SYS-CON

THE AGE OF RIAS, FLASH, FLEX, AND NOW APOLLO

FORMERLY MX DEVELOPER'S JOURNAL

Web Developer's & Designer's Journal

WWW.WEBDDJ.COM

VOLUME 5 ISSUE 1 **2007**

BUILDING ENGAGING APPLICATIONS AND CONTENT WITH ADOBE TECHNOLOGIES

Simplifying AJAX with the Spry Framework

Delivering great experiences while fitting the workflows and skill sets of both designers & developers

Plus:
- Binary Data, ColdFusion & Flex
- Flex 2 Metadata Tags
- Video Rock 'n Roll with Flex 2

Presorted
Standard
US Postage
PAID
St. Croix Press

YUM!
INTERSPECTACULAR

Adobe® Creative Suite® 2.3

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. Introducing Adobe® Creative Suite® 2.3. 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 the Creative Suite to help bring their ideas into the world. See how they do it at adobe.com/creativemind. It's everything but the idea. Better by Adobe.™

Luis Blanco and Michael Uman,
INTERspectacular

adobe.com/creativemind

Adobe

Web Developer's &
Designer's Journal
January/February 2007

Adobe Solution PARTNER

INTERGRAL information solutions

</cf_bugs>

2.0

FusionDebug™ - The Interactive CF Debugger

Faster. Easier. Smarter.

fusion debug™ 2.0

www.fusiondebug.com

Announcing FusionDebug™ 2.0

- installers for Windows, MacOS, Linux...
- improved user interface layout
- easy to use configuration tool
- improved support for CF frameworks
- step though CF pages, tags and CFC's
- set breakpoints
- view and modify variables
- watch expressions

ColdFusion Hosting is our Complete Focus

POWERFUL HOSTING PLANS
FREE SQL server access | FREE account setup | Unlimited email accounts Generous disk space & data transfer
30 day money-back guarantee | Great value

RELIABLE NETWORK
99.99% average uptime | State-of-the-art data center with complete redundancy in power, HVAC, fire suppression, bandwidth and security | 24/7 network monitoring

FANTASTIC SUPPORT SERVICES
24/7 suppport services | Knowledgeable phone support | We focus on your individual needs

CFDynamics

866.233.9626 CFDYNAMICS.COM

For years we have been involved in the Cold Fusion community and have come to know what developers and project managers look for in a web host. The combination of our powerful hosting plans, reliable network, and fantastic suppport sets us apart from other hosts.
Real service. Real satisfaction. Real value. Real support. Real Freedom.

macromedia ALLIANCE PARTNER

BlueDragonCertified V.I.P. Hosting Partner

Web Developer's & Designer's Journal

Group Publisher Jeremy Geelan
Art Director Louis F. Cuffari

Editorial Board
Aral Balkan
Erik Bianchi
Craig Goodman
Andrew Powell
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

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.

We Have Entered the Age of RIAs, Flash, Flex...and Now Apollo

by Jeremy Geelan

Already, even in pre-release, Adobe's Spry seemed to catch the imagination of many Web professionals wrestling with how to integrate new AJAX frameworks into existing workflows. Created with designers in mind, Spry uses regular HTML tags, CSS, and JavaScript, and is easy to use – boiling down to a couple of JavaScript libraries that you include in your Web page in order to be in a position to add dynamic interactive content to your site.

What spurs me to mention Spry is the approach of AJAXWorld Conference & Expo, of which Adobe is a Platinum sponsor. There is no doubt that it will be the most intense three-day conference anywhere on the East coast this year, devoted to the issues most preoccupying those who design, build, and deploy Web sites and Web-based applications.

There are several Flex sessions at the Conference, including one by Kevin Hoyt – known to many readers of WebDDJ from his regular appearances at various user group forums and conferences throughout the United States. Kevin will be talking about using Flex and AJAX "to bring the 'sexy' back to the enterprise," as he puts it. His session will provide an interactive deep dive on how to integrate Flex, AJAX and Apollo to bring the 'sexy' elements of Web-based RIAs to the enterprise.

As for Apollo, that will naturally be making an appearance at AJAXWorld 2007 East, too. In a session called "Denting the Browser's Chrome: Intense Experiences, Advanced RIA Development and Apollo," Cynergy's Andrew Trice, an Enterprise Applications Consultant for Cynergy Systems, will remind us of the dark decade or so during which the user experiences that drove applications on the web were primitive at best; when, as Trice expresses it, "forms and grids trapped in the browser's content-centric chrome ruled the earth."

The thing is, as Trice notes, grids are not how users think; grids are how programmers think. And the problem is... most of the users aren't programmers.

"In the past," Trice says, "it was easy to claim that the platform was simply too primitive to work any other way. That however doesn't fly anymore. Today with RIAs you now not have only a vast array of rich controls but access to a complete robust vector graphics drawing API. Thanks to all of this, if you can imagine an interface or a way to present data and information to the user, you can make it come alive. More importantly, these RIAs are no longer trapped inside the browser's chrome and the vision of the web driving the applications that run the business of the world is coming alive."

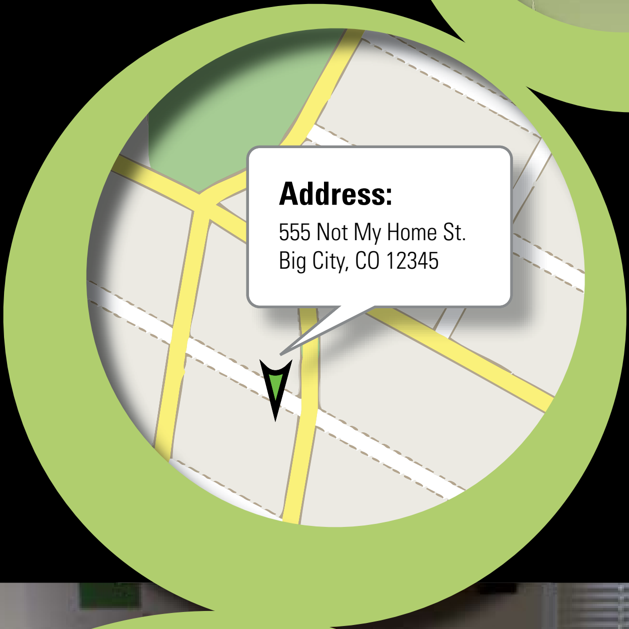
Many other sessions at AJAXWorld are devoted to demonstrating how to turn designs into reality and how to build out incredible user experiences for the Web. Enterprise application developers are embracing AJAX – and increasingly, Flex – to bring the same next-generation functionality to the enterprise. They are designing enterprise applications that perform like desktop software but have the connectivity of the Web and integrate with enterprise systems. Adobe's product road map is guiding developers and designers into exactly the right part of the technology landscape at exactly the right time.

Drive safely, and enjoy!

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. He is executive producer and presenter of "Power Panels with Jeremy Geelan" on SYS-CON.TV, and is actively helping build out the AJAXWorld brand as well as developing entirely new Conferences and One-Day Seminars for SYS-CON Media & Events.

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

CYNERGY

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

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 executive 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.

JavaOne℠

Sun's 2007 Worldwide Java Developer Conference℠

May 8-11, 2007

The Moscone Center
San Francisco, CA

JavaOne™ Pavilion: May 8-10, 2007

IT'S AN EXCITING TIME FOR THE ENTIRE JAVA™ TECHNOLOGY COMMUNITY.

With the evolution of the Java platform, new opportunities are emerging for developers, thought leaders, and entrepreneurs. That's why for 2007 the Conference is featuring an expanded program that embraces technologies outside the core Java platform while keeping Java™ technology at the center. You'll get the same in-depth content we have always focused on, plus more topics and issues relevant to today's evolving market.

LEARN MORE ABOUT:*

> Java Technology
> Scripting
> Open Source and Community Development
> Integration and Services-Oriented Development
> Web 2.0
> Compatibility and Interoperability
> Business Management
> And More

Java™

PLATINUM COSPONSORS

intel ORACLE

GOLD COSPONSOR

NAVTEQ

SILVER COSPONSORS

InterSystems PARASOFT. We make software work.

SAVE $200**

REGISTER BY APRIL 4, 2007

Please use priority code: J7PA1WDDJ

*Content subject to change.
**Offer not available on site.

Register Today @ java.sun.com/javaone

Sun microsystems

Binary Data, ColdFusion & Flex

Sending BitmapData to a server and saving it as a JPG file

by Andrew Trice

Several months ago I posted some articles on my blog about Flex 2 components and accessing/modifying their BitmapData. In one example, I sent the BitmapData to the server and saved it as a JPG file, and I've been asked numerous times since… "How did you do that?" It's surprisingly easy to do once you understand the concepts involved. There are four ways to get binary data from the Flex application back to your server: AMF3 (RemoteObject), Web Services, HTTP Services, or through a Socket connection. In this article I'll cover the first three topics as they pertain to Flex 2; Socket connectivity could take an article all by itself.

Binary data can't be pushed to the server in its native format using a Web Service or a standard HTTP POST method. To save the data using Web Services or HTTP POST, you must first convert the binary data to a text string using Base64-encoding. On the other hand, AMF3 (RemoteObject method) lets you send the binary data to the server in its native binary form. One thing to keep in mind with Base64-encoding is that the encoding process will actually increase the size of the data that's being sent across the wire.

Regardless of how you're sending the data to the server, it's a good practice to compress the data client side whenever possible. I've used the JPGEncoder class at http://code.google.com/p/as3corelib with great success. You can use this class to convert binary image data into a compressed JPG ByteArray that can be sent to the server. This is a good practice for two reasons:

- The data is compressed, which helps decrease latency when communicating with the server.
- The data is encoded into the format that you want to save, so no additional processing/conversion is required on the server. You simply need to save the data either in your file system or in a binary object in your database.

Here's how you get data from a Flex component into a JPG ByteArray: First, you'll have to retrieve the BitmapData from your Flex component. You can pass any Flex component into the following function to retrieve its BitmapData:

```
private function getUIComponentBit-
mapData( target : UIComponent ) :
BitmapData
{
var bd : BitmapData = new BitmapData(
target.width, target.height );
var m : Matrix = new Matrix();
bd.draw( target, m );
return bd;
}
```

Once you have the BitmapData, you'll have to create an instance of the JPGEncoder class and encode the BitmapData. (This example uses the JPG quality of 75.) It's also important to remember that your Flex application will have a slight pause while the encoding is being processed:

```
var bd : BitmapData = getUIComponent-
BitmapData( paintCanvas );
var encoder : JPEGEncoder = new
JPEGEncoder(75);
var data : ByteArray = encoder.encode(
bd );
```

Once you have the data converted to a JPG ByteArray, you're ready to push it to the server and save it. The fastest and easiest way to do that is to use a RemoteObject method and serialize the data using AMF3. This example shows you a method in a ColdFusion Component (CFC) that will let you send the data and save it to the local file system:

```
<cfcomponent name="ImageSave"
displayname="ImageSave"
output="false">
 <cffunction name="ROsave"
access="remote" output="false"
returntype="void">
  <cfargument name="data"
type="binary" required="true" />
  <cffile action="write" file="c:\
temp\ro_data.jpg"
output="#arguments.data#" />
 </cffunction>
</cfcomponent>
```

You can see that the code is actually

very simple. The CFC's ROsave (remote object save) method is expecting binary data as a parameter. When executed, the data is written to the file system using the <CFFILE /> "write" method.

On the Flex side, we'll have to instantiate a mx:RemoteObject:

```
<mx:RemoteObject
id="ro"
showBusyCursor="true"
destination="ColdFusion"
source="BinaryData.cf.ImageSave">

<mx:method name="ROsave"
result="onResult('Data Saved via mx:RemoteObject')"
fault="onFault(event)" />
</mx:RemoteObject>
```

To save the data, we'll invoke the ROsave method and pass the JPG-encoded ByteArray as a parameter:

```
var bd : BitmapData = getUIComponentBitmapData(
paintCanvas );
var encoder : JPEGEncoder = new JPEGEncoder(75);
var data : ByteArray = encoder.encode( bd );

ro.ROsave( data );
```

If you aren't using remoting, you can save the data using Web Services or HTTP services. Most seasoned ColdFusion developers might stop me here and say… "If you're using CFCs as Web Services, why wouldn't you just use them as RemoteObject methods since they are faster?" My response is this: This is just an example. You may be able to take this method and apply it to other technologies where it may be applicable (.NET, Java, PHP, etc.).

```
<cfcomponent name="ImageSave"
displayname="ImageSave" output="false">
 <cffunction name="WSsave" access="remote"
output="false" returntype="void">
  <cfargument name="data" type="string"
required="true" />
  <cffile action="write" file="c:\temp\ws_data.jpg"
output="#ToBinary(arguments.data)#" />
 </cffunction>
</cfcomponent>
```

You can see that the code for the Web Service method is very similar to the previous example. The only difference is that the toBinary method is being used to convert the data from a Base64-encoded string into binary data. The CFC's WSsave (Web Service save) method is expecting a Base64-encoded string as a parameter. When executed, the data is also written to the file system using the <CFFILE /> "write" method.

In Flex, we need an instance of a mx:WebService to save the data:

```
<mx:WebService
id="ws"
 showBusyCursor="true"
 wsdl="/BinaryData/cf/ImageSave.cfc?wsdl">

<mx:operation name="WSsave"
result="onResult('Data Saved via mx:WebService')"
fault="onFault(event)" />
</mx:WebService>
```

To save the data, we first need to Base64-encode it. The following function will take care of that for us:

```
private function base64Encode( data : ByteArray )
: String
{
var encoder : Base64Encoder = new Base64Encoder();
 encoder.encodeBytes( data );
 return encoder.flush();
}
```

We'll then invoke the WSsave method and pass the Base64-encoded ByteArray as a parameter:

```
var bd : BitmapData = getUIComponentBitmapData(
paintCanvas );
```

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

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

Associate Sales Managers
Corinna Melcon, 201 802-3026
corinna@sys-con.com

Events Manager
Lauren Orsi, 201 802-3022
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

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
Richard Walter, 201 802-3042
richard@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

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

```
var encoder : JPEGEncoder = new
JPEGEncoder(75);
var data : ByteArray = encoder.encode(
bd );
ws.WSsave( base64Encode( data ) );
```

If you want to save the binary data without using RemoteObjects or Web Services, you can always use a standard HTTP post method. In Flex, you'll have to create an instance of an HTTPService object, with the method set to "POST":

```
<mx:HTTPService
id="hs"
 showBusyCursor="true"
 useProxy="false"
 method="POST"
 resultFormat="text"
 url="/BinaryData/cf/HTTPImageSave.
cfm"
result="onResult('Data Saved via mx:
HTTPService')"
fault="onFault(event)" />
```

The file HTTPImageSave.cfm is actually very simple. The save occurs with only two lines of code:

```
<cfparam name="data" type="string"
default="">
<cffile action="write" file="c:\temp\
http_data.jpg" output="#ToBinary( data
)#" />
```

You'll notice in this case that the data is also being written to the file system using the toBinary method (to convert the Base64-encoded string back into binary data). When invoking the HTTPService, you'll have to create an object containing the parameters for the HTTPService and send it as the parameter of the HTTPService.send method (note "hs" is the id of the HTTPService instance):

```
var bd : BitmapData = getUIComponent-
BitmapData( paintCanvas );

var encoder : JPEGEncoder = new
JPEGEncoder(75);
var data : ByteArray = encoder.encode(
bd );
var params : Object = { data : bas-
e64Encode( data ) };
hs.send( params );
```

…and that is how you save binary image data using RemoteObjects, Web Services, or HTTPServices. This doesn't just apply to images. This applies to any kind of binary data; the only difference is that the image data first gets encoded to a JPG ByteArray. The AS3 corelib project on Adobe Labs/Google Code also includes class libraries that enable you to save data as PNG images instead of JPG images, so JPG isn't your only option.

A few things to keep in mind for real-world applications… It's best practice to use <CFFILE /> in a <CFTRY/> statement to catch any file system errors that may occur (not enough space, permissions, etc.). It's also best practice to use <CFLOCK /> with <CFFILE /> to prevent any errors due to data synchronization, threading access, or deadlock scenarios.

A working example and the source code from this article can be viewed online at http://www.cynergysystems.com/blogs/blogs/andrew.trice/binary_data_example/ or downloaded from http://www.cynergysystems.com/blogs/blogs/andrew.trice/binary_data_example/BinaryData.zip.

"Regardless of how you're sending the data to the server, **it's a good practice to compress the data client side whenever possible."**

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

Simplifying AJAX with the Spry Framework

Delivering great experiences while fitting the workflows and skill sets of both designers & developers

by Paul Gubbay

Over the past year the Dreamweaver product design and development team have been out on the road talking to Web designers and developers (Web pros) about the prospect of using AJAX to build more engaging and interactive Web pages and applications. While many Web pros were excited by what could be built, we also heard big concerns both about how to get started and how to integrate new AJAX frameworks into existing workflows.

- ***Takeaway:*** The technology is a means to an end. What resonates with Web pros is the ability to offer their customers richer and more engaging experiences. Top of mind is the ability to incorporate new technology into existing workflows.

  We took this to heart and set out to create a framework that leverages AJAX to deliver great experiences while fitting the workflows and skill sets of both designers and developers. The Spry framework for AJAX was developed with the following objectives:
- ***Keep it open:*** Spry works with all HTML editors, including Dreamweaver. Anyone can download it off Adobe Labs to start building high-performing interactive Web pages and applications. No proprietary tags or server-side code needed.
- ***Make it easy to use:*** Spry uses the same technologies that you already know to build Web pages (HTML, CSS, and JavaScript). Include a couple of JavaScript libraries with your page and you are ready to go. Create and style dynamic regions and interactive widgets using the same techniques as any other HTML elements on your page.
- ***Enable innovation:*** At the end of the day Spry is about delivering richer experiences for your customers. Spry provides a lightweight yet powerful model for adding data, interactivity, and rich UI widgets into your Web pages while putting you in complete control of the design.

Who Is It For?

Spry was built for both Web designers and developers. Before we built the framework we created a profile of the typical user for Spry. This profile helped us define our guiding principles.

- Works in Web production. Focused on the Web UI and x(HTML) production.
- Expert with x(HTML) and CSS.
- Familiar with JavaScript and the DOM.
- Cares about the quality of the code.
- Wants to create next-generation Web pages.

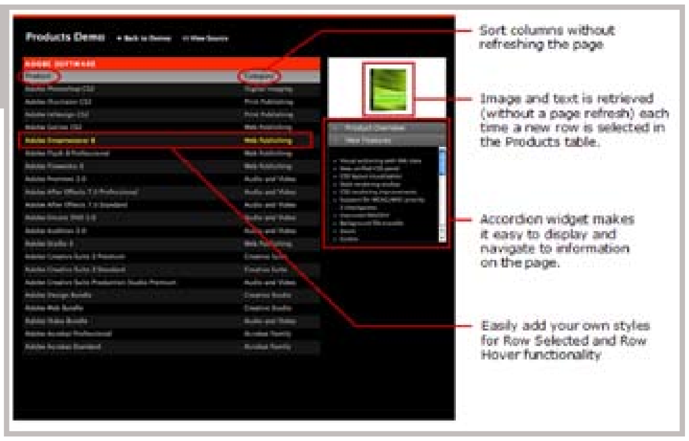
Guiding Principles

- Keep the framework familiar, lightweight and transparent.
- Keep the "framework" to a minimum (page-centric).
- Don't obscure the code.
- Make it feel like a natural extension to HTML.
- Integrate well with other technologies.

Enable a better designer-developer workflow

- Promote separation of design from data/content.
- Support "design-time XML."
- A framework easily leveraged within design-time tools

Figure 1


Products Demo
Sort columns without refreshing the page
Image and text is retrieved (without a page refresh) each time a new row is selected in the Products table.
Accordion widget makes it easy to display and navigate to information on the page.
Easily add your own styles for Row Selected and Row Hover functionality

Next-Generation Web UI OK

So what do we mean by more engaging and interactive Web pages? To get a first-hand experience you can check out some of the sample pages that have been built using Spry at Adobe Labs http://labs.adobe.com/technologies/spry/demos/index.html.

In the course of this article we're going to look at one page in detail and explore the code that was used to generate it.

Some of the characteristics of Spry-enabled Web pages include:

- Single page experience.
- Seamless interactivity/more responsive.
- Use of transitions and effects.
- More sophisticated UI elements.

The Spry Framework

Spry is a client-side framework in the form of JavaScript libraries that you can easily add to your new and existing Web pages. Spry is server- and tool-agnostic. You can use it with Dreamweaver or your HTML editor of choice. Just a few lines of code can produce powerful results. More importantly, it should all look familiar. You should be able to jump in and be productive right away. Spry does the heavy lifting.

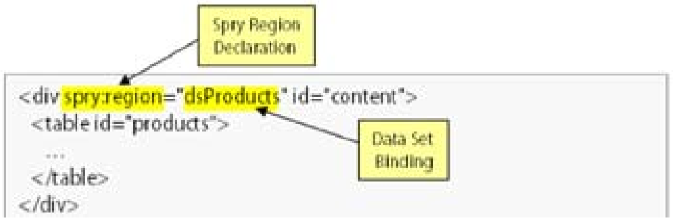
The Spry JavaScript libraries host three modules (see Figure 2).

Spry Data

Data is accessed and displayed using Spry Data Sets and Regions. The Spry Data Set is a JavaScript object that is responsible for loading and managing (e.g., sort, filter) data. The Data Set base class is architected so that multiple flavors can be built to access data from different sources (see Figure 3).

Spry Regions

A Spry dynamic Region is an area on a Web page that's bound to a Data Set. When a Data Set is modified (e.g., loading data off the server, filtering, sorting), the Spry Region is updated to reflect the new data. A Spry Region can be created for any HTML block element such as a <div> or a <p> tag.



```
<div spry:region="dsProducts" id="content">
  <table id="products">
  ...
  </table>
</div>
```

If we take a look at our example we can see that the Products table and Product image are contained within Spry Regions that are bound to a Spry XML Data Set. When the data changes within the Data Set, both regions automatically update themselves to reflect the new data.

Loading Data

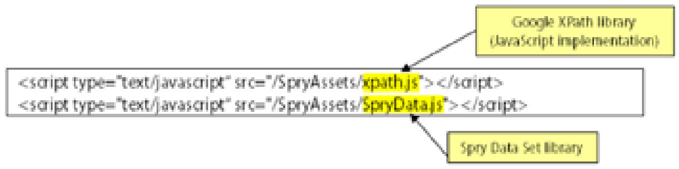
The Spry Data Set is responsible for loading and processing data behind the scenes (see Figure 4). XML is frequently used to transfer data from the server to the client. The XML Data Set in Spry can be used to retrieve this data through the browser via the XMLHTTPRequest object. The XML can be contained in a file or returned from a server-side function call (e.g., PHP, ColdFusion, ASP).

Once the XML data is retrieved it's flattened into a standard record/field format to make it easier for users to bind the data to Spry Regions in the Web page. Spry uses XPath, a W3C standard for describing a set of nodes in XML, to identify the nodes(s) that represent a record of data.

In Figure 5 you can see a sample XML file (products.xml) that represents the products in our table. Using XPath (products/product) we identify the node(s) in the XML file that represent the records of data for our Data Set. Finally, we can see the records and fields in the Spry Data Set once the conversion is complete.

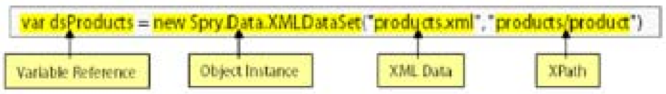
Data Set Declaration

Spry Data Sets are added to your page in two steps. First, you must include the appropriate JavaScript libraries:



```
<script type="text/javascript" src="/SpryAssets/xpath.js"></script>
<script type="text/javascript" src="/SpryAssets/SpryData.js"></script>
```

Second, create an instance of the Data Set by passing in the location to the XML data and defining the repeating node that represents a record of the data. This code is added to the head of the document.



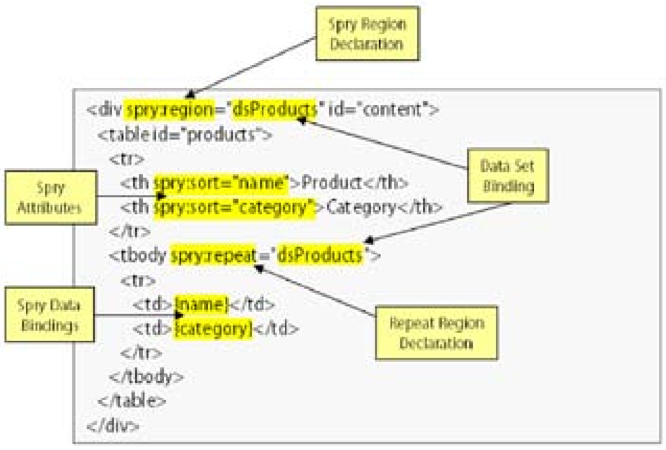
```
var dsProducts = new Spry.Data.XMLDataSet("products.xml","products/product")
```

- ***Variable reference:*** Name of the Data Set. Referenced through data bindings or JavaScript.
- ***Object Instance:*** Creates an instance of the Data Set object using the "new" keyword.
- ***XML Data:*** Pointer to the XML file or server-side function that returns XML.
- ***XPath:*** Pointer to the node in the XML file that represents a record of the data.

Displaying Data

Data is displayed on the Web page using Spry Regions that are loaded with live data when the Web page is displayed in the browser. Creating the bindings from the Data Set to the Spry Regions is easily accomplished by attaching Spry attributes to your standard HTML tags.

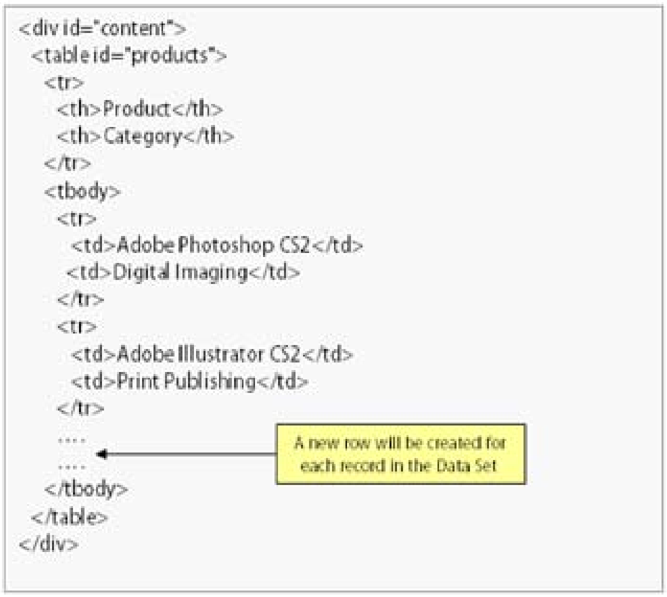
In the following example we can see how the Spry Region for the Products table is built:



```
<div spry:region="dsProducts" id="content">
  <table id="products">
    <tr>
      <th spry:sort="name">Product</th>
      <th spry:sort="category">Category</th>
    </tr>
    <tbody spry:repeat="dsProducts">
      <tr>
        <td>{name}</td>
        <td>{category}</td>
      </tr>
    </tbody>
  </table>
</div>
```

- ***Spry Region Declaration:*** Defines an HTML element as a Spry region.
- ***Data Set Binding:*** Binds a Spry Region to a Data Set.
- ***Spry Attributes:*** Specialized attributes that provide additional functionality such as sorting.
- ***Spry Data Bindings:*** Bindings to fields located in the Data Set.
- ***Repeat Region Declaration:*** Repeats a Spry Region once for each record in the Data Set.

Assuming that we're using the XML sample from our previous example this code snippet creates the following HTML:



```
<div id="content">
  <table id="products">
    <tr>
      <th>Product</th>
      <th>Category</th>
    </tr>
    <tbody>
      <tr>
        <td>Adobe Photoshop CS2</td>
        <td>Digital Imaging</td>
      </tr>
      <tr>
        <td>Adobe Illustrator CS2</td>
        <td>Print Publishing</td>
      </tr>
      ....
      ....
    </tbody>
  </table>
</div>
```

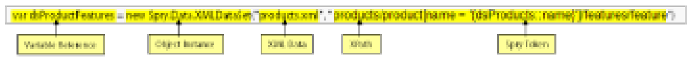
Master/Detail Data Sets

The Spry framework supports the concept of Master/Detail Data Sets. This implies that the selected record in a Master Data Set drives the contents of the Detail Data Set. There are many ways to create this relationship using Spry. In our example the Master Data Set and the Detail Data Set retrieves data from different sections of the same XML file. Each time a new record is selected in the Master Data Set, the Detail Data Set retrieves a subset of the data.

The Master Data Set is used to retrieve the products for the Products table. The Detail Data Set is used to retrieve the product features once a specific product is selected in the Product table.

Detail Data Set Declaration

The Detail Data Set and its relationship to the Master Data Set can be set up easily using one additional line of JavaScript:

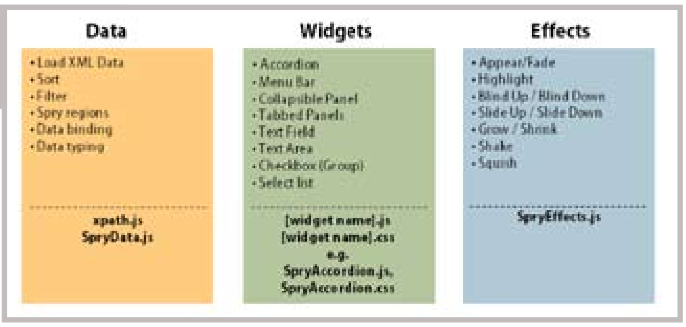


```
var dsProductFeatures = new Spry.Data.XMLDataSet("products.xml", "products/product[name = '{dsProducts::name}']/features/feature")
```

Note that the main difference in the Detail Data Set declaration is the XPath. This XPath includes a filter expression and uses a Spry Token. The filter expression limits the nodes that will be returned for the dsProductFeatures Detail Data Set. The Spry token represents the "name" field for the current record in the dsProducts Master Data Set.

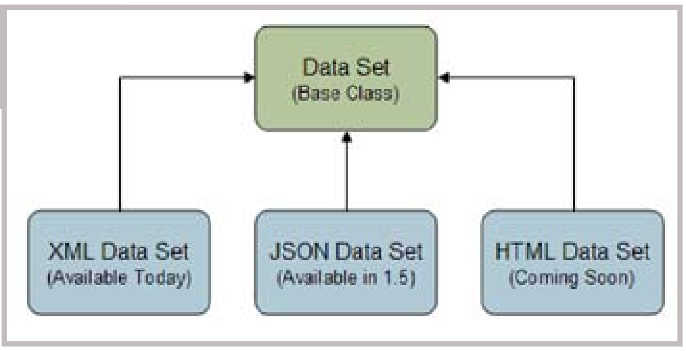
In effect the declaration states that the dsProductFeatures Detail Data Set will fetch its data based on the value of the "name" field in the current

Figure 2



Data	Widgets	Effects
• Load XML Data • Sort • Filter • Spry regions • Data binding • Data typing	• Accordion • Menu Bar • Collapsible Panel • Tabbed Panels • Text Field • Text Area • Checkbox (Group) • Select list	• Appear/Fade • Highlight • Blind Up / Blind Down • Slide Up / Slide Down • Grow / Shrink • Shake • Squish
xpath.js **SpryData.js**	**[widget name].js** **[widget name].css** **e.g.** **SpryAccordion.js,** **SpryAccordion.css**	**SpryEffects.js**

Figure 3

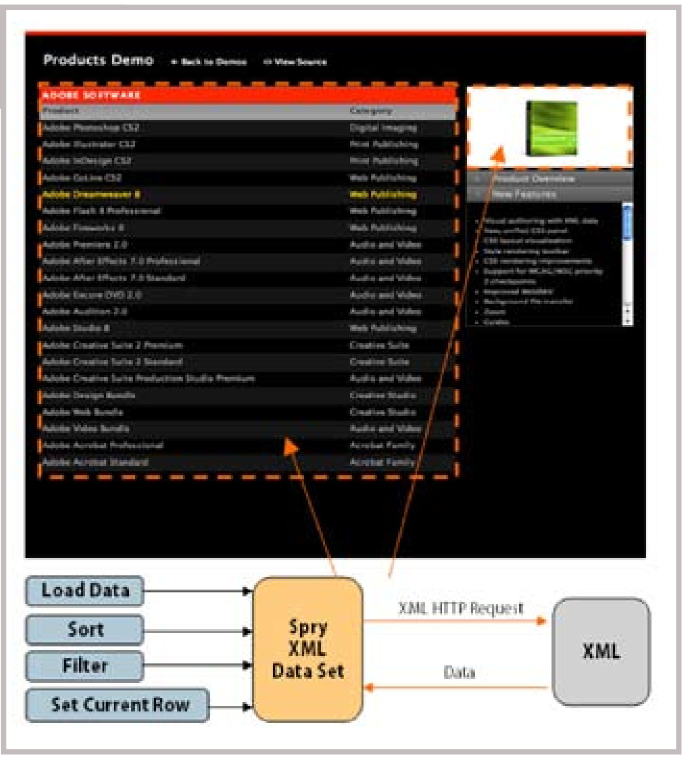


record of the dsProducts Master Data Set (see Figure 6). Every time the user selects a new record in the Master Data Set, this Detail Data Set will automatically update.

Master/Detail Regions

In Figure 7 the dsProductFeatures Detail Data Set is used to display the features of a selected product in the Accordion widget. When the user chooses a new product in the Product table, the features will change in the Accordion without needing a page refresh.

Figure 4



Spry Widgets

Spry widgets are UI elements that make it easier for your users to interact with the content on the page. The current set of widgets (so far) includes:

- Accordion
- Menu Bar
- Collapsible Panel
- Tabbed Panels
- Text Field
- Text Area
- Checkbox & Checkbox Group
- Select list

The Widget philosophy follows the Spry guiding principles:

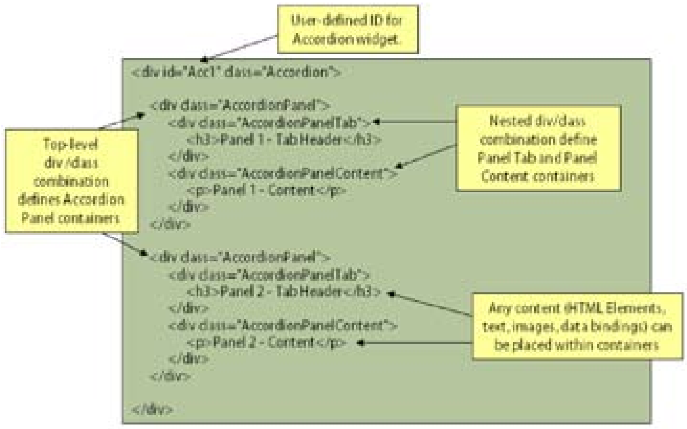
- Widgets must be easy to modify (re-style).
- Code must be easy to understand.
- HTML-centric markup.
- No custom tags.
- No programmatic injection of markup.
- Self-contained (limit dependencies on additional libraries).

Anatomy of a Widget

Each widget is broken down into three modules as in Figure 8.

Widget Structure

In our example we use an Accordion widget to display additional information about the selected product in the Products table. The structure of the widget is defined by the HTML code.



```
<div id="Acc1" class="Accordion">

  <div class="AccordionPanel">
    <div class="AccordionPanelTab">
      <h3>Panel 1 - TabHeader</h3>
    </div>
    <div class="AccordionPanelContent">
      <p>Panel 1 - Content</p>
    </div>
  </div>

  <div class="AccordionPanel">
    <div class="AccordionPanelTab">
      <h3>Panel 2 - TabHeader</h3>
    </div>
    <div class="AccordionPanelContent">
      <p>Panel 2 - Content</p>
    </div>
  </div>

</div>
```

User-defined ID for Accordion widget.
Top-level div/class combination defines Accordion Panel containers
Nested div/class combination define Panel Tab and Panel Content containers
Any content (HTML Elements, text, images, data bindings) can be placed within containers

As you can see, the composition of the widget is quite easy to understand for anybody familiar with HTML. If you want to add or delete a panel, you can simply add or delete the appropriate HTML container in your code.

Widget Behavior

The functionality for the Accordion widget is built into a JavaScript file that understands the HTML structure for the widget. This approach makes it easy for users to modify the look-and-feel of the widget without having to worry about its functionality. As long as the HTML structure is correct, the widget will function as expected. The JavaScript is included using two lines of code.

The first line simply links in the JavaScript file that contains the Accordion functionality.

```
<script type="text/javascript" src="/widgets/accordion/SpryAccordion.js"></script>
```

The second line creates an instance of the Accordion JavaScript object and links it to the HTML by passing in the user-defined ID (Acc1).

```
<script type="text/javascript">
  var acc1 = new Spry.Widget.Accordion("Acc1");
</script>
```

Note that the JavaScript to create an instance of a Spry widget must be included under your markup at the bottom of the Web page.

Widget Styling

The Accordion widget is styled just as you would any other HTML element on your page using CSS. The CSS file (SpryAccordion.css) is included using the following line of code:

```
<link href="css/SpryAccordion.css" rel="stylesheet" type="text/css" />
```

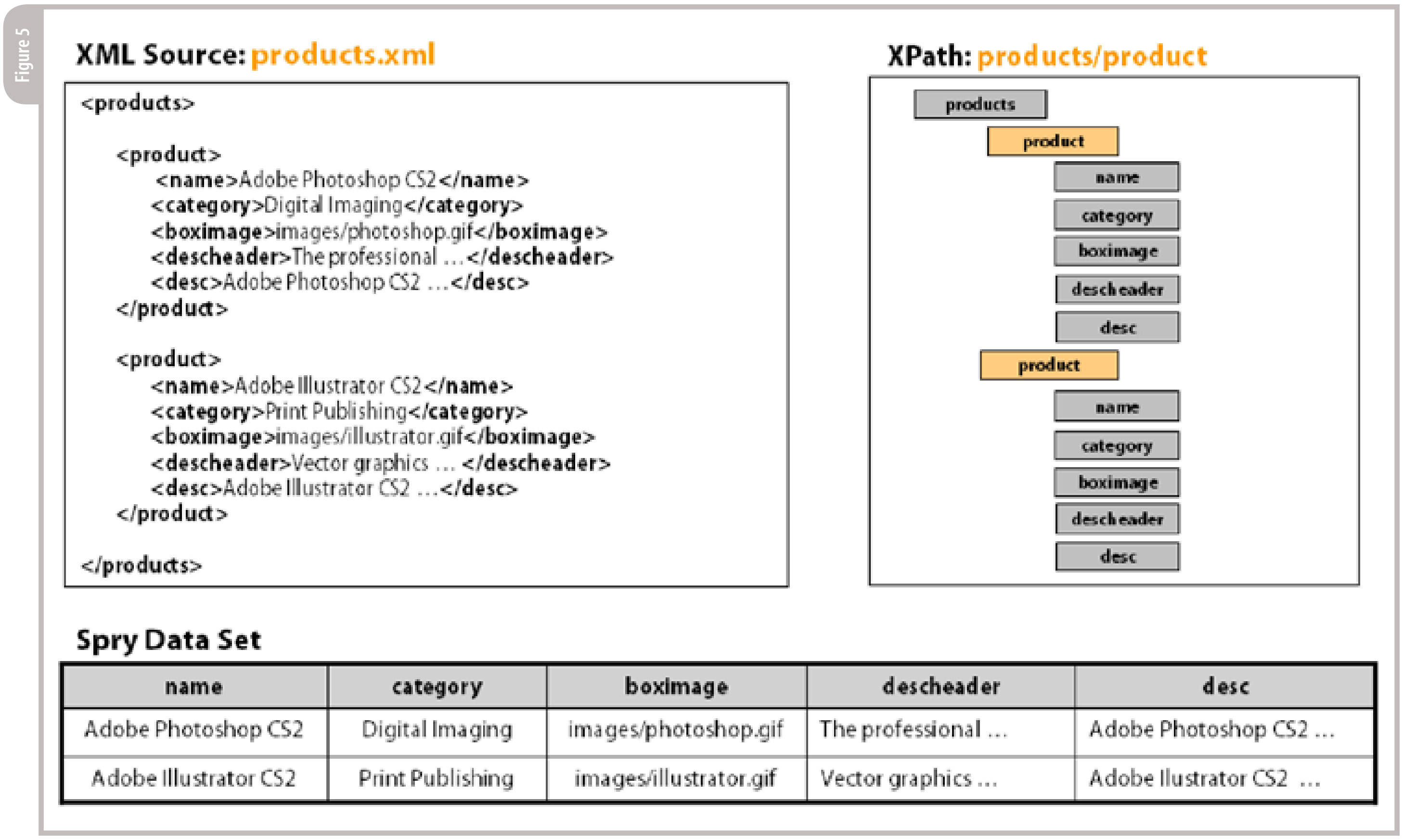
In the HTML example in Figure 8 you will notice that there are CSS classes applied to each of the relevant block-level elements (Accordion, AccordionPanel, AccordionPanelTab, and AccordionPanelContent). The CSS rules for these classes are defined in the SpryAccordion.css file. They can be changed at will by the Web developer or designer.

Spry Effects

Spry effects are visual enhancements that you can apply to almost any element on an HTML page. For example, an effect might highlight information, create animated transitions, or visually alter a page element for a certain period of time. Effects are a simple but elegant way of enhancing the look-and-feel of your Web site.

Paul Gubbay is a director of engineering at Macromedia. Previously, Paul held the role of CEO at CyberSage Software, where he spent several years building the vision and infrastructure of the company. Under Paul's guidance, CyberSage focused on emerging technologies such as XML, Java, and Macromedia Flash to deliver leading edge product offerings.
pgubbay@adobe.com

Figure 5



XML Source: products.xml

```
<products>

    <product>
        <name>Adobe Photoshop CS2</name>
        <category>Digital Imaging</category>
        <boximage>images/photoshop.gif</boximage>
        <descheader>The professional ...</descheader>
        <desc>Adobe Photoshop CS2 ...</desc>
    </product>

    <product>
        <name>Adobe Illustrator CS2</name>
        <category>Print Publishing</category>
        <boximage>images/illustrator.gif</boximage>
        <descheader>Vector graphics ... </descheader>
        <desc>Adobe Illustrator CS2 ...</desc>
    </product>

</products>
```

Spry Data Set

name	category	boximage	descheader	desc
Adobe Photoshop CS2	Digital Imaging	images/photoshop.gif	The professional ...	Adobe Photoshop CS2 ...
Adobe Illustrator CS2	Print Publishing	images/illustrator.gif	Vector graphics ...	Adobe Illustrator CS2 ...

The current set of effects (so far) includes:

- Appear/Fade
- Highlight
- Blind Up/Blind Down
- Slide Up/Slide Down
- Grow/Shrink
- Shake
- Squish

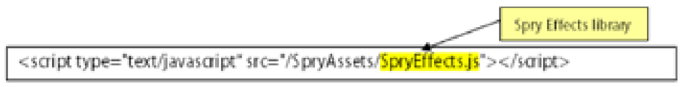
The Effect philosophy follows the Spry guiding principles:

- Simple JS Syntax.
- Single JS include.
- Compatible with other effect libs/frameworks.
- Focus on key interaction/UE behaviors.

“Our goal in putting Spry on Adobe Labs was to make it an **open technology that could be driven by customer feedback”**

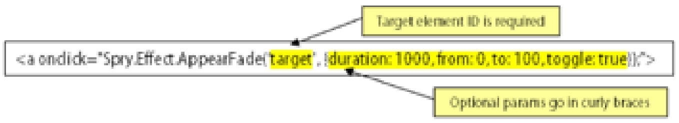
Implementing an Effect

Spry Effects are added to your page in two steps. First, you must include the appropriate JavaScript library:



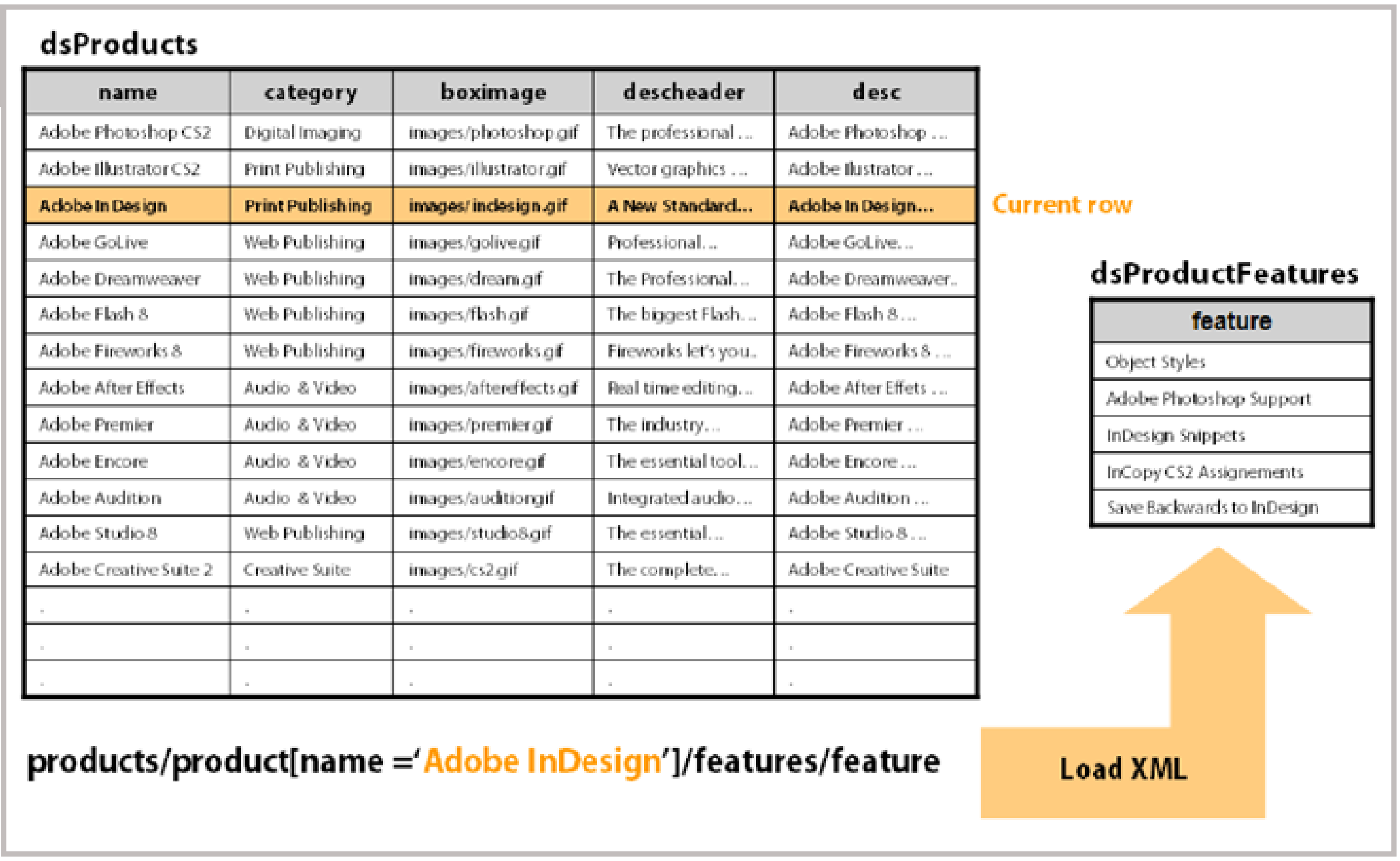
```
<script type="text/javascript" src="/SpryAssets/SpryEffects.js"></script>
```

Second, add an event handler to a supported element (e.g., <a> tag). Note: Supported elements differ by effect – please refer to the Spry API docs.


Target element ID is required
Optional params go in curly braces

```
<a onclick="Spry.Effect.AppearFade('target', {duration: 1000, from: 0, to: 100, toggle: true});">
```

Figure 6



dsProducts

name	category	boximage	descheader	desc
Adobe Photoshop CS2	Digital Imaging	images/photoshop.gif	The professional ...	Adobe Photoshop ...
Adobe Illustrator CS2	Print Publishing	images/illustrator.gif	Vector graphics ...	Adobe Illustrator ...
Adobe InDesign	**Print Publishing**	**images/indesign.gif**	**A New Standard...**	**Adobe InDesign...**
Adobe GoLive	Web Publishing	images/golive.gif	Professional...	Adobe GoLive...
Adobe Dreamweaver	Web Publishing	images/dream.gif	The Professional...	Adobe Dreamweaver..
Adobe Flash 8	Web Publishing	images/flash.gif	The biggest Flash...	Adobe Flash 8 ...
Adobe Fireworks 8	Web Publishing	images/fireworks.gif	Fireworks let's you..	Adobe Fireworks 8 ...
Adobe After Effects	Audio & Video	images/aftereffects.gif	Real time editing...	Adobe After Effets ...
Adobe Premier	Audio & Video	images/premier.gif	The industry...	Adobe Premier ...
Adobe Encore	Audio & Video	images/encore.gif	The essential tool...	Adobe Encore ...
Adobe Audition	Audio & Video	images/audition.gif	Integrated audio...	Adobe Audition ...
Adobe Studio 8	Web Publishing	images/studio8.gif	The essential...	Adobe Studio 8 ...
Adobe Creative Suite 2	Creative Suite	images/cs2.gif	The complete...	Adobe Creative Suite
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

dsProductFeatures

feature
Object Styles
Adobe Photoshop Support
InDesign Snippets
InCopy CS2 Assignements
Save Backwards to InDesign

products/product[name ='Adobe InDesign']/features/feature

For each effect, you must set the target element to which you want to apply the effect. The element parameter can be either a string containing the id of the element or a JavaScript DOM element object. In the example above the AppearFade effect will run on the "target" element when the user clicks on the <a> link. If the target element is a <div id="products"> tag, you would pass in "products" as the target parameter.

Effect Composition

Spry effects are created by combining core effects. The core effects include:

- Move
- MoveSlide
- Size
- Opacity
- Color

The combined effects of the Spry framework include:

- Highlight
- Grow/Shrink
- Shake
- Squish

Spry supports the ability to cluster effects together to create your own variations. This is done by using the Cluster effect that acts like a container for multiple core effects. Like a regular effect, the Cluster effect has a start() function. But unlike regular effects, Cluster provides the methods addNextEffect() and addParallelEffect(). These functions let you chain effects together to create new variations.

Clustered effects are a flexible mechanism for creating new effect variations (see Figure 9). However, they require some JavaScript programming knowledge. Please refer to the Spry API docs for more information.

What's Next for Spry?

Our goal in putting Spry on Adobe Labs was to make it an open technology that could be driven by customer feedback. This method has produced great results for our engineering team and our customers. We continue to use your feedback on the forums to help guide Spry's future direction. Spry is currently in beta with official release scheduled for early spring, 2007. This new release will include support for JSON and nested Data Sets.

Besides the official release of Spry on Adobe Labs, we'll be shipping a new version of Dreamweaver with support for Spry Data Sets, Widgets, and Effects.

Spry is a free download and can be accessed on Adobe Labs using http://labs.adobe.com/technologies/spry/.

Figure 7

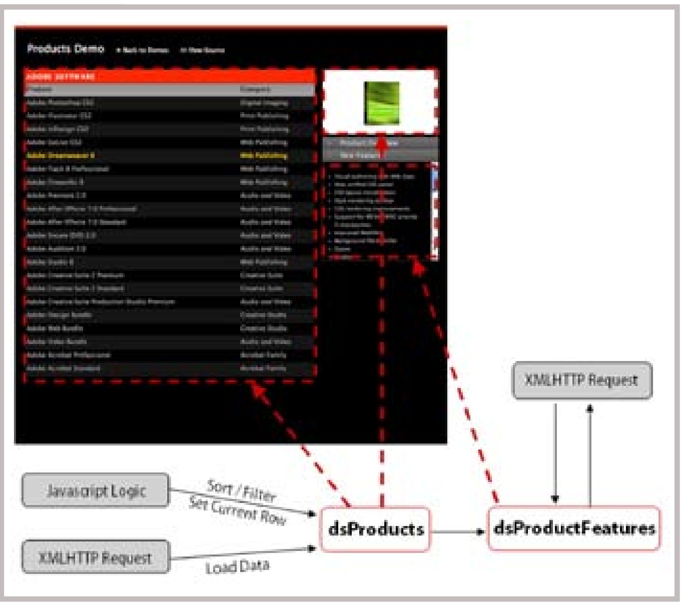
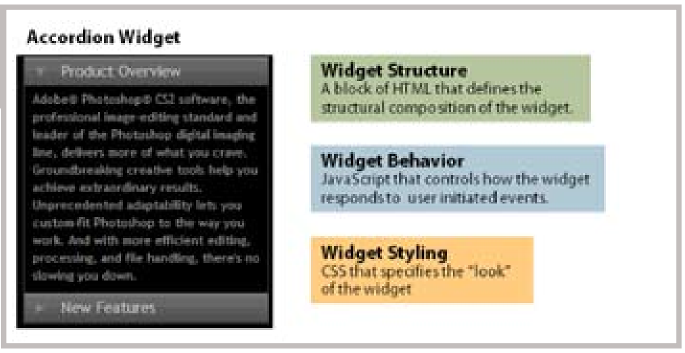
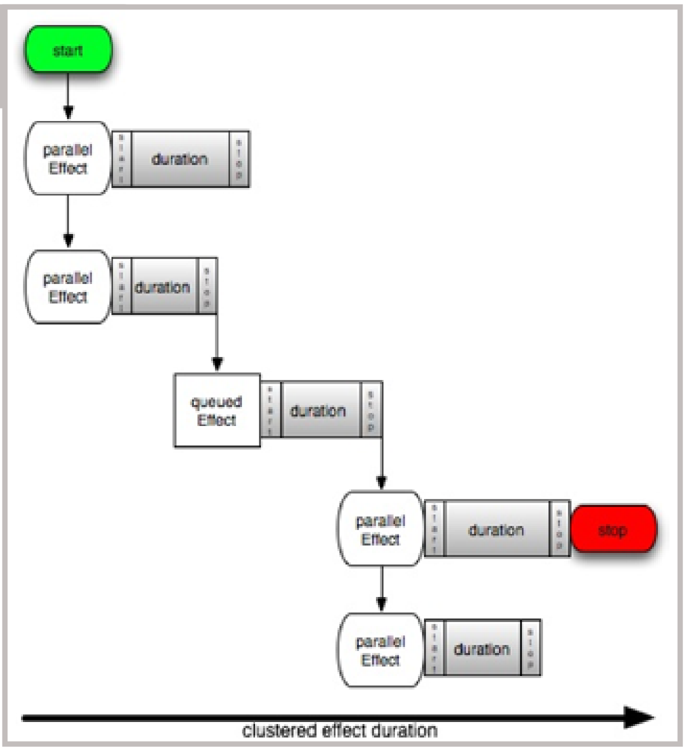


Figure 8


Accordion Widget
Product Overview
New Features
Widget Structure
A block of HTML that defines the structural composition of the widget.
Widget Behavior
JavaScript that controls how the widget responds to user initiated events.
Widget Styling
CSS that specifies the "look" of the widget

Adobe® Photoshop® CS2 software, the professional image-editing standard and leader of the Photoshop digital imaging line, delivers more of what you crave. Groundbreaking creative tools help you achieve extraordinary results. Unprecedented adaptability lets you custom-fit Photoshop to the way you work. And with more efficient editing, processing, and file handling, there's no slowing you down.

Figure 9



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

Flex 2 Metadata Tags

Telling the compiler how to compile

by Rich Tretola

Rich Tretola is a senior software developer at Herff Jones, Inc., specializing in Rich Internet Applications. He is an award-winning Flex developer and self-proclaimed Flex evangelist. Rich is highly regarded in the Flex community as an expert in RIA and is an Adobe Community Expert. He runs a popular Flex blog at HYPERLINK "http://www.everythingFlex.com" http://www.everythingFlex.com, contributes to the Indianapolis Flex user group, and is the lead author of the Wiley/Wrox Publication Professional Flex 2.

most Flex developers have seen and used the [Bindable] tag but not many know what this tag does or even what it is. [Bindable] is what is known as a metadata tag. Metadata tags are special tags that are inserted in your source code that give the compiler information on how to compile the application. These tags aren't actually compiled into the generated SWF file but provide instructions on how the compiler should create the SWF. There are 12 documented metadata tags. This article will offer definitions of these metadata tags and give examples of their use. By the end of this article you will understand when and where to use metadata tags in your Flex 2 applications.

[ArrayElementType]

Defining an array is usually very generic in nature because the elements of the array can be any type. However, using the ArrayElementType metadata tag lets you define the data types of the array elements. Here is the sample syntax of [ArrayElementType]:

```
[ArrayElementType(“String”)]
public var arrayOfStrings:Array;

[ArrayElementType(“Number”)]
public var arrayOfNumbers:Array;

[ArrayElementType(“mx.core.
UIComponent”)]
    public var arrayOfUIComponents:
Array;
```

[Bindable]

The Bindable metadata tag is one of the most used metadata tags because it allows for easy data synchronization within the components of your application. Bindable can be used to bind simple data, classes, complex data, and functions. To bind a simple piece of data, you must simply define the data with the metadata tag included, as shown in Listing 1. Figure 1 shows the results of Listing 1

Bindable also allows binding to events. Listing 2 shows how to bind a property using getters and setters, along with an event binding. This example has a private variable named phoneNumber, as well as a public getter and setter. The getter method uses the Bindable tag to bind to an event named phoneNumberChanged. This setter method dispatches the phoneNumberChanged even whenever its data changes. By using a setter method, the data can be manipulated before it's set to the private variable. In this example, the data is formatted only when the length of the value coming into the method is 10. When the phoneNumberChanged event is dispatched, the second TextInput component updates because its text property is bound to the phoneNumber variable.

Figures 2 and 3 show the results of Listing 2.

[DefaultProperty]

The DefaultProperty metadata tag is used to set a single property as a default property of a class. This allows the property to be set within a container tag without needing to define the property name. A simple example of this would be a custom Button class. Listing 3 shows a simple Button class that has the label property set as the DefaultProperty. Listing 4 shows how the label is defined as a string within the custom Button container tags.

[Embed]

The Embed metadata tag is used to import images into your application. There are two ways to use Embed. You can either embed the image in ActionScript and assign it to a variable (as in the first example in the following code), or you can assign it directly to the component property (using the syntax shown in the second example of the following code).

```
[Embed(source=”myIcon.gif”)]
[Bindable]
public var myIcon:Class;

<mx:Button label=”Icon Button 1”
icon=”{myIcon}”/>

<mx:Button label=”Icon Button 2”
icon=”{myIcon}”/>
```

The output from the following is identical to the previous code block. The benefits of creating the myIcon class are that it can be defined one time in a single class and bound to multiple components in your application.

```
<mx:Button label="Icon Button 1"
icon="@Embed(source=myIcon.gif')"/>

<mx:Button label="Icon Button 2"
icon="@Embed(source=myIcon.gif')"/>
```

[Event]

The Event metadata tag is used to declare events that will be dispatched by your custom class. Adding this metadata tag to your class definition allows you to add an event handler function to the MXML tag used to instantiate your custom class. Listing 5 creates a custom Button class that will dispatch an event whenever its label property changes. The main application file shown in Listing 6 instantiates the custom Button and creates the event handler function, which dumps the new label property to a TextArea component to show the occurring changes.

Figure 4 shows the results of Listing 5 and Listing 6.

[Effect]

The Effect metadata tag is used to define a custom effect that will be dispatched when an event occurs. This can be easily demonstrated by building on the earlier Event examples. By simply changing a single line to the ButtonLabel class (Listing 7), an effect is defined that can be assigned to an Effect instance (Listing 8).

[IconFile]

IconFile is used to identify the filename of a jpg, gif, or png file that will be used as the icon for your custom class. While the [Embed] meta tag can be used to embed images files, SWF files, music files, video files, etc, IconFile is only used to embed a file that will be used as the icon for the custom class. Here is the example of the IconFile syntax:

```
[IconFile("icon.png")]
public class CustomButton extends
Button
{

}
```

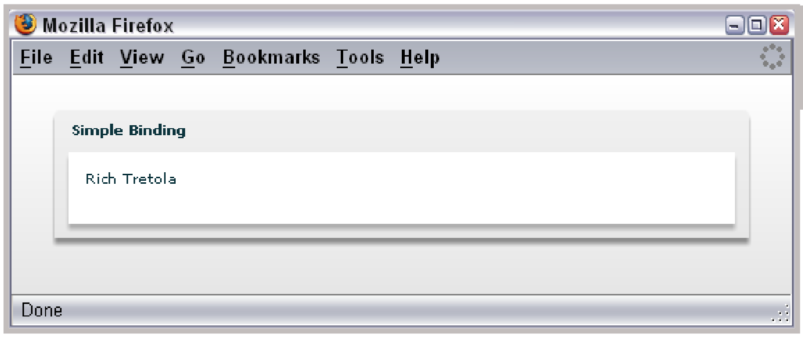
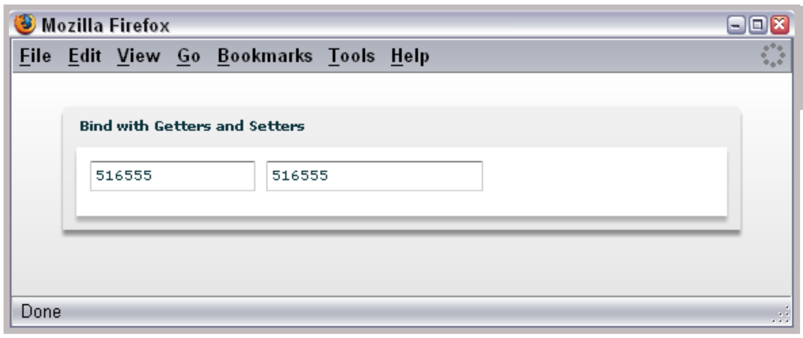
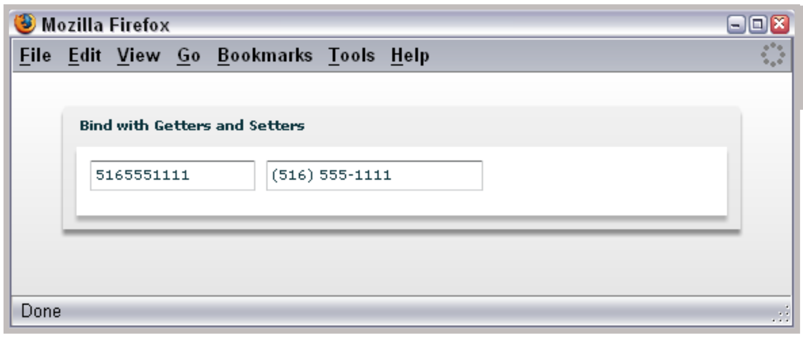
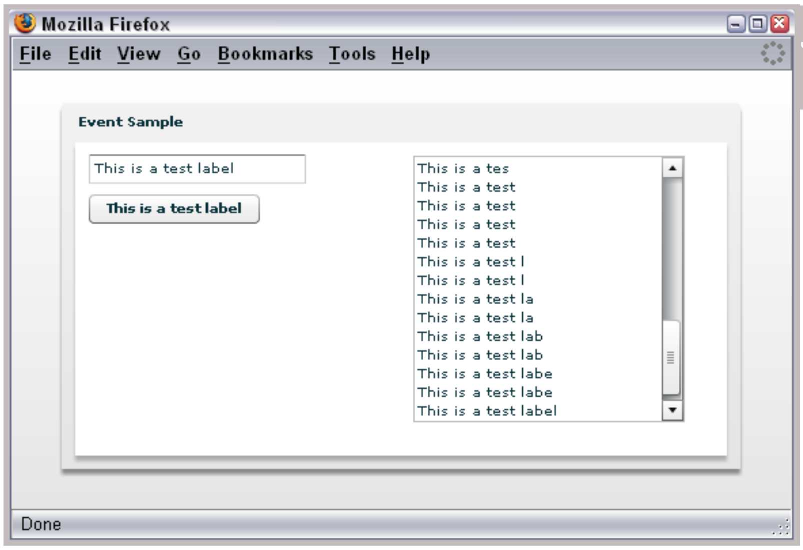


Figure 1


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Bind with Getters and Setters
516555
516555
Done
Figure 2


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Bind with Getters and Setters
5165551111
(516) 555-1111
Done
Figure 3



```
This is a tes
This is a test
This is a test
This is a test
This is a test
This is a test l
This is a test l
This is a test la
This is a test la
This is a test lab
This is a test lab
This is a test labe
This is a test labe
This is a test label
```

Figure 4

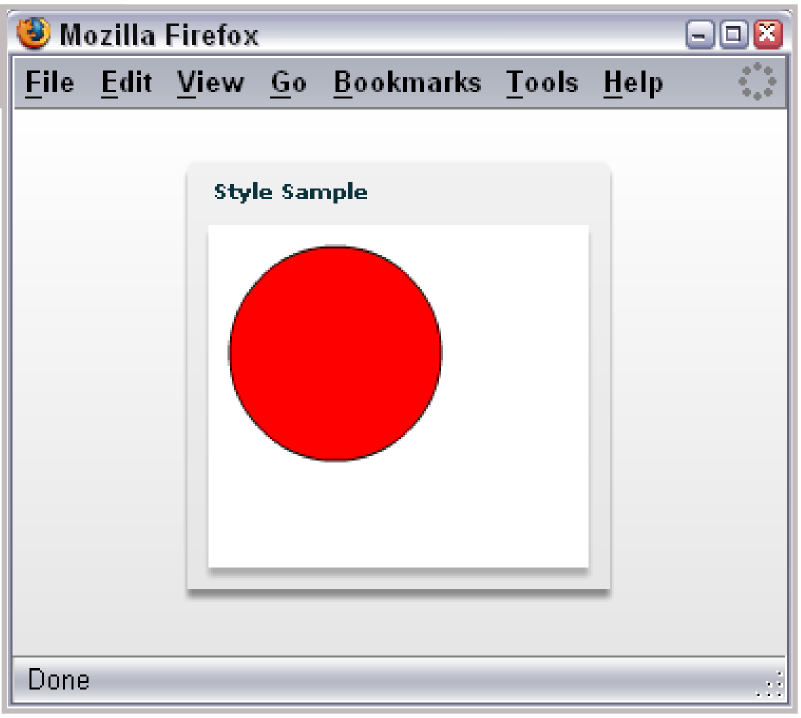
[Inspectable]

The Inspectable metadata tag is used to define the attributes of your custom component that you would like to display in the code hints and property inspector of Flex Builder 2. The example shown in Listing 9 defines a variable named ccType that is inspectable. It defines a defaultValue of Visa, a category of Credit Card and enumeration values of Visa, Mastercard, Discover, and American Express.

Figure 5 shows the example of the code hints being displayed as the component is added to an application.

Figure 6 shows the same example, but this time in design view, which exposes the property inspector. You can see that the category of properties is Credit Card with the property showing as ccType and the available values in the drop-down.

Figure 6



[InstanceType]

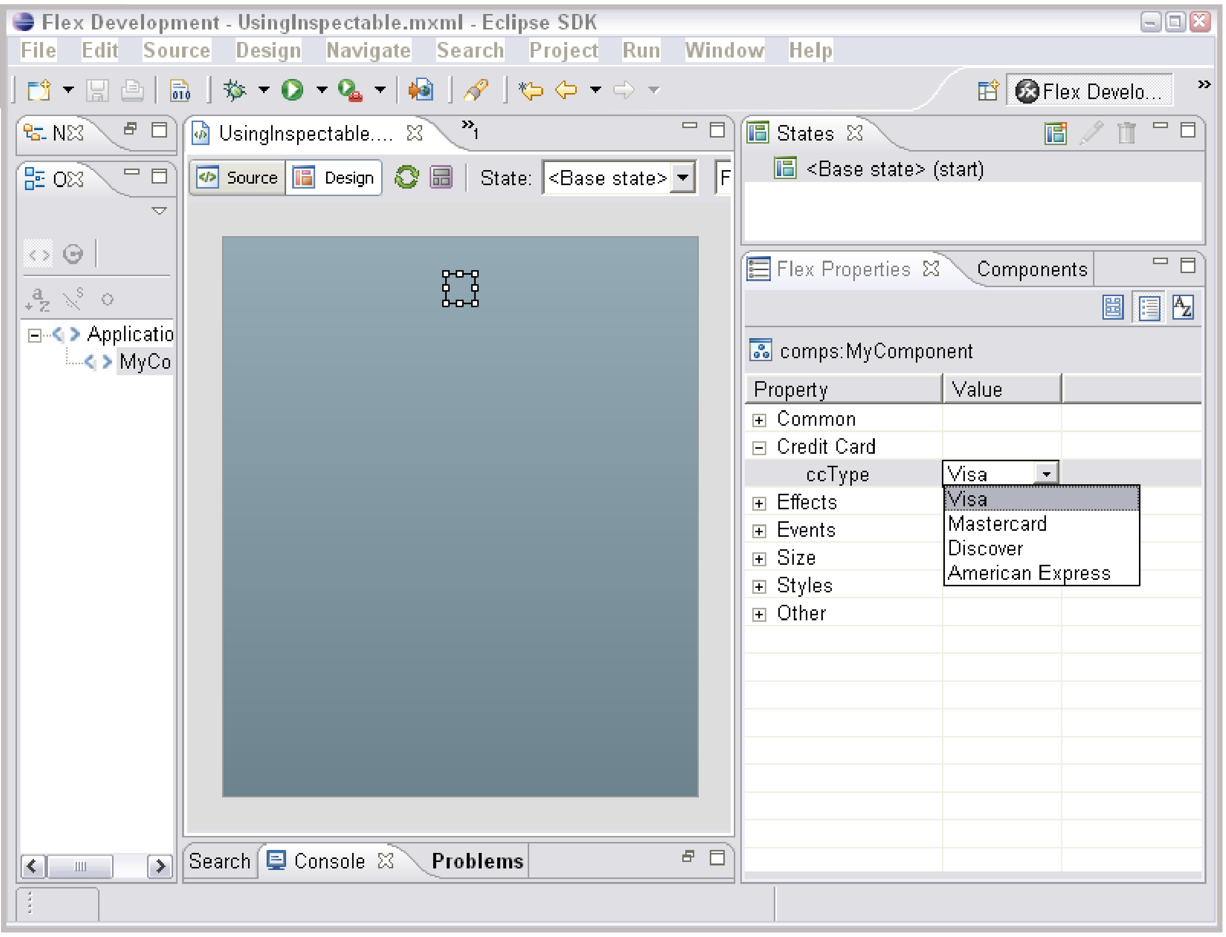
The InstanceType metadata tag is used to declare the type of object that will be allowed when declaring a variable as IDeferredInstance in a template object. The syntax of InstanceType looks like this:

```
[InstanceType("package.className")]
```

[NonCommittingChangeEvent]

The NonCommittingChangeEvent is a metadata tag that will prevent a change from occurring when a specified event occurs. Listing 10 demonstrates how this works. A private variable named s of type String is created and bound to the ti2 TextInput component. The other TextInput component with the id of ti1 sets the value of s equal to the value of its text property whenever the text changes. Additionally, the Binding metadata tag attached to the s variable is set to bind when the triggerBinding event is dispatched. The trig-

Figure 5


Flex Development - UsingInspectable.mxml - Eclipse SDK
File Edit Source Design Navigate Search Project Run Window Help
Flex Develo...
UsingInspectable....
Source Design State: <Base state>
States
<Base state> (start)
Applicatio
MyCo
Flex Properties Components
comps:MyComponent
Search Console Problems

Property	Value
Common	
Credit Card	
ccType	Visa
Effects	
Events	
Size	
Styles	
Other	

Visa
Mastercard
Discover
American Express

"Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises

— Yafim Natis, Gartner Analyst

3-DAY EVENT!

SOAWorld 2007 Plus Enterprise OpenSource Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services	Open Source
› AJAX and SOA	› Open Source Business Models
› Web 2.0	› Open Source ESB
› Universal SOA	› OpenAjax Alliance
› Protecting Web Services	› SaaS and Open Source
› Troubleshooting SOA	› Spring, Hibernate and Eclipse
› Governance	› Seam
› Open-Source SOA	› Open Source Penetration
› XBRL	› Monetizing Open Source
› Service Virtualization	› Open Source Databases
	› AMQP
	› Open Source Middleware

June 25-27, 2007

Roosevelt Hotel / New York City

Register Online! www.SOAWorld2007.com

SOAEOSCONFERENCE.SYS-CON.COM

REGISTER ONLINE TODAY

SAVE $200!

(HURRY FOR EARLY-BIRD DISCOUNT)

11th International 2007 SOAWORLD™ CONFERENCE & EXPO

2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-27, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

BROUGHT TO YOU BY:

› SOA World Magazine focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.

› Enterprise Open Source Magazine EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Exhibit and Sponsorship Info:

Call 201-802-3020 or email events@sys-con.com

gerBinding event is dispatched only when the Enter key is pressed while typing in the ti1 TextInput component.

[RemoteClass]

RemoteClass can be used to bind an ActionScript class to a Java class or a ColdFusion CFC. This will allow for automatic data type conversions. Below is a sample of an ActionScript class named MyClass in the package com.mydomain being bound to a Java class named MyClass in the package com.mydomain:

```
package com.mydomain {
    [Bindable]
    [RemoteClass(alias="com.mydomain.MyClass")]
    public class MyClass {
        public var id:int;

        public var myText:String;

    }
}
```

[Style]

The Style metadata tag is used to define custom style properties for your components. Simply add the Style metadata tag or tags to your class definition and then use the getStyle method to retrieve its value.

Listings 11 and 12 give examples of how to define two styles named borderColor and fillColor, both of which are defined as a uint data type. The styles are set in the component tag when the class is instantiated. The updateDisplayList function is overridden and the custom styles are used to draw the circle border and fill.

Figure 7 shows the results of Listing 12 and Listing 13.

By now you should have had a few "Wow, I know where I could have used that" or "Hmm, I think I will try this metadata tag in a new project." If you haven't, then you need to go back and read the article again. OK, so what I'm trying to say is that the metadata tags provided for us by the Adobe Flex team are not only extremely powerful, allowing us to extend and customize what we do with Flex, but are also very easy to use. They are a very quick way to accomplish a great deal with only a few lines of code. If you're not using these tags, you're working too hard to accomplish things that are built into Flex 2.

• • •

The content of this article is excerpted from the upcoming book titled *Professional Flex 2* (ISBN 0470102675) by Rich Tretola, Simon Barber, and Renaun Erickson from Wiley Publishing, Inc./Wrox Press. To pre-order please visit http://www.everythingflex.com or http://www.amazon.com/Professional-Flex-2-Rich-Tretola/dp/0470102675.

Listing 1 A simple use of [Bindable]

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    backgroundColor="#FFFFFF">
  <mx:Script>
    <![CDATA[
      [Bindable]
      private var me:String="Rich Tretola";
    ]]>
  </mx:Script>
  <mx:Panel title="Simple Binding"  width="500" height="90"
    paddingTop="10" paddingLeft="10" paddingRight="10" pad-
dingBottom="10"
    layout="horizontal">
      <mx:Label text="{me}"/>
  </mx:Panel>
</mx:Application>
```

Listing 2 Using [Bindable] with getters and setters

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">

  <mx:Script>
    <![CDATA[
    private var _phoneNumber:String = "";
    // Bind getter function to phoneNumberChanged event
    [Bindable(event="phoneNumberChanged")]
    public function get phoneNumber():String {
      return _phoneNumber;
    }
    // Setter method.
    public function set phoneNumber(value:String):void {
      if (value.length<10) {
        _phoneNumber = value;
      } else {
        _phoneNumber = phoneFormatter.format(value);
      }
      // Create and dispatch event
      var eventObj:Event = new Event("phoneNumberChanged");
      dispatchEvent(eventObj);
    }
    ]]>
  </mx:Script>
  <mx:PhoneFormatter id="phoneFormatter"
    formatString="(###) ###-####" validPatternChars="#-()
"/>
  <mx:Panel title="Bind with Getters and Setters"
width="500" height="90"
    paddingTop="10" paddingLeft="10" paddingRight="10" pad-
dingBottom="10"
    layout="horizontal">
    <mx:TextInput id="ti1" change="phoneNumber=ti1.text"
maxChars="10" restrict="0-9"/>
    <mx:TextInput id="ti2" text="{phoneNumber}"/>
  </mx:Panel>
</mx:Application>
```

Listing 3 Custom Button class named MyButton

```
package
{
  import mx.controls.Button;

  [DefaultProperty("label")]

  public class MyButton extends Button
  {

  }
}
```

Listing 4 Using the MyButton class wih [DefaultProperty]

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
 xmlns:comps="*">

  <comps:MyButton>
    <mx:String>Test</mx:String>
  </comps:MyButton>

</mx:Application>
```

Listing 5 Custom ButtonLabel class using [Event]

```
package
{
 import mx.controls.Button;
 import flash.events.Event;
 // Define the custom event
 [Event(name="labelChanged", type="flash.events.Event")]
 public class ButtonLabel extends Button {
   // property to hold label value
   private var _myLabel:String;
    // public setter method
   public function set myLabel(s:String):void {
     _myLabel = s;
     this.label = s;
     // Create and dispatch custom event
     var eventObj:Event = new Event("labelChanged");
     dispatchEvent(eventObj);
   }
 }
}
```

Listing 6 Using the ButtonLabel class with the labelChanged [Event]

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
 xmlns:comps="*"
  backgroundColor="#FFFFFF">
  <mx:Script>
    <![CDATA[
      import mx.controls.Alert;
      import flash.events.Event;

      // method to handle custom event
      public function labelChanged(eventObj:Event):void {
        myTA.text= myTA.text + "\n"+ eventObj.target.label;
        myTA.verticalScrollPosition = myTA.verticalScrollPosition +
20;
      }
    ]]>
  </mx:Script>
  <mx:Panel title="Event Sample" width="500" height="275"
    paddingTop="10" paddingLeft="10" paddingRight="10" paddingBot-
tom="10"
    layout="absolute">
   <mx:TextInput id="buttonLabelTI"
     change="myButton.myLabel=buttonLabelTI.text" x="10" y="9"/>
   <!--Instantiate custom class and define method to handle label-
Changed event-->
   <comps:ButtonLabel id="myButton" labelChanged="labelChanged(event)
;"
     x="10" y="39"/>
   <mx:TextArea id="myTA" width="200" height="200"  x="249" y="10"/>
 </mx:Panel>
</mx:Application>
```

Listing 7 Add the Effect metadata tag

```
...
// Define the custom event
 [Event(name="labelChanged", type="flash.events.Event")]
 [Effect(name="labelChangedEffect", event="labelChanged")]
 public class ButtonLabel extends Button {
...
```

Listing 8 Add labelChangedEffect to the Component Instantiation MXML Tag

```
<comps:ButtonLabel id="myButton" labelChanged="labelChanged(event);"
     labelChangedEffect="myEffect" x="10" y="39"/>
```

Listing 9 Custom component with [Inspectable] defined

```
<?xml version="1.0" encoding="utf-8"?>
<mx:HBox xmlns:mx="http://www.adobe.com/2006/mxml">

  <mx:Script>
    <![CDATA[
      [Inspectable(defaultValue="Visa",
        enumeration="Visa,Mastercard,Discover,American Express"
        category="Credit Card" type="String")]
      public var ccType:String;
    ]]>
  </mx:Script>

</mx:HBox>
```

Listing 10 Using [NonCommittingChangeEvent]

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    backgroundColor="#FFFFFF">

  <mx:Script>
    <![CDATA[
      import flash.events.Event;
      private var eventObj:Event;

      [Bindable(event="triggerBinding")]
      [NonCommittingChangeEvent("change")]
      private var s:String;

      private function triggerBinding():void{
        eventObj = new Event("triggerBinding");
       dispatchEvent(eventObj);
      }
    ]]>
  </mx:Script>
  <mx:Panel title="NonCommittingChangeEvent Sample"  width="500"
height="90"
    paddingTop="10" paddingLeft="10" paddingRight="10" paddingBot-
tom="10"
    layout="horizontal">
    <mx:TextInput id="ti1" change="s=ti1.text" enter="triggerBinding()"
/>
    <mx:TextInput id="ti2" text="{s}" />
  </mx:Panel>
</mx:Application>
```

Listing 11 Custom Class CustomCircle using [Style] tags

```
package
{
  import mx.core.UIComponent;

  [Style(name="borderColor",type="uint",format="Color",inherit="no")]
  [Style(name="fillColor",type="uint",format="Color",inherit="no")]
  public class CustomCircle extends UIComponent {

    public function CustomCircle(){
      super();
    }

    override protected function updateDisplayList(unscaledWidth:Number,
unscaledHeight:Number):void {
      super.updateDisplayList(unscaledWidth, unscaledHeight);
      graphics.lineStyle(1, getStyle("borderColor"), 1.0);
      graphics.beginFill(getStyle("fillColor"),1.0);
      graphics.drawEllipse(0,0,100,100);
    }
  }
}
```

Listing 12 Using CustomCircle and assigning custom style properties

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:comps="*"
  backgroundColor="#FFFFFF">
  <mx:Panel title="Style Sample"  width="200" height="200"
    paddingTop="10" paddingLeft="10" paddingRight="10" paddingBot-
tom="10"
    layout="horizontal">
    <comps:CustomCircle borderColor="#000000" fillColor="#FF0000" />
  </mx:Panel>
</mx:Application>
```

Video Rock ’n Roll with Flex 2

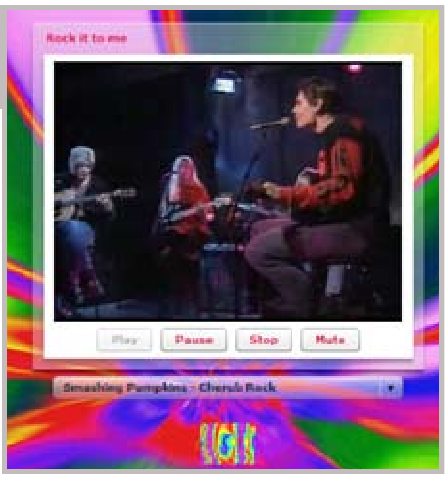
Streaming videos

by Michael Givens

flex 2 and the Flash Media Server are a match made in heaven. By combining the two technologies, streaming Flash videos at my office has become a nice part of my daily routine and a good excuse to add another monitor to my desktop. Whether your musical genre taste is classical or rock ’n roll, it’s relatively easy to quickly create a desktop library of Flash videos. In this article, I’ll describe one approach that will have you streaming in no time.

You will need the Flex 2 Builder (with Flex 2 SDK), the Flash Media Server (the Macromedia Communication Server works as well), and at least one or two Flash videos (flv). I will omit the installation details in this discussion and get to the good stuff right away. For a quick demo, try this example: http://www.insideflex.com/fmsvideo/bin/fmsvideo.html. The full source code is available at http://www.insideflex.com/fmsvideo/bin/srcview/index.html.

Figure 1



Stepwise, first create a new Flex 2 project in the Flex 2 Builder IDE (for this project there’s one named fmsvideo_flex). The code and the explanation for the main MXML file are discussed below (see Listing 1).

The application is initialized with the initApp()1 function where the VideoDisplay control is displayed and the HttpService is called. This accomplishes two main things. First, Flex creates a VideoDisplay control (http://livedocs.macromedia.com/flex/201/langref/mx/controls/VideoDisplay.html) with no visible user interface; by setting the source attribute to an flv and the autoPlay attribute to false, the control gets displayed in the UI. Second, the data (see Listing 2) for the ComboBox video list is requested.

The creationComplete attribute includes a call to the PageLoaded()2 function that kicks off, at one-second intervals, a function that determines if the HttpService is complete; the readytoContinue()3 function loads the videos’ names and labels into an array, dp, that is the dataProvider of the ComboBox control, cbxVideos.

Selecting a video triggers the comboBox’s change event and the function, playVideo(cbxVideo.selectedItem.name)4, and the video begins streaming its content.

The standard Play, Mute, Pause and Stop buttons are included, providing complete playback control. Utilizing an EventListener, vidDisplay.addEventListener(Event.COMPLETE, onComplete)5 , the buttons are enabled or disabled in a logical fashion depending on the video’s playback state.

This article would not be complete without discussing the placement of the Flash video flvs. You should create a new Flash Media Server application – by adding new folders (for this demo, videos\streams_definst_) in the applications folder (for a default installation, you would end up with this folder: “C:\Program Files\Macromedia\Flash Media Server 2\applications\videos\streams_definst_”). Copy the flvs and paste them here. If you need some flvs for your VideoPlayer, you can grab the URLs from YouTube.com and use a tool you’ll find at TechCrunch – http://www.techcrunch.com/get-youtube-movie/. This effectively allows you to download the flvs of your choice.

I hope you found this article useful. If you have any follow-up questions, feel free to ping me at the e-mail address listed with this article. Rock on!

Mike Givens is the CTO of U Saw It Enterprises, a Web technology consulting firm based in Marietta, GA. He is an Adobe Community Expert (Flex) as well as an Adobe Corporate Champion known to share his experience and evangelism of all things Adobe. Certified in both ColdFusion 5 and as an Advanced CFMX Developer, he has been using ColdFusion since the days of Allaire Spectra. Mike blogs at http://www.flexination.info/. info@webmxml.com

Listing 1: fmsvideo.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application pageTitle="My FMS Video Player" xmlns:
mx="http://www.adobe.com/2006/mxml" initialize=" HYPERLINK
\l "initAppfx" initApp()1" creationComplete=" HYPERLINK  \l
"PageLoadedfx" PageLoaded()2" layout="vertical">
 <!—this http service is used to supply the dataProvider for
the ComboBox list of videos
<mx:HTTPService id="videoService" url="theVideos.xml"
showBusyCursor="true" />

<mx:Script>
 <![CDATA[
 import mx.events.VideoEvent;
 import mx.controls.Alert;

 [Bindable] private var dp:Array;
 [Bindable] private var bgimage:String = "";
 private var blnLoadDelayRunOnce:Boolean = false;

 private function initApp()1:void {
 vidDisplay.autoBandWidthDetection = false;
              vidDisplay.source = "rtmp://your_domain/videos/
defleppard_photograph.flv";
  videoService.send();
 }

 private function PageLoaded()2:void {
  var myTimer:Timer = new Timer(1000, 0);
  myTimer.addEventListener("timer", onTimer);
  myTimer.start();
 }

 private function onTimer(event:TimerEvent):void {
  if (!blnLoadDelayRunOnce) {
   readytoContinue();
  }
 }

 private function readytoContinue()3:void {
  if (videoService.lastResult.videos.video!=undefined) {
   //mx.controls.Alert.show("number of videos: " + videoSer-
vice.lastResult.videos.video.length);
   var aryTemp:Array = new Array();
   for (var i:Number=0; i<videoService.lastResult.videos.
video.length; i++) {
 aryTemp.push({name: videoService.lastResult.videos.video[i].
name, label: videoService.lastResult.videos.video[i].label});
 }
 dp = aryTemp;
 cbxVideo.dataProvider = dp;
 }
 blnLoadDelayRunOnce = true;
}

private function playVideo(stream:String)4:void {
 if (stream!=null) {
 //Alert.show(stream);
 vidDisplay.autoBandWidthDetection = false;
 vidDisplay.source = "rtmp:// your_domain /videos/" + stream
+ ".flv";
 vidDisplay.addEventListener(Event.COMPLETE, onComplete)5;
 vidDisplay.play();
  btnPlay.enabled = false;
  btnMute.enabled = true;
  btnPause.enabled = true;
  btnStop.enabled = true;
               swfLoad.source = "delirio.swf";
 }
}

 private function onComplete(event:Event):void {
 swfLoad.source = null;
  btnPlay.enabled = true;
  btnMute.enabled = false;
  btnPause.enabled = false;
  btnStop.enabled = false;
 }

 private function pauseVideo(stream:String):void {
  if (stream!=null) {
   vidDisplay.pause();
   btnPlay.enabled = true;
```

```
                swfLoad.source = null;
 }
}


private function stopVideo(stream:String):void {
 if (stream!=null) {
  vidDisplay.stop();
  vidDisplay.close();
  btnStop.enabled = false;
  btnPause.enabled = false;
  btnMute.enabled = false;
  btnPlay.enabled = true;
                swfLoad.source = null;
 }
}

private function muteVideo(stream:String):void {
 if (stream!=null && btnMute.label=='Mute') {
  vidDisplay.volume = 0;
  btnMute.label = "Sound On";
                swfLoad.source = null;
 } else if (stream!=null && btnMute.label=='Sound On') {
  vidDisplay.volume = 1;
  btnMute.label = "Mute";
                swfLoad.source = "delirio.swf";
 }
}

private function viewSource():void {
var u:URLRequest = new URLRequest("srcview/index.html");
                              navigateToURL(u,"_blank");
 }
 ]]>
</mx:Script>

<mx:Panel title="Rock it to me" color="#E21C51"
height="{vidDisplay.height + 80}" width="{vidDisplay.width +
40}" horizontalAlign="center" verticalAlign="middle">
<mx:VideoDisplay id="vidDisplay" source="{null}"
autoPlay="false" maintainAspectRatio="true"/>
 <mx:HBox>
         <mx:Button id="btnPlay" label="Play" click="playVid
eo(cbxVideo.selectedItem.name)" enabled="false"/>
         <mx:Button id="btnPause" label="Pause" click="pause
Video(cbxVideo.selectedItem.name)" enabled="false"/>
         <mx:Button id="btnStop" label="Stop" click="stopVid
eo(cbxVideo.selectedItem.name)" enabled="false"/>
         <mx:Button id="btnMute" label="Mute" click="muteVid
eo(cbxVideo.selectedItem.name)" enabled="false"/>
 </mx:HBox>
</mx:Panel>
<mx:ComboBox id="cbxVideo" dataProvider="{dp}"
labelField="label" change=" HYPERLINK  \l "playVideo"
playVideo(cbxVideo.selectedItem.name)4" toolTip="Select the
video you want to play..."/>
<mx:Spacer height="10"/>
<mx:SWFLoader id="swfLoad" source="{null}" width="50"
height="50" click="viewSource()" toolTip="Click to view the
source..."/>

</mx:Application>
```

Listing 2: theVideos.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<videos>
 <video>
  <label>Select a video...</label>
  <name>null</name>
 </video>
 <video>
  <label>Def Leppard - Hysteria</label>
  <name>defleppard_hysteria</name>
 </video>
 <video>
  <label>Def Leppard - Love Bites</label>
  <name>defleppard_love_bites</name>
 </video>
</videos>
```

Advertising Index

Advertiser	URL	Phone	Page
Adobe	www.adobe.com		2
CFDynamics	www.cfdynamics.com	866-233-9626	6
CFUnited	www.cfunderground.com		19
Community MX	www.communitymx.com		15
Cynergy			8
Hot Banana			35
Host My Site			12
JavaOne			11
Intergral	www.fusiondebug.com		5
Omniture			11
PaperThin	www.paperthin.com		37
Stream57	www.stream57.com	212-909-2550x1012	15
Vitalstream	www.vitalstream.com	800-254-7554	24

Hot Banana Web Sites Are Built Fully-Loaded!

Why settle for building and managing a vanilla Web site when you can savor a full featured Hot Banana Web site?

Hot Banana is a fully-loaded Web Content Management System with all the Web Site Optimization and Marketing Automation tools your Marketing department craves. And, it's easy-to-use, search engine friendly and fine tuned for generating and converting the maximum number of leads.

So what's the cherry on top? It's ColdFusion of course - you'll love how easy Hot Banana is to implement, integrate and customize with a robust, yet semi-open API.

Schedule your Free Taste Trial Today!

Features...

- 100% browser-based
- Updated RTE
- Powerful DAM
- XHTML & CSS compliant
- Multilingual support
- Granular security
- Flexible workflow
- Press release manager
- Keyword analysis
- Web analytics center
- A/B Testing
- Web Forms
- RSS & blogs
- SEO tools
- Content reuse & scheduling
- Lead source tagging & tracking
- Custom metadata
- Event registration
- Email manager

Contact Us Now:
hotbanana.com/cfdj-demo
1.866.296.1803
sales@hotbanana.com

hotbanana®
Web Content Management For Marketing

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.

HostMySite.com

WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL