciting things that were becoming possible. This development will be ongoing, and actually, for someone who wants to develop a modern Web site, there is even more to learn.

The second major change was the progressive replacement of Internet Explorer 5-generation by IE6, which integrated better JavaScript and CSS support. Together with the emergence of standard-compliant browsers such as Firefox, Safari and Opera, this led to the next hot thing - AJAX.

In the Web world, where not everybody needs the very complex interactions that Flash can provide, AJAX works well with Flash to provide an enhanced user experience. For the Flash developer who wants to use some AJAX features in his existing Web site, the technologies are complementary.

The Server-Side of Things

In order to develop a Web site, you also need to know some server-side technology. Since the unique Perl CGI choice from a few years ago, a lot of server-side languages are now available, including Java, PHP, ASP, Python, and more recently, Ruby-on-Rails. All these languages are clearing the gap by providing indirect and filtered access to the server filesystem and database.

To master server-side technologies, you need to learn a good number of frameworks, sometimes concurrent, rarely standardized. You need also to understand how to communicate efficiently with the client-side, either with Flash (by using Remoting) or JavaScript (using XML or JSON).

Issues

Whether for the single developer or the team that has mastered all these different technologies, a set of new problems arises from using them all together in one Web site. Communications between the different developers of a team are not always easy since they use different technologies. Sending messages between the different languages is not always transparent, due to their technical differences, so an XML intermediate representation is often needed, which adds duplicate work for both client and server. And since different languages are preventing code reusability, developers need to write the same code twice, and keep everything synchronized. A typical example is a complex form validation that needs to be performed on both server- and client-side. That's where haXe comes in.

What Is haXe?

haXe is a new technology that simplifies Web development by providing a common foundation programming language for both client- and server-side technologies. You can use haXe to program Flash, JavaScript/AJAX, and the server-side of your Web site. You can even use haXe to develop desktop applications.

By reducing the number of pieces in the Web puzzle, haXe simplifies the overall Web site development process. Because the developers in a team are all using the same technology, they can understand each other and share ideas more efficiently. Since this is the same language everywhere, messages can be sent transparently between the different layers of the Web site, enabling full Flash/JS/Server automated communications. And with the same syntax for all platforms, some code can be reused on

both the server- and client-side, without having to rewrite it or keep everything synchronized.

For the Flash developer, haXe is a command-line compiler that can be used to compile a set of haXe classes into a SWF file. Flash Players from 6 to 9 are supported, giving the haXe developer a lot of freedom of choice (ActionScript3, by comparison, only supports Flash Player 9). For people already familiar with the MTASC ActionScript2 compiler, haXe is its successor.

To develop JavaScript/AJAX code, the haXe compiler will output a single .js file, including all the compiled classes. Programming haXe instead of JavaScript is increasing productivity: haXe is strongly typed so most errors are caught early in the compilation phase. haXe is class-based so the JS developer can abstract some visual components into classes and reuse them later in several projects. Since only needed classes are compiled to the .js file, it's easier to manage than a lot of small .js includes. More importantly, haXe-generated JavaScript code is cross-browser, and haXe includes some standard libraries - such as XML manipulation and XmlHttpRequest - needed to write cross-browser AJAX code.

Finally, haXe classes can be compiled to run on the Neko Virtual Machine - a small embeddable VM with a low footprint and very good performance. There is a mod_neko for the Apache Web Server (similar to mod_perl or mod_php), so you can use haXe to connect to a database and generate dynamic Web pages. Standardized API are available to wrap database tables using classes (SPOD), generating HTML from a Template system, or simply accessing the local filesystem and network.

This article will focus its examples on both JavaScript and Flash technologies, which might be more familiar to the reader.

haXe API

haXe is sometimes wrongly accused of trying to do too many things at one time, with the belief that it will somehow be the less common denominator between all these platforms. But that's not how haXe works.

haXe is a programming language with which you can define classes and write code. It has a standard library that has been unit-tested to work the same on all platforms. This standard library includes API such as XML, String, Date, Array, and Reflection.

Each platform has its own specific API, such as Browser DOM for JS, MovieClip and Sound for Flash, database and filesystem for Neko. These APIs are the same ones you would usually use in ActionScript or JavaScript. This makes haXe very easy to learn; all you need to do is to adapt to the small differences in haXe syntax.

Hello haXe!

Everything I've said until now might sound like nothing more than a nice theory, so let's dive into practical programming to see how things are working in the real world.

```
class Hello {
    static function main() {
        trace("Hello haXe !");
    }
}
```

This class is the most simple haXe example. Let's try to compile and run it on the three supported platforms. For that, we need to write an .hxml file that will list compiler parameters:

```
-main Hello
-swf hello.swf
```

If haXe is installed, you should be able to run the HXML file by double-clicking on it, and running it from the command line by using "haxe file.hxml". This will produce a hello.swf file that, when opened, will display the text directly on the screen. Now, let's change the HXML file to compile for JavaScript instead of Flash:

```
-main Hello
-js hello.js
```

Again, this will produce a hello.js file. Let's include the JS in the following HTML page and open it:

```
<html>
  <body>
    <div id="haxe:trace"></div>
    <script type="text/JavaScript"
src="hello.js"></script>
  </body>
</html>
```

That will display the text at the position of the div. As you will have noticed, this is not terribly different from what one would write in Java or ActionScript. haXe syntax is very user-friendly and 90% compatible with most popular programming languages. Therefore, it's easy for any programmer familiar with one of these languages to write haXe code.

Multiplatform haXe Code

So how is the trace function - which has different behavior depending on the platform - implemented? Is it some kind of haXe black magic? Not exactly. While platform-specific code can't be compiled for other platforms, haXe accepts conditional compilation statements that enable a different implementation depending on the platform. For example, let's look at the following method:

```
function displayError( msg : String
) {
```

```
  #if flash
    // flash-specific code
    var mc = flash.Lib._root;
    mc.createTextField("tf",mc.getNex-
tHighestDepth(),0,0,100,30);
    mc.tf.text = msg;
    mc.tf.textColor = 0xFF0000;
  #else js
    // JavaScript-specific code
    js.Lib.alert(msg);
  #end
}
```

When the displayError function is compiled for Flash, it will use the Flash code, and when it's compiled for JavaScript, it will use the JavaScript code. This enables developers to have some code that - while using platform-specific code - can be compiled and run on the different platforms. This is actually the way some of the haXe standard libraries are implemented.

It is interesting to notice that conditional compilation statements can also be used with your own tags, such as the following:

```
function myMethod() {
   #if mydebug
   // debug version, adding traces
and doing more checks
   ....
   #else true
   // standard version, optimized
   ...
   #end
}
```

Then, you only have to compile with haXe command line parameter -D mydebug to use the debug version of this method. This mechanism enables you to compile several versions of your application depending on your needs.

Debugging

We already saw how the trace method can be used on all platforms; each trace code adds information about the file, class, method, and line number where the trace is performed, and you can redefine at runtime the trace method behavior to log information according to your needs.

Also, haXe programs can be compiled with -debug in order to add debugging information. In Flash and JavaScript, this will produce bigger files and has a little performance post, but you will get information about uncaught exceptions, with the complete stack trace. This feature is enabled natively in Flash 9, if you are using the Debug Player, but haXe adds it to Flash before Flash 9, and to JavaScript, no matter what browser is being used. These features save a great amount of time and energy in debugging haXe programs.

Conclusion

haXe is open source software that is available on Windows, Mac, and Linux. Its license scheme enables you to use it to develop both open source and commercial Web sites without any restrictions. It's a young language, but it's already supported by an active and enthusiastic community of developers who are contributing additional libraries and tools.

The haXe Web site (http://haxe.org) features an automatically generated API reference and a good number of tutorials in order to get started with haXe. The haXe distribution includes a tool called haXelib to share libraries between developers, and a small Web development server to test your Web site locally. Tools such as ScreenWeaver HX (http://haxe.org/swhx) enable you to create a cross-platform haXe desktop application.

haXe technology is used in the production of several professional Web sites with tens-of-thousands daily users. While haXe productivity is maximized when it's used on both the client- and server-side, it's possible, by design, to use it only on one platform, since haXe can interact with other languages as well. For example, it's possible to interoperate between haXe and ActionScript and use Flex components from haXe. A haXe Flash client can also communicate with a Flash Remoting Server by using haXe Remoting. This kind of interoperability enables developers to use haXe without having to change their whole architecture.

Will the haXe approach of simplifying Web development - by having a standardized language - be adopted by a large amount of Web developers? It's up to you to decide.

"For the Flash developer who wants to use some AJAX features **in his existing Web site, the technologies are complementary."**

Visit the *New* www.SYS-CON.com Website Today!

The World's Leading *i*-Technology News and Information Source

24/7

FREE NEWSLETTERS
Stay ahead of the i-Technology curve with
E-mail updates on what's happening in your industry

SYS-CON.TV
Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!
Read web logs from the movers and shakers or create your own blog to be read by millions

WEBCAST
Streaming video on today's i-Technology news, events, and webinars

EDUCATION
The world's leading online i-Technology university

RESEARCH
i-Technology data "and" analysis for business decision-makers

MAGAZINES
View the current issue and past archives of your favorite i-Technology journal

INTERNATIONAL SITES
Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

IT Solutions Guide
Information Storage+Security Journal
JDJ
Web Services Journal
.NET Developer's Journal
LinuxWorld Magazine
Linux Business News
Eclipse Developer's Journal
MX Developer's Journal
ColdFusion Developer's Journal
XML Journal
Wireless Business & Technology
Symbian Developer's Journal
WebSphere Journal
WLDJ
PowerBuilder Developer's Journal

Using a Hierarchy of Components

...to Populate a Flex Tree Control

by Steve House

This article is about creating a Flex tree control that uses a component hierarchy as the data provider. As with most of my Flex development, after struggling for days and then finally getting something to work, I later find out that there is a much easier way to do it that none of my searches ever turned up. I am a beginner Flex 2 developer, so any constructive feedback would be greatly appreciated.

So, if you are like me you probably have a hierarchy in your database that looks something like this:

```
iSubjectId = 1
   sSubjectName = Science
   iParentSubjectId = null
iSubjectId = 2
   sSubjectName = Biology
   iParentSubjectId = 1
```

In the above example, the top level subjects have null for their parent IDs and then second+ levels use the ID of their parent. When using CFCs to represent this hierarchy I would do something like this:

```
Subject.cfc
   iSubjectId = 1
   sSubjectName = Science
   children = array of Subjects
    [1] = Subject.cfc
      iSubjectId = 2
      sSubjectName = Biology
```

Above I have a nested hierarchy of subjects where the top level subject contains an array named children that has child subjects in it and so on and so on down the hierarchy. My SubjectGateway component would contain a function named readSubjectHierarchy that would return an array of top level Subject components, each containing its child hierarchy. Like so:

```
Returned Array:
[1] Subject: Science
  children: Array
  [1] Biology
  [2] Chemistry
[2] Subject: History
  children: Array
  [1] American History
    children: Array
    [1] 1492 to 1860
    [2] 1861 to present
  [2] European History
[3] Subject: Math
  children: Array
  [1] Algebra
  [2] Calculus
```

OK, if you are still with me…

Enter Flex 2

So with Flex 2 in the picture, I thought that the best way to allow my user to select a Subject was with a tree control…right? All of the tree control examples I found had a hardcoded XML dataprovider which made great trees, but didn't help me one bit. Who uses hardcoded data? I then found an example where the hierarchical data was converted to XML and used as the dataprovider. Details of me trying to use that were recorded on the Adobe Forums Database driven tree control (http://www.adobe.com/cfusion/webforums/forum/messageview.cfm?catid=582&threadid=1159438).

This was working but didn't seem ideal at all.

The Flex 2 documentation surrounding Hierarchical Data kept pointing me to the ITreeDataDescriptor interface (http://livedocs.macromedia.com/flex/2/langref/mx/controls/treeClasses/ITreeDataDescriptor.html), which I tried using in so many ways I lost count, but it never seemed to work right. Most of the time, I would end up with either a flat list of items or all items showing up as folders including the bottom levels which had nothing beneath them. Since I only wanted the user to select the bottom levels, this was not allowing them to select anything. Not good.

Hmmmmm…

What I kept seeing, was this reference to a property of the object called children which was supposed to be an array of the child objects. Well wait a minute, I have that, but why does it think that all the levels have children? Then it hit me, I wondered if the presence of the array (even when empty) was making it think that there are children. ColdFusion has no NULL right, so I am always initializing an array to get it to return correctly from the getter method. Maybe this is the problem. So how can I get the getter to not return an array and not error. Well here is my solution:

```
Subject.cfc (important sections shown)
<cfcomponent output="false"
alias="extensions.CF.Subject">
<cfproperty name="iSubjectId"
```

```
type="numeric" default="0">
<cfproperty name="iQueueId"
type="numeric" default="0">
<cfproperty name="iParentSubjectId"
type="numeric" default="0">
<cfproperty name="sSubject"
type="string" default="">
<cfproperty name="children"
type="array" default="null">
//Initialize the CFC with the default
properties values.
//Note that the children array is not
initialized
variables.iSubjectId = 0;
variables.iParentSubjectId = 0;
variables.sSubject = "";
<cffunction name="getChildren"
output="false" access="public"
returntype="any">
<cfif structKeyExists(variables,"chil
dren")$gt;
<cfreturn variables.children>
</cfif>
</cffunction>
<cffunction name="setChildren"
output="false" access="public"
returntype="void">
<cfargument name="val"
required="true">
<cfif isArray(val) or arguments.val
EQ "">
<cfset variables.children = arguments.
val>
<cfelse>
<cfthrow message="'#arguments.val#' is
not a valid array of children"/>
</cfif>
</cffunction>
… other getters and setters removed
for space…
</cfcomponent>
```

Notes From the Above Code:

- the children cfproperty is listed to support translation to Flex, the default of null is just a reminder for me
- the children propery is not initialized
- the getChildren function returns a type of any to support not returning anything
- the getChildren function checks for the existence of the variable children before trying to return anything
- this code is obviously incomplete but I stripped out a whole lot to get it in this article

My Flex object looks like this:

```
package extensions.components.org.
mydomain.model
{
import mx.collections.ArrayCollection;
[RemoteClass(alias="extensions.
CF.Subject")]
[Bindable]
public dynamic class Subject
{
public var iSubjectId:Number = 0;
public var iQueueId:Number = 0;
public var iParentSubjectId:Number =
0;
public var sSubject:String = "";
public var children:Array;
public function Subject()
{
}
}
```

So the magic is in not returning anything with getChildren when there are no children in the array as well as not initializing it to an empty array from the start as I tend to do.

The tree control looks like this:

```
<mx:Tree id="Subject" dataProvider="{s
ubjectHierarchy}" labelField="Subject"
showRoot="false" width="300"
height="231" itemClick="selectSubject
(event)" />
```

Tune in for Part 2 (which is a ways out since I am waiting on my company to purchase me a license of Flex Builder 2 now that my trial has expired) where I will show how I created a hybrid dropdown/tree control that provides the convenience of dropdown size and selection display with the hierarchy of a tree control.

Figure 1

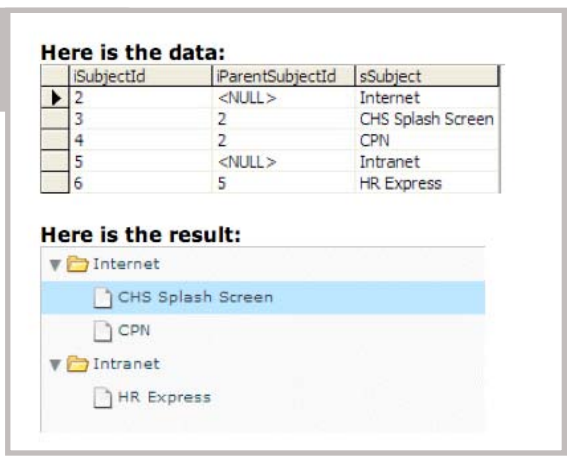
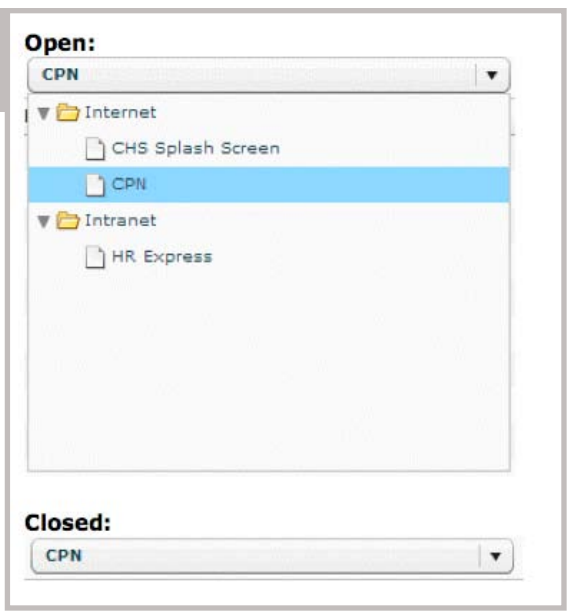
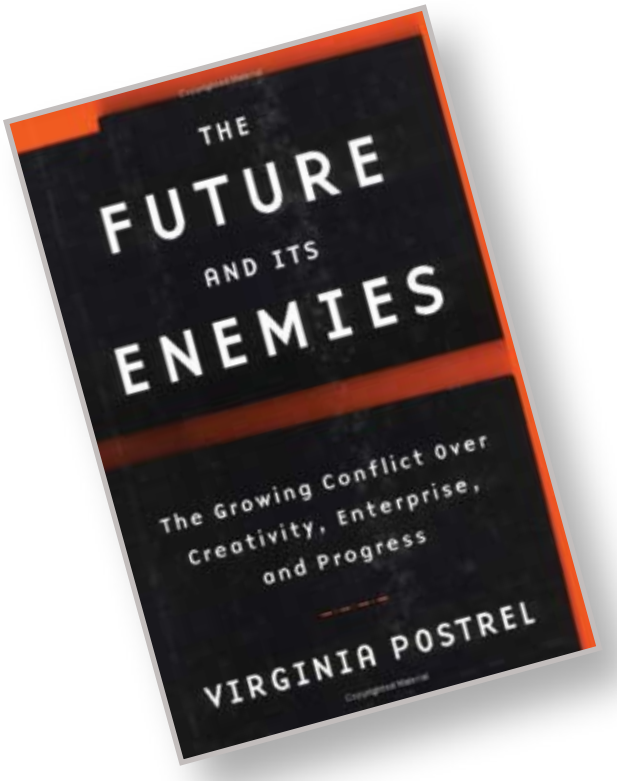


Figure 2





"There is no boundary between 'Web 1.0' and 'Web 2.0' - just a blurry verge"

– continued from page 7

so much as its opportunism, its chaos, and its exhuberance.

Dave Winer - with whom, like TBL, one takes issue somewhat reluctantly - seems to me to make the same mistake, i,e, that of dismissing "Web 2.0" on technological grounds thus missing the point that only one of the elements of "Web 2.0" is technology. Here is what Winer wrote on September 1:

"One sure sign of a bubble is the meta-ness of the excitement. How far removed from actual user experience is the euphoria? Is there any technolog1y involved?"

Robert Scoble take the same stance as Winer, blogging just yesterday:

"The other day I was talking with a developer...and he told me about all the froth he was seeing in the Web 2.0 space. He was wondering where the people were who were paying attention to business. Profits. Customers. He pointed to a lot of the events he's been attending lately and said 'they are frothy.'"

I wonder whether what Scoble's unnamed developer friend really meant was not so much "frothy" as "fuzzy" or "blurry." In which case he has hit the nail on the head, though perhaps not in the way that Scoble realized.

"Web 2.0" is an example of what the historian Daniel Boorstin would have called "the Fertile Verge" - "a place of encounter between something and something else." Boorstin (and here I am wholly indebted to Virginia Postrel) pinpointed such "verges" as being nothing short of the secret to American creativity.

Postrel sums up what Boorstin was saying as follows:

"A verge is not a sharp border but a frontier region: where the forest meets the prairie or the mountains meet the flatlands, where ecosystems or ideas mingle. Verges between land and sea, between civilization and wilderness, between black and white, between immigrants and natives...between state and national governments, between city and countryside - all mark the American experience."

The creativity of America, Boortsin argued - the hope of the nation - lies "in its verges, in its new mixtures and new confusions."

My point is that there is no boundary between "Web 1.0" and "Web 2.0" - just a blurry verge. And the richness of the verge lies in the cross-fertilization and new combinations they encourage.

Web 2.0 is a Boom Town, and - as Postrel points out - "Boom towns break down barriers; they mix together talent from everywhere; they challenge complacency and overturn assumptions. They are sometimes ugly and almost always stressful, but they foster invention, progress, and learning. And they let people chase their dreams."

So my view is that, unlike Asia, the tsunami of change heralded by the current storm in Web development and online business models, coming as it does together with a simultaneous revolution in the way that users are choosing to use the Web, is an unprecedented opportunity for us all. Users, content providers, Web developers, businesses of every stripe.

"Web 2.0" lends itself to totally new types of applications, and its immediacy and openness can be a powerful differentiator in existing business interactions. The trick is to get involved right now, armed with an application architecture that will carry you forward safely and profitably, wherever the "Web 2.0" whirlwind may take you.

Advertising Index

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.


Adobe
Beyond boundaries.
In the right environment, imagination has no limits.
MAX 2006, the annual Adobe user conference, offers a unique opportunity to learn about Adobe software, interact with industry experts, connect with the Adobe community, and explore ideas yet to be imagined.
Choose from over 100 different workshops and hands-on sessions presented by Adobe experts and industry leaders. Exchange ideas with other community members at a variety of networking events. Experience current and emerging Adobe technology.
Join us for MAX 2006 this **October 23-26** in Las Vegas to learn, network, and move beyond the boundaries of what you believe is possible.
Register by September 25 to qualify for a special Early Bird discount of $200 off the standard registration fee.
www.adobe.com/go/wddj
MAX
Beyond boundaries. The 2006 Adobe conference.
Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.