MX developer's journal

THE LEADING MAGAZINE FOR MACROMEDIA MX DEVELOPMENT, DESIGN, & PRODUCTION TOOLS

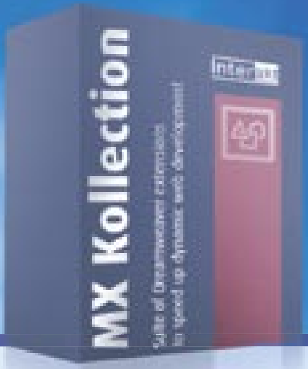
volume 3 issue 7 2005 www.mxdj.com

SUBTLE MULTITHREADING

FLASH
Flash Panels

CAPTIVATE
Delivering Captivate Content

FLEX
Using Runtime Shared Libraries

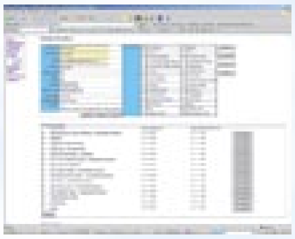

MX Kollection
Suite of Dreamweaver extensions to speed up dynamic web development
interakt

MX Kollection 3

A new era in web development

4 of the **12.226 applications** developed with MX Kollection

Intranets



E-Commerce

CMS

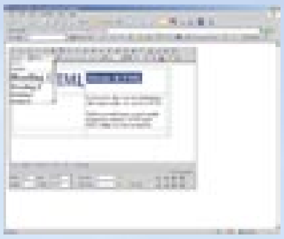
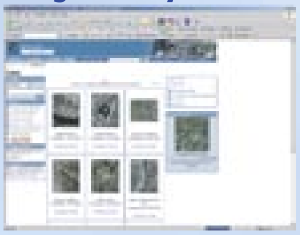
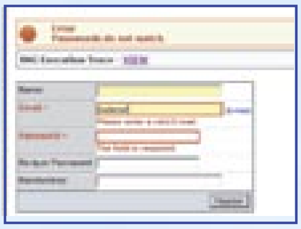


Image Gallery



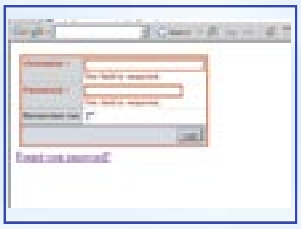
12 of the **423 features** included in MX Kollection

Form Validation



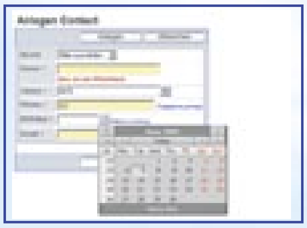
Validate form fields
Client and server side validation
Rich validation formats library
Preserve submitted values on error
Date picker, Numeric control
Masked Textfield, Restricted Textarea

User Login



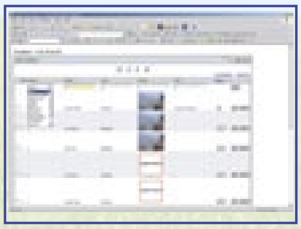
User registration and login
Encrypted password
"Remember me" feature
Account activation by e-mail
Restrict access to page
Forgot password feature

Rich Internet Applications



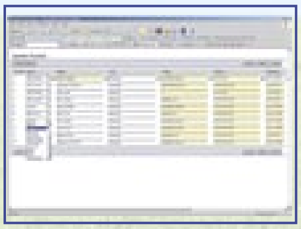
Create incredibly good looking forms
Date Picker
Dependent Drop-downs
Combo-box
Masked textfield
Numeric textfield

Manage your database



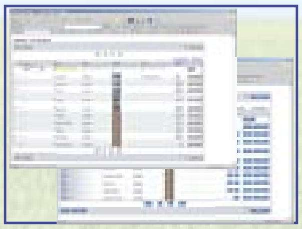
Search records in the database
Insert/Update/Delete forms
Table and CSS form generation
Automatic navigation bars
Automatic list filter
Order your database information

Edit Multiple Records at once



Excel-like multiple records form
Select Records from the list
Edit records in an HTML form
Insert multiple records at once
Delete multiple records directly from list

CSS skins for custom look



4 predefined CSS skins
Create your own skins with ease
Define fonts, colors and layouts
Duplicate edit buttons on top

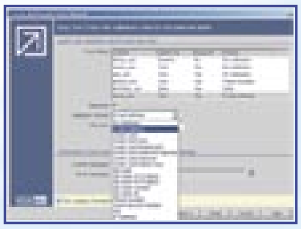
Interakt   macromedia approved   macromedia® ALLIANCE PARTNER

1 of the **5628 customers** working with MX Kollection

"You cannot regard this product as an extension or an add-on or even an upgrade. I regard it as a new era in RAPID dynamic web development. You will need to come up with a new term for Dreamweaver add-ons. It definitely borders on the lines of a CASE (computer aided software engineering) tool."

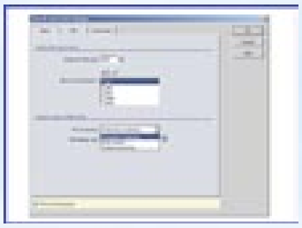
Michael Hamlett, web developer

Dreamweaver Wizards



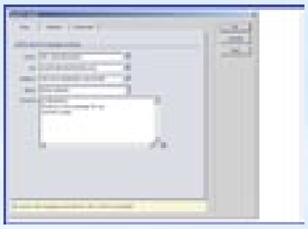
Step by step wizards
Contextual help in each interface
Multi-tabbed server behaviors
Control Panel to manage your site
Automatic included files update

Upload Files and Images



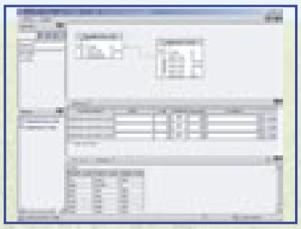
File and Image Upload
Proportional Image resize
Impose maximum file size
Dynamic Thumbnail Creation
Download Uploaded File
Delete files on record delete

Send E-mails



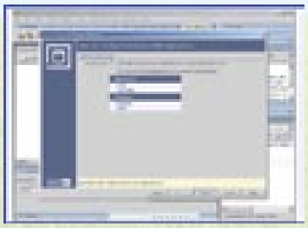
Send e-mail on form submit
Send e-mail to multiple recipients
Send page section by e-mail
CSS and image links in e-mail body
Set e-mail priority

Create SQL Queries Visually



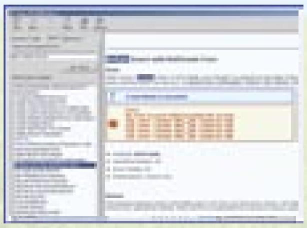
Create, Edit recordsets visually
Visually add tables to query
JOINS between tables
Define and apply complex conditions
Extract information from multiple tables
Synchronize with database changes

Horizontal loopers



Horizontal looper
Vertical looper
Nested repeat regions
Create image galleries or reports

Documentation & tutorials



CHM Documentation with search
In-depth tutorials
Online troubleshooter
Online knowledge base
How-to section make problems history

work smart

http://www.interaktonline.com/

CFDynamics

Dedicated Server Packages Starting at $189/mo.

All dedicated servers include:

- FREE STATS SOFTWARE!
- No long term commitments!
- FREE SQL server access!
- FREE MAIL SOFTWARE!
- Fair and simple pricing!
- Optional server maintenance!

As one of the premier ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to provide **better service** to ensure your satisfaction. Not only do we offer the **finest in shared hosting plans**, but we now offer the **finest in 100% dedicated server plans**! Now you can afford the freedom of having your own dedicated server!

When your needs have outgrown shared hosting look to CFDynamics for **total freedom.** With dedicated server packages they're not offering an oxymoron; "virtually private" or "virtually dedicated" is NEITHER private nor dedicated. CFDynamics offers a solution that is **100% completely dedicated**. They don't play games with the fake stuff; CFDynamics only offers the real deal. Real Service. Real Satisfaction. Real Value.

Real Freedom.

macromedia ALLIANCE PARTNER

Paper|Thin Partner

BlueDragonCertified V.I.P. Hosting Partner

Visit us online or call to order!

WWW.CFDYNAMICS.COM

1 - 866 - 233 - 9626
866 - CFDYNAMICS

Seapine Software™
changing the world of software development

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

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

Surround SCM lets you...

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

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

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

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

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

Remotely access your source code repository from Dreamweaver MX.

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

Features:

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

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

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

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

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

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

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

Want to learn more?

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

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

Surround SCM Source Control
TestTrack Pro Issue Management
QA Wizard Automated Testing

macromedia ALLIANCE PARTNER

The MX Blogosphere

Our regular feature taking an online stroll around the MX world

by mxdj news desk

don't forget that you can blog yourself now, too, under the MXDJ domain – just follow the blog-n-play link from http://mxdj.sys-con.com.

Blog Topic: Breeze

Open Studio Discussions Breeze Room

by alan musselman

from http://weblogs.macromedia.com/amusselman/

The Open Studio Discussions (http://macromedia.breezecentral.com/osd/) is a Breeze room that will be open for 2 hours from 7-9pm PST every other Tuesday (starting July 5, 2005) for anyone that uses Studio MX 2004 (Fireworks, Dreamweaver, FreeHand, Flash) to come in and ask questions, share ideas, tips, techniques, and explore Breeze on a more personal level.

This is pretty much an underground thing and generally after hours for me, but I'm looking forward to seeing sites and apps, chit-chat or anything cool you have to show off and/or help you get an issue sorted out.

A lot of times passing emails back and forth can become tedious, time consuming and complex in support so I use Breeze to communicate directly with the customer to identify the issue, determine a solution, and provide guidance and help to implement that solution. It's that simple...if I can't understand the issue through email, I'll ask them to join a Breeze Meeting. I feel like were both on the same level during the entire experience, its great!

Remember: *2 hours*
7:00-9:00PM sync time zone (http://www.worldtimeserver.com/meeting-planner.aspx) I'm in California
07-05-2005 (every other Tuesday)

"You should always declare local variables at the top of your ‹cf function› tag"

Blog Topic: ColdFusion

Persistent CFCs and CFQUERY

by tom muck

From http://www.tom-muck.com/

I see a lot of people using CFCs in session and application scope who do not take into account that you should always declare local variables at the top of your

```
<cffunction> tag:
<cffunction name="blah"
returntype="any">
<cfset var i = 0>
<cfloop from="0" to="10" index="i">
<!--- Some code --->
</cfloop>
</cffunction>
```

Many people are doing this, but I wonder how many people apply the same principle to recordsets within the CFQUERY tag. . . .I see this a lot:

```
<cffunction name="testRS"
access="public" output="false">
  <cfquery name="rs"
datasource="Northwind">
  SELECT * FROM Products
  </cfquery>
  <cfreturn rs>
</cffunction>
```

If this is in a persistent scope, the variable rs will be available even after the return call. In fact, it will hang around for the life of the persistent CFC. To properly scope the query, you should declare it first:

```
<cffunction name="testRS"
access="public" output="false">
  <cfset var rs = "">
  <cfquery name="rs"
datasource="Northwind">
  SELECT * FROM Products
  </cfquery>
  <cfreturn rs>
</cffunction>
```

Now, the rs query will be destroyed after the function returns the variable to the caller -- it is not persisted within the CFC. You can try it like this. Create a cfc:

```
<cfcomponent>

<cffunction name="testRS"
access="public" output="false">
<cfquery name="rs"
datasource="Northwind">
SELECT * FROM Products
</cfquery>
<cfreturn rs>
</cffunction>

<cffunction name="testRSBad"
access="public" output="false">
<cfreturn rs>
</cffunction>

</cfcomponent>
```

The function testRSBad() looks like it should throw an error, because rs is not defined, however if this is in persistent state and you hit the testRS() method first, then rs is persisted for the entire session.

Try it out: make sure you have sessions turned on in the Application.cfm file. Then put some code on a page called testrs.cfm:

```
<a href="testrs.cfm?hit=true">Next
page</a>

<cfif not isdefined("url.hit")>
  <cfset session.user1 = createobject(
"component","testuser").new()>
  <cfset session.user1.testRS()>
</cfif>

<cfdump var=#session.user1.testRS-
Bad()#>
```

You have a link to the page, and you are instantiating the session instance of the CFC only once -- when you preview the page. When you hit the link, you will see the page again with the same recordset dumped out, but this time it is coming from the leftover persisted global variable rs that was not scoped properly in the CFC.

Blog Topic: Flash Lite

Quest for a Flash Lite Enabled Phone

by tim walling

From http://www.timwalling.com/

I think one of the biggest questions right now for many Flash developers getting involved with Flash Lite is "Where can I get a phone that will let me play with this stuff?" Of course you can jump right in and use the standalone player but that's not as much fun as carrying your Flash apps around, showing your friends, impressing the ladies, etc.

There's been some great input on the FlashLite mailing list (http://groups.yahoo.com/group/FlashLite/) and I've definitely learned a lot regarding which phones to get, steps to using your PC's Internet access via Bluetooth and lots of other things.

So here's where I'm at right now. I picked up a Nokia 3650 on eBay with the intent of either a) using Bluetooth to get some internet access on it and/or b) getting a data only plan from T-Mobile for real-life testing. Right now I don't have either of these options working.

I didn't realize I need a SIM card in order to just use my phone for local testing (option A). When I turn the phone on it asks for a SIM card and I'm not sure if there are ways around this. If there are, someone please let me know.

Option B is still feasible, I just don't know if I'm ready to get a $30 data-only plan. Let me tell you the people at T-Mobile weren't very helpful in this area either. The first person I spoke with had no clue about data-only plans and said they didn't have anything like that. Finally when someone else got involved he confirmed it. I tried explaining how I just wanted to do some software development on this phone I had picked up from a friend. After finally establishing this I asked what my options were and if they had any deals on any Series 60 phones since any plans would require a one-year contract. Asking to see what Series 60 phones they had in store got me a few weird looks also and I think he didn't understand the question. There's nothing like going to a store and knowing more about their products and services than they do.

What I'd love to see down the road to help developers:

1. Hardware manufacturers like Nokia, please get word out about current trends and uses for your phones (example: Flash Lite).
2. Mobile service providers, please take advantage of the growing market here. Educate your sales people and offer some more data plan options. As soon as Flash Lite takes off the first provider to offer some simple data plans are going to gain some new customers. My plan is through Verizon right now, but as soon as a I see a cheap and easy plan for my Flash enabled phone I'm going to either completely switch over or have 2 plans going.
3. Either of the above companies, it would be great to see you at some of the upcoming Flash conferences (FlashForward, MAX 2005, etc). Being able to talk to some representatives who are familar with Flash would be awesome.

Might be a lot to ask for, but why not. It's still a grey area for Flash developers and I'm sure it'll get better. I know I'll find some way to get this phone working soon.

Blog Topic: Dreamweaver

Coming Back Around to Dreamweaver

by rich rodecker

From http://www.visible-form.com/blog/

Lately I find myself being pulled back into liking Dreamweaver. I had stopped using it for a long time becasue I had felt it was too bloated, and offered up a lot of features I didn't need. What happened then was that I had separate apps open for coding my PHP and XML files, and then a separate app for my ftp.

What I'm really appreciating now is that I can code most of my files in one place (PHP, XML...not ActionScript though, that still stays with SEPY), and upload and test them with one key command. That's a welcome change form editing my code in one editor, switching to the ftp editor to upload, then opening the page in the browser to test. I know Eclipse can probably do the same thing, but eclipse...I dunno I just can't get into it. too clunky?

The Site Manager is pretty sweet too, it really helps keeps thing organized, and it's pretty cool to be able to jump back and forthe between different sites just by selecting from a drop down (in the files panel).

There's some downsides too. I still don't wind up using like half the panels available to me. I wish there was an option to only load the panels I need, in order to make the startup faster...then be able to load them as necessary (that wish applies to all software). I know I can open and close panels at will but it's not the same thing.

The live data feature is pretty cool, but it still get all sort of wonky. Being that I can test my files live with a click of a button, I really don't care about that (or use that feature) anyway. The reference panel needs some work too.

I wonder what will happen now with the Adobe merger and Macromedia joining the Eclipse Foundation? I think DW would actually improve with some help from an Adobe UI.

Harnessing Shared Objects

How to make use of draggable Flash elements

by sam coles

With our tunes playing and the volume knob to about the 3:00PM position (I prefer it loud), let's get started. First you need to create some content to drag. I just made some nice looking mockup boxes for now.

With the content boxes created they need instance names. Anything will work, just give them appropriate and easy to remember names. In my case I gave them the instance names news and images accordingly.

Ok with the 'design' section out of the way now it's time to get dirty with some ActionScript. I like to keep everything on a frame in the main timeline. It allows all my code to be easily accessible and even allows me to shorten code! We'll get to that later but for now we need to figure out how we're going to set this up.

To remember where the objects are dragged to we're going to have to use the shared object. Shared object is an extremely useful tool when designing Flash applications. It's exactly like a cookie but specific to Flash. It is a special beast though. The special object doesn't allow data to be directly written to it like so:

"Shared object is an extremely useful tool when designing flash applications."

```
sharedObj.data.myName = "Sam";
```

All data that is to be written to the shared object has to be by reference. For example:

```
var name:String = "Sam";
sharedObj.data.myName = name;
```

This isn't a problem however as we only have to slightly change how we design the code portion of this project. With that sorted out let's define the three variables we'll need for this project – positions array, instances array, and the shared object itself.

```
//Insert your own instance names into
the array
var instances:Array = [_root.news,
_root.images];
var so:SharedObject = SharedObject.
getLocal( "pps.com/tutorial/userPosi-
tions");
var positions:Array = [[],[]];
```

I chose to have the positions of the movie clips in the instances array stored in a 2d array. I could have used objects with the properties x and y in a 1d array but I just like arrays. The positions will also be stored in the shared object as a 2d array named posArr. Now onto writing the functions we need.

For each movie clip that we want to be draggable on stage we have to write a handler. You can imagine that this would be a pain because for each instance we'd have to write these handlers.

```
mc.onPress = function(Void):Void{
   this.startDrag();
}
mc.onRelease = function(Void):Void {
   this.stopDrag();
}
```

But (bum bum bummm!) thanks to some clever coding, we can use a for loop to loop through an array we take in as a parameter to set all our handlers. Let's begin function header and the for loop. (Note: I personally prefer putting all the functions etc. I write above where I define my vars. It helps keep code more organized.)

```
function setHandlers(mcs:Array):Void {
   for( var a:Number = 0; a <= mcs.
length-1; a++){
```

Now we need to set up the on press and on release handlers of our movie clips, then just close the for loop and function.

```
      mcs[a].onPress = function(Void):
Void{
         if(this.onEnterFrame){
            delete this.onEnterFrame;
         }
         this.startDrag();
      }
      mcs[a].onRelease =
function(Void):Void {
         this.stopDrag();
         setPositions(); //inserts
object's x/y coords into SO
      }
   }//end for a
}//end setHandlers
```

Q. Why are we checking for a nonexistent on enter frame handler?
A. Don't worry for now, it'll be clear later on why we need this if test.

Q. What's this setPositions function about?
A. The setPositions function is the function we're about to write that will actually update the shared object with the positions of our elements on stage.

The set positions function is extremely simple. First we have to use a for loop to go through the instances array we defined earlier.

```
function setPositions(Void):Void {
   //Cycle through all the instances
   for( var a:Number = 0; a <=
instances.length-1; a++){
```

Now, we could just reset every instance's x/y coordinates but that'd take extraneous processing power if they hadn't been moved. Because of that we just need to have a simple if test to check if the x or y coordinates of the movie clip in question are different then those in the array.

```
      if( instances[a]._x !=
positions[a][0] || instances[a]._y !=
positions[a][1]){
         positions[a][0] =
instances[a]._x;
         positions[a][1] =
instances[a]._y;
      }
   }//end for a
```

With our if test done and for loop closed we just have to set the shared object's 2d array posArr to positions.

```
   so.data.posArr = positions;
   so.flush(); //force so to write
}//end setPositions
```

Q. Wait up a second, wouldn't we have to use slice to copy over the positions array since it'd be passed by reference?
A. Although this would be the case any other time you wanted to completely copy an array the shared object is an exception.

Now that we have the setHandlers and setPositions functions written we only have one more function to write. Although it's the longest function it's by no means the hardest. This function is of course the placeMovieClips function. Lets begin by writing the function header. Now we'll probably want to take in the array with the instance names and the shared object's position array as parameters. We do this for two reasons—increased flexibility and to shorten so.data.posArr to a much simpler name.

```
function placeMovieClips(mcs:Array,
pos:Array):Void {
```

Now we're going to have to loop through all the instances in the mcs array. We can save some CPU here by having an if test right off the bat to see if the current item we're examining in the mcs array isn't in the position that's stored in the shared object.

```
   for(var a:Number = 0; a <= mcs.
length-1; a++){
      if(mcs[a]._x != pos[a][0] ||
mcs[a]._y != pos[a][1]){
```

Onto some easing! We need two values right off the bat – the x destination, and the y destination. Now since it's impractical to retrieve values from the shared object from an on enter frame the best option is to use local variables. In our case there's two: xDest, and yDest.

```
         mcs[a].destX = pos[a][0];
         mcs[a].destY = pos[a][1];
```

Now we have to set up the on enter frame for the movie clip. Easing is often portrayed as a very hard subject when in fact it's extremely simple. The easing we're going to be using works off the difference between the destination x or y coordinate versus the current x or y coordinate. By adding a fraction of the x and y difference to the current x and y coordinates we get a nice exponential easing effect. But as some of you may or may not know this has a problem inherit to exponential decay. The amount we add to the x and y never reaches 0. To circumvent this waste of CPU we need to check if |xDist| and |yDist| are greater than one. Enough talk, lets code this bad boy.

Q. Why do we have to use absolute value?
A. If we didn't use absolute value the easing wouldn't work because xDiff and yDiff can both be negative so we have to force them to be positive.

```
mcs[a].onEnterFrame = function(Void):
Void {
            //Get the differences
            var xDiff:Number = this.
destX-this._x;
            var yDiff:Number = this.
destY-this._y;

            trace("XD:"+xDiff);
            trace("YD:"+yDiff);
            //check if |yDiff| and
|xDiff| >1
            if( Math.abs(yDiff) > 1
&& Math.abs(xDiff) > 1){
               //add to x and y
               this._x += xDiff/8;
               this._y += yDiff/8;
```

Now we have to write an else that will stop the easing outright. To do this we need to write an else that first places the movie clip to the destination x and y coordinates and second kills the on enter frame of our movie clip.

```
            }else{
               this._x = this.xDest;
               this._y = this.yDest;
               trace("OEF Killed");
               delete this.onEnter-
Frame;
            }
         }//end OEF
      }//end if
   }//end for a
}//end func placeMovieClips
```

Ok we're almost done; we only have a few more lines of code to insert. We need to put this code below where we defined all our variables earlier. We just have to execute setHandlers and execute the placeMovieClips function if the shared object has been written to before.

```
setHandlers(instances);
if(so.data.posArr != null){ //check if
so has been written to before
   placeMovieClips( instances,
so.data.posArr);
}
```

Test your movie and it should come out something like this (move the elements then refresh to see easing). I bet you can't guess what song the lyrics are from!

sjcoles@charter.net

In this article I will explain how Flash Panels fit into the grand scheme of extending Flash MX 2004. We'll also discuss some of the benefits and pitfalls you may encounter when using Flash Panels in your day to day work.

by steven grosvenor

Creating & Implementing Flash Panels

Through this tutorial, you'll create your very own Flash Panel to control the rotation of Movie Clips on the stage using standard Flash MX 2004 components, a hefty sprinkling of ActionScript and some tips and tricks along the way. I hope you'll come away from this tutorial feeling empowered to create your own Flash Panels, and to explore the capabilities and possibilities of Flash MX 2004 – and your own mind!

Before we set out on this extensibility trip, let me point out a couple of resources that will be invaluable in your pursuit of Flash Panel excellence:

- Flash MX 2004 JavaScript Dictionary: An invaluable bible that contains nearly all the Flash API information that you'll ever need. (http://www.macromedia.com/support/documentation/en/flash/#flashjsdict)
- JSFL File API: Not included in the Flash MX 2004 JavaScript Dictionary; functionality added in Flash MX 2004 7.2 udpater. (http://www.macromedia.com/devnet/mx/flash/articles/jsapi.html)

The creation of Flash Panels for use in Flash MX 2004 basically hinges around the understanding and use of the JSAPI (JavaScript API). It's based on a Document Object Model (DOM), which allows both Flash Documents and the internal functions of Flash MX 2004 to be accessed via simple JavaScript-based commands.

Since the release of Flash MX 2004, many JSFL (Flash JavaScript) commands, Flash Panels and custom tools have been created to help automate tasks and add custom interfaces to complex controls that directly influence feedback in the Flash authoring environment. Most of these are easily found via search (use 'JSFL commands' or 'Flash Panels' as your keywords).

If you're comfortable with ActionScript, pushing the boundaries to develop your own custom commands and panels is hardly a leap of faith – it's a small step forward. As the JSAPI is based around the Netscape JavaScript API and Flash's Document Object Model, developing and writing Flash JavaScript should be a natural progression.

By their very nature, Flash Panels are exported SWF files. However, they're subtly different from the standard JSFL files that are used to create commands, as they utilise a wrapper function called MMExecute(). This allows interaction between the compiled SWF and the Flash MX 2004 API.

Consider the following line of JSFL, which returns the current width of the first selected item on the stage:

```
var objectWidth= fl.getDocumentDOM().
selection[0].width;
In order to gain the same functional-
ity within your SWF Panel, this code
needs to be changed as follows:
var objectWidth=MMExecute("fl.getDocu-
mentDOM().selection[0].width");
```

If we examine the code contained within the MMExecute("JavaScript String"), we'll note that it's exactly the same piece of Flash JavaScript we saw above. The only difference is that it's now encapsulated within the wrapper. The MMExecute() function takes the Flash JavaScript string as a single argument and passes it to the Flash API. It's then processed and a return value is optionally given. This value can then be assigned to a variable.

Flash Panel Location

All the major Flash Panels can be found in one simple location within the authoring environment. Simply select 'Window > Other Panels >' to access it in Flash MX 2004.

When you're creating Flash Panels and testing in the live environment, keep the following locations in mind. These are the folders in which Flash MX 2004 locates the custom panels:

```
Windows 2000 or Windows XP
Drive:\Documents and Settings\user\
Local Settings\Application Data\
Macromedia\Flash MX 2004\language\
Configuration\WindowSWF
Windows 98
Drive:\Windows\Application Data\
Macromedia\Flash MX 2004\language\
Configuration\WindowSWF
Mac OS X
Drive:/Users/userName/Library/
Application Support/Macromedia/Flash
MX 2004/language/Configuration/
WindowSWF
```

We will make use of these directory locations later, when we test and deploy the extension.

Inspiration

Sometimes when you're working, you suddenly think 'Gee, wouldn't it be quicker if I could automate [Insert Task Here]?' More often than not, the answer is usually, 'Yep, it'd be great to automate that task ...but how on earth do I do it?'

Enter: Flash Panels... Actually, it's not just the automation of tasks that warrants the creation of Flash Panels; the need for can stem from any of the following (and some other) requirements:

- ***Automation:*** Automate often laborious and time consuming tasks within Flash MX 2004 (Code Addition, Timeline Effects)
- ***Speedier Access:*** Quicker access to menu hidden commands
- ***GUI Control:*** Add a GUI to control real-time effects (rotation, scaling, position etc)

The creation of a Flash Panel can be a daunting task, which is why you need a clear goal for the panel before you begin. Once you decide specifically what you want the panel to do, you're already most of the way to creating the panel (apart from the obvious coding and hooking into the interface).

The next step is to sketch the process flow of the command (how it all works) either on paper, or in a text editor of your choice.

Note: when I'm working in Flash, I always keep next to me a notebook that's dedicated to ideas/workarounds. Sometimes, as you're working away, a need or idea will spring into your mind that you can automate, speed up, or add an interface to, in order to make your life – and those of your colleagues – easier. Keep a list of these ideas so that those fleeting thoughts are never lost and everyone may benefit from the creation of your time-saving panel!

In the example that we're about to create, we will use a single instance of the NumericStepper component to control the rotation of Movie Clips. Consider the following diagram, which shows the command process flow of the command we're about to create in Flash MX 2004:

To this, we'll add a change event handler to catch when the value of the NumericStepper component increases or decreases. When the value changes, the event handler will trigger a function called rotateMe(), which contains all the Flash JavaScript encapsulated in the MMExecute() wrapper function, which is necessary for the function to carry out its given task.

Anyone for a History Lesson?

The History Panel (Window > Other Panels > History), can be a useful insight into the inner workings of Flash MX 2004. When you're looking to recreate an effect via scripted methods, the History Panel can be a good place to start.

During the majority of your user interaction with the application on the stage, if you have the History Panel open, you'll notice events appearing within it. This is a visual representation of the communication history between you the user and the application in JSFL.

figure 1

figure 2

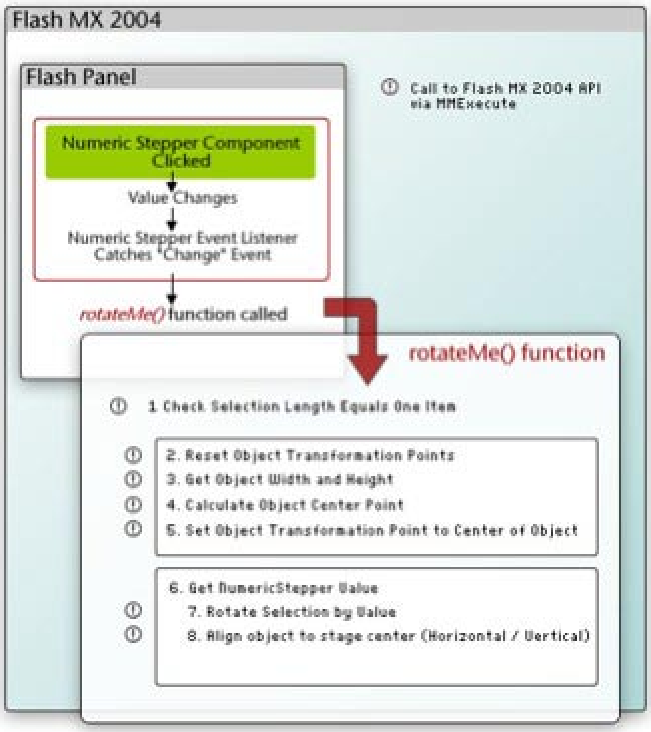
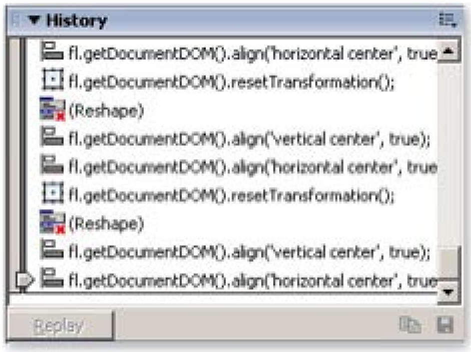

Flash MX 2004
Flash Panel
Call to Flash MX 2004 API via MMExecute
Numeric Stepper Component Clicked
Value Changes
Numeric Stepper Event Listener Catches "Change" Event
rotateMe() function called
rotateMe() function
1 Check Selection Length Equals One Item
2. Reset Object Transformation Points
3. Get Object Width and Height
4. Calculate Object Center Point
5. Set Object Transformation Point to Center of Object
6. Get NumericStepper Value
7. Rotate Selection by Value
8. Align object to stage center (Horizontal / Vertical)

figure 3



```
fl.getDocumentDOM().align('horizontal center', true
fl.getDocumentDOM().resetTransformation();
(Reshape)
fl.getDocumentDOM().align('vertical center', true);
fl.getDocumentDOM().align('horizontal center', true
fl.getDocumentDOM().resetTransformation();
(Reshape)
fl.getDocumentDOM().align('vertical center', true);
fl.getDocumentDOM().align('horizontal center', true
```

The majority of elements within the history can be copied to the clipboard and pasted into your favourite text editor for investigation, except those beside which a red cross appears.

If you're trying to identify the relevant API reference to carry out a given stage-based task that you're trying to automate, and you can't find it withinthe Flash MX 2004 JavaScript Dictionary, execute the task on the stage, and simply copy and paste from the History Panel. It makes an excellent starting point for your own custom commands!

You may also save selected steps (but not those denoted with a red 'x') as a command, which will be made available from the 'Commands' menu provided it doesn't require any user interaction.

Creation

Enough with the introduction! Let's dive into creating a command that rotates the Movie Clips.

Create the Rotator Flash Panel

I've provided the code for the panel here ()The RotatorStart.fla contains the timeline layer structure and the background image for the panel. The finished FLA for this example is called RotatorFinal.fla.

If at any time you need to look up the process flow for the function, refer to the diagram shown above.

Setting the Scene

Our first course of action is to add the component that will control the effect; as the background and layers have already been set up, we need only to add a single component to the stage before we insert the controlling ActionScript.

Of course, it goes without saying that the more complex the panel, the more controls you may have on screen at any one time. I'll leave it to you to experiment with your own creations after you've created this simple but effective example.

1. Open the starting point FLA (RotatorStart.fla from the code archive) and drag an instance of the NumericStepper Component from the 'UI Components' section of the Components Panel onto the first frame of the 'Interface' layer. Name the instance stepSizer.
2. Position the NumericStepper component instance centrally over the rounded rectangle background, and change the default parameter values to the following:
   - Maximum: 360
   - Minimum: 0
   - stepSize: 5
   - Value: 45
3. Save your Flash Document to a location of your choice.
4. Copy the JXLFLAPI.as file from the code archive to the location of your saved FLA. (this is a JSFL Wrapper that's used to simplify some tasks).

Now that we've created the interface, we need to add the controlling ActionScript to bring the effect to life.

Add the ActionScript

It's pretty obvious, but the more things that your panel tries to accomplish, the more complex both the ActionScript and the encapsulated JSFL becomes.

In this example, the code is pretty simple and linear, but as you create your own Flash Panels and begin to extend Flash MX 2004, things can get a little more complex. For this reason, it's often extremely helpful to sketch out the data flow of your command, as I mentioned earlier. You won't regret it!

5. Select the first frame of the Actions layer and add the following code within the Actions Panel:

```
//Stage Controls
Stage.align = "TC";
Stage.scaleMode = "noScale";
Stage.showMenu = false;
//Flash API Wrapper (Courtesy Jesse
Warden)
#include "JXLFLAPI.as"
//Main Rotation Function
function rotateMe()
{
 var selectionChecker = MMExecute("fl.
```

CommonSpot™.

Efficient Content Management

fast. easy. affordable.

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

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

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

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

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

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

Paper | Thin

```
getDocumentDOM().selection.length");
 if (selectionChecker == 1)
 {
   //Reset Transformation Points to
allow easy rotation
   MMExecute("fl.getDocumentDOM().
resetTransformation()");
   //Get Selection Width and Height
   var objectWidth =  MMExecute("fl.
getDocumentDOM().selection[0].width");
   var objectHeight = MMExecute("fl.
getDocumentDOM().selection[0].
height");
   //Calculate Center Points
   var objCenterHorz = int(objectWidth
/ 2);
   var objCenterVert =
int(objectHeight / 2);
   //Move Transformation Point to Dead
Center (Helps when rotating)
   MMExecute("fl.getDocumentDOM().
setTransformationPoint({x:" + Math.
floor(objCenterHorz) + ", y:" + Math.
floor(objCenterVert) + "})");
   //Get Rotation Value
   var incrementer = stepSizer.value;
   //Rotate Selection
   MMExecute("fl.getDocumentDOM().
rotateSelection(" + incrementer +
")");
   //Align H/V to Center of Stage
   MMExecute("fl.getDocumentDOM().
align('vertical center', true)");
   MMExecute("fl.getDocumentDOM().
align('horizontal center', true)");
   //Update Preview Information
 } else
 {
   break;
 }
}
//==========================
//Miscellaneous Functions
//==========================
//Middle Mouse Wheel Support
//==========================
var mouseListener:Object = new
Object();
mouseListener.onMouseWheel =
function(delta)
{
 stepSizer.value += delta;
};
Mouse.addListener(mouseListener);
//==========================
//Create Event Handler / Dispatcher
for Numeric Stepper
//==========================
stepsListener = new Object();
stepsListener.change = function()
{
 rotateMe();
};
stepSizer.addEventListener("change",
stepsListener);
//Numeric Stepper Event Handler Ends
```

Steven Grosvenor is founder of phireworx.com, a Fireworks resource site. Steven is author of Flash Anthology : Cool Effects & Practical Actionscript,and contributing author of Fireworks MX Magic (New Riders), Special Edition Using Fireworks MX (Que), and Fireworks MX Fundamentals (New Riders). He has also authored numerous articles on the Macromedia Developer Center

Let's step through the code and see how it fits together. First, we set the main stage settings, aligning the contents of the stage to TC (Top Centre). We switch off the ability to zoom in, and stop the right click menu from appearing.

```
//Stage Controls
Stage.align = "TC";
Stage.scaleMode = "noScale";
Stage.showMenu = false;
```

We then include a nifty JSFL wrapper from Jesse Warden (www.jessewarden.com), which allows us to encapsulate some flavors of JSFL without needing to worry about sometimes complex single and double escape strings in the MMExecute() function.

```
#include "JXLFLAPI.as"
```

Note: Using the JSFL wrapper, we can simplify the following trace statement: MMExecute("fl.trace(\"Tracing to the Output Panel\")");

The JSFL wrapper simplifies the code as follows:

```
flapi.trace("Tracing to the Output
Panel")
```

Moving on through the process flow of the panel, we must consider the listener object for the NumericStepper component instance that we have on the stage. We use the change event so that, when the user clicks the up or down controllers of the NumericStepper, the rotateMe() function is called:

```
stepsListener = new Object();
stepsListener.change = function() {
 rotateMe();
};
stepSizer.addEventListener("change",
stepsListener);
```

The rotateMe() function is called every time the listener object detects that the selected value of the NumericStepper component has changed. If we refer to the previous process flow diagram, we can see clearly the chain of events that occurs.

First of all, we check that the user has selected only a single item from the stage:

```
var selectionChecker =
MMExecute("fl.getDocumentDOM().selec-
tion.length");
if (selectionChecker == 1) {
```

We then reset the transformation point of the object to a central location. The reason for this is simple: when we rotate the object, it rotates around this transformation point. If the transformation point is off-centre, it can be difficult to gauge what's going on. Resetting the transformation point to the centre point of the object using the object's width and height makes the rotation easier to observe and keeps things tidy.

```
//Reset Transformation Points to allow
easy rotation
MMExecute("fl.getDocumentDOM().
resetTransformation()");
//Get Selection Width and Height
var objectWidth =
   MMExecute("fl.getDocumentDOM().
selection[0].width");
var objectHeight =     MMExecute("fl.
getDocumentDOM().selection[0].
height");

//Calculate Center Points
var objCenterHorz =
int(objectWidth/2);
var objCenterVert =
int(objectHeight/2);
//Move Transformation Point to Dead
Center (Helps when rotating)

MMExecute("fl.getDocumentDOM().
setTransformationPoint({x:"+Math.
floor(objCenterHorz)+", y:"+Math.floor
(objCenterVert)+"})");
```

We then get the current value of the NumericStepper Component, store it in the incrementer variable, and rotate the selection accordingly using rotateSelection(value). As the NumericStepper component facilitates

"My new home is at HostMySite, they have been incredibly helpful and dedicated, and went the extra step to make the transition, well, painless."

- Ben Forta

- Plans starting from 1GB HD storage and 20GB transfer
- Custom tags supported
- Full account management with HostMySite's Control Panel
- Call 24x7x365 for expert ColdFusion support

Visit us at CFUnited June 29th—July 1st
Go to hostmysite.com/cfdj for 2 Months Free Hosting
And $100 off a CFUnited Conference Pass

HostMySite.com
1-877-215-4678

the use of continuous feedback by holding down the direction buttons, this can lead to a pleasing and functional effect.

```
//Get Rotation Value
var incrementer = stepSizer.value;
//Rotate Selection
MMExecute("fl.getDocumentDOM().rotateS
election("+incrementer+")");
```

Finally, we align the object centrally to the stage while rotating it. It's a personal choice of mine to add this code. If it's omitted, the object can drift as a result of the way Flash MX 2004 applies the centralised transformation point (see the earlier discussion).

```
//Align H/V to Center of Stage
MMExecute("fl.getDocumentDOM().
align('vertical center', true)");
MMExecute("fl.getDocumentDOM().
align('horizontal center', true)");
```

That's all we need to do in order to rotate the selected object; however, there's an additional snippet of ActionScript that will give the Flash Panel middle mouse wheel support. This allows us to increase or decrease the value of the rotation either by clicking on the up and down arrows, or by scrolling the mouse wheel up or down. This utilises the same methodology as the event handler for the NumericStepper component, but uses the onMouseWheel event handler to increase or decrease the component's value.

```
var mouseListener:Object = new
Object();
mouseListener.onMouseWheel =
function(delta) {  stepSizer.value +=
delta;
};
Mouse.addListener(mouseListener);
```

6. Save your Flash document, and export the SWF with a suitable name to your Flash MX 2004 'WindowSWF' directory as follows.

```
Windows 2000 or Windows XP
Drive:\Documents and Settings\user\
Local Settings\Application Data\
Macromedia\Flash MX 2004\language\
Configuration\WindowSWF
Windows 98
Drive:\Windows\Application Data\
Macromedia\Flash MX 2004\language\
Configuration\WindowSWF
Mac OS X
Drive:/Users/userName/Library/
Application Support/Macromedia/Flash
MX 2004/language/Configuration/
WindowSWF
```

7. Restart Flash and access the panel from Window > Other Panels > [Name of Exported SWF]

To use the command, simply select a single object from the stage, then use the controls within the Flash panel to control rotation of the object. Now you have a fully functional Flash Panel that controls the rotation of your object in a quick, defined and timely manner!

Note: I usually use the Flash JSFL Wrapper to trace out information to the Output Panel during the development phase. For example, if in this case, I wanted to trace out the current value of the NumericStepper component when middle mouse wheel was scrolled, I would add to our code the lines denoted in bold.

```
var mouseListener:Object = new
Object();
mouseListener.onMouseWheel =
function(delta) {
 flapi.trace("Object rotation is now
"+stepSizer.value+ " degrees");
 stepSizer.value += delta;
};
```

Note also that there are a couple of extra functions I've included at the end of this article to help you on your way!

Now all that remains is to package the SWF into a manageable MXP file that can be installed onto your machine, or computers of your colleagues or anyone that you wish!

Implementation

Before we package the Flash Panel into a distributable format, there are a couple of 'gotchas' that we need to examine!

Updating The Panel While Still in Flash MX 2004

When you make changes to the interface of, or add code to, your Flash

Extra Functions

Here are a few of extra functions to help you on your way with the development of Flash Panels.

Show an Alert

When called from a compiled SWF, this simple piece of code will produce an alert within Flash MX 2004.

```
errMsg = "alert('Please Save Your FLA
before Applying the Effect');";
MMExecute(errMsg);
```

Check the File is Saved

This next section of code will check to see whether the current document has been saved or not, and carries out a conditional function:

```
function checkDocumentIsSaved() {
 var fileDestinationTemp =
MMExecute("fl.getDocumentDOM().path");
 if (fileDestinationTemp != "unde-
fined") {
   //Document is Saved, do something
 } else {
   //Document is NOT Saved, do some-
thing
 }
}
```

Iterate Through Selected Stage Objects

This simple code will iterate through an array of currently selected objects on the stage. This can be extremely useful to change en masse properties of groups of selected objects:

```
var objLength = MMExecute("fl.getDocu-
mentDOM().selection.length");
 for (var i = 0; i<objLength; i++)
     {
     //Do Something to the selected
object here
     flapi.trace(i);
     }
```

HOSTING.COM

THE FIRST NAME IN HOSTING

Developers!

Developers!

Developers!

We Love Them!

SERVE. SUPPORT. SECURE. STORE.

www.hosting.com

Panel projects, you will obviously need to export your updated SWF to the 'WindowSWF' folder. However, in order to see the updates, you'll need to close the panel by clicking the window 'x' button when the panel is undocked and reopen it from the 'Window > Other Panels >' menu, rather than selecting 'Close Panel' from the Options flyout. The reasoning behind this is that clicking the 'Close Panel' option seems merely to hide the panel from view, rather than properly closing it and releasing it from memory.

Name the Exported SWF

I've experienced several 'Name Clash' issues when developing extensions for Flash MX 2004, and they can be slightly irritating – to say the least! Sometimes, when you export a SWF to the 'WindowSWF' directory and attempt to open the panel within Flash MX 2004, a different panel opens!

There is apparently no workaround for this – you simply have to change the name of the SWF until it opens the correct panel when you select the panel from Window > Other Panels > [Your Panel]. To me, it looks like Flash MX 2004's built-in directory parsing uses a simple regular expression to iterate through the directory, and it can easily get confused! Hopefully, this will be rectified in the next minor (or major) release of Flash MX 2004.

figure 4

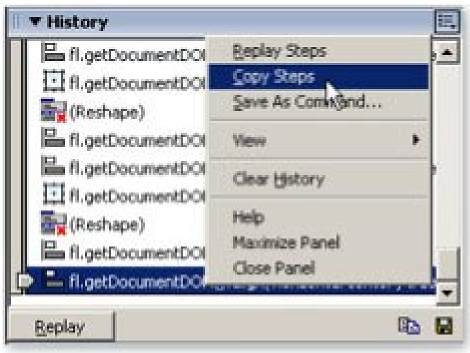
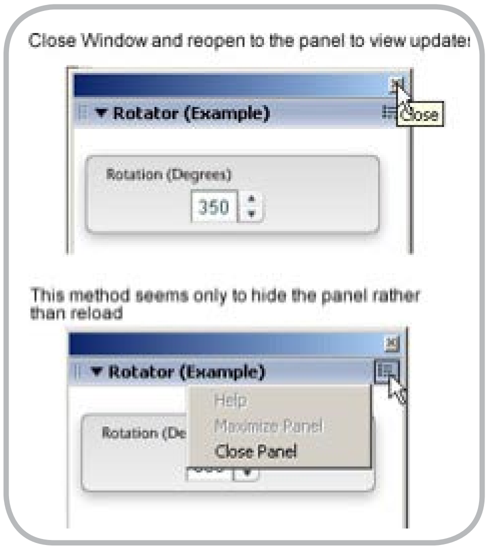


figure 5


Close Window and reopen to the panel to view update!

This method seems only to hide the panel rather than reload

Package Your Panel

In order to make your shiny new panel easily shareable, you need to create an MXP file that can be installed with the Macromedia Extension Manager. The first step is to create an MXI file that the Extension Manager can use to compile the MXP file. The MXI is essentially an XML file that contains simple information about the extension: version information, extension name and description, as well as the files to compile.

Note that an example .mxi file is included within the article source code, so you can alter it for your needs.

Although it's outside the scope of this article to describe all the options available to those creating distributable MXPs, I'll cover some of the basics here to get you started.

In order to create an MXI file for the Flash Panel we've just created, open your favorite text editor and add the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<macromedia-extension
  name="Rotator Panel for Flash MX
2004"
  version="1.0.1"
  type="Command"
  requires-restart="true">
 <author name="Phireworx" />
 <products>
   <product name="Flash" version="7"
primary="true" />
 </products>
 <description>
 <![CDATA[
 Happily rotate your objects in Flash
MX 2004 using this Simple     Panel
 ]]>
 </description>
 <ui-access>
 <![CDATA[
 Access to the command panel is by
selecting 'Window > Other Panels
> Rotator Panel' in Flash MX 2004.

 ]]>
 </ui-access>

 <license-agreement>
 <![CDATA[
 ]]>
 </license-agreement>
 <files>
 <file source="Rotator Panel.swf"
destination="$flash/WindowSWF" />
 </files>
</macromedia-extension>
```

The MXI file contains different information, all of which can be easily understood and edited to suit your own needs. Here's a quick overview of where the information is located:

- ***Author Name:*** Within the <author> tag name attribute
- ***Description:*** Within the <description> tag
- ***Access and Usage Instructions:*** within the <UI-access> tag
- ***Source File:*** within the <file> tag source attribute
- ***File Destination:*** the location at which you should install the file is within the <file> tag destination attribute

The most important section is the name of the SWF file that we are going to add:

```
 <file source="Rotator Panel.swf"
destination="$flash/WindowSWF" />
```

We simply place the name of the exported SWF into the 'file source' section, and add the 'WindowSWF' directory as the destination ($flash/WindowSWF).

Note that the name of the exported SWF file that you include within the extension will appear as it does in the Flash MX 2004 menu system under 'Other Panels'.

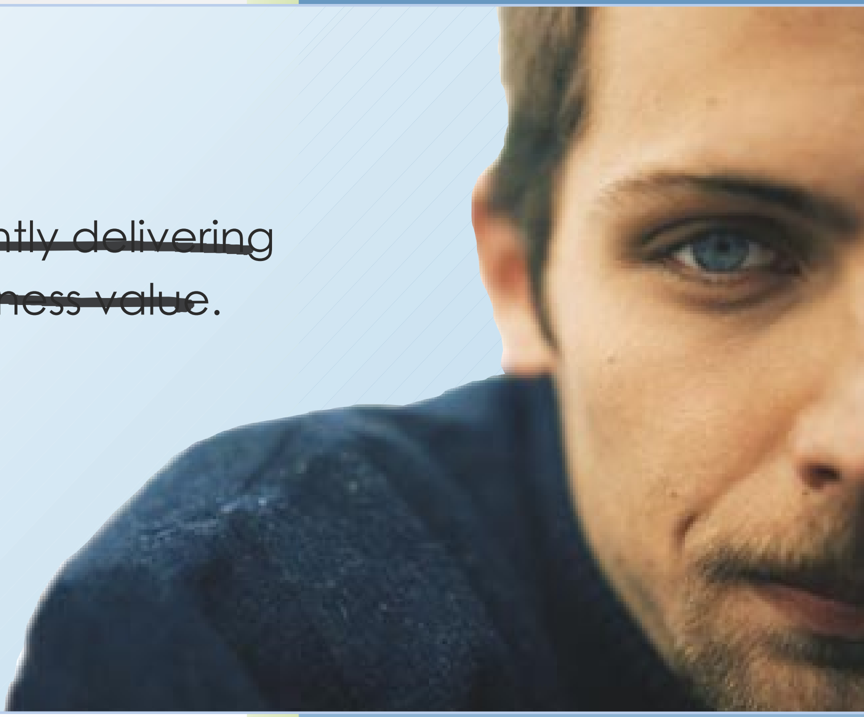
Once you've edited the options to your needs, save the file with the extension .mxi (e.g. Rotator Panel;.mxi).

Now, you can double-click the MXI file, and (if Macromedia Extension Manager is installed), you'll be prompted for an extension (MXI) to package. You'll also be asked for a name by which the extension package (MXP) can be saved. The Macromedia Extension Manager automatically creates the MXP file, which can then be distributed as you see fit!

I've only skimmed the surface of creating your own custom Flash Panels in this article, but I certainly hope that this information has given you the incentive to create your own Flash Panels! Don't be afraid to experiment with your own cool effects and ideas for panels and commands.

Thi~~nk Java development is complex & overengineere~~d?

~~Think **again**~~.



I ~~want to be known for consistently delivering applications and providing business value~~.
~~That is why I use ThinkCAP.~~

ThinkCAP™

ThinkCAP 6.0 simplifies and accelerates the development and maintenance of J2EE-based web applications by 50%.

~~ThinkCAP's Visual Workbench and integrated MVC framework bring high productivity to corporate application developers (those with PowerBuilder, RPG, or Oracle-like skills) while letting J2EE programmers build reusable business logic & components using their tools of choice. The result is productivity for the whole development team~~.

~~Because of features such as Smart Data Binding, Visual Page Flow, and seamless Hibernate/Castor support, your team focuses on value added functionality—not low-level "plumbing." ThinkCAP integrates over 20 open-source projects providing a practical mix of high productivity, robust functionality, and standards support~~.

Download a free trial version or view a demo at www.clearnova.com.

CLEARNOVA
Deliver Web Applications at the Speed of Business

~~Highly Productive Visual Workbench~~

~~Actions-based MVC Framework~~

~~Page Flow Designer & Generator~~

~~Easy to Learn Event Model~~

~~Advanced Data Aware Components~~

~~Forms, DataViews, Queries, Navigations~~

~~Workflows, Graphs, Treeviews, Grids, Tabs~~

~~Multirow Updatable DataViews~~

~~Extensible Browser & Server-side Validation~~

~~Point & Click Smart Data Binding™~~

~~Service Flow Designer for Web Services/SOA~~

~~Content Management & Workflow Engine~~

~~Platform Independence: Any Java Server/OS/IDE/Database~~

~~Integrated Role-Based Security~~

~~Seamless Integration with over 20 Open Source Libraries~~

~~Team Development with CVS Support~~

TagCloud Flash Integration

Easy, breezy, beautiful XML

by eric e. dolecki

tagCloud is an automated Folksonomy tool. Essentially, TagCloud searches any number of RSS feeds you specify, extracts keywords from the content and lists them according to prevalence within the RSS feeds. Clicking on the tag's link will display a list of all the article abstracts associated with that keyword.

Previously the only way to get TagCloud feeds to display on your website, etc. was to include a bit of javascript (as an include) in your page. Which meant you have almost no control outside of CSS on how that data was presented. It worked, it just wasn't as flexible as some might like. Now, you're able to get at any particular feed using XML as your data source!

Part of the reason I used Flash to render my XML data was the fact that I couldn't supply a target for the links coming out of the javascript fetch. So if I had access to all the data myself, I could simply add my targets. Then I was told by John Herren about an upcoming XML service from TagCloud that I could play with and test. With that, I built my application, and thought it would be a good thing to let others build theirs if they wanted to. This sample application is very simple, but if you're fairly proficient in Flash, you could build some pretty wild UI.

This tutorial assumes you have a copy of Flash MX 2004 Pro installed. We use the Data Binding Component and also need to publish to Flash 7.

Figure 1 is an example screen shot of what the result of this quick tutorial will allow you to put into your webpage:

To see a live implementation of this application, visit my website (http://www.ericd.net/new_css/) where you'll see it at the top of my blog section.

Getting Started

I will assume you already know how TagCloud works and how you can place it on your website. If not, please go to TagCloud (http://www.tagcloud.com/) and read up on it. It's pretty simple.

TagCloud can now supply XML from a URL request. For instance http://www.TagCloud.com/cloud/xml/ericd/default/22.

If you open that URL in Internet Explorer, you'll see the XML as it is returned to the browser from TagCloud. I previously set that cloud to include the Fullasagoog.com flash feed. The trailing 22 in the URL means to return 22 matching keywords.

If you open the URL in Safari RSS, you'll see a lot of data. View Source on the page to see the XML.

The Returned XML

Here's how the returned XML is formatted (I am not showing you all of the XML here - you should get the idea).

So, you see all the data you need is there to access. Each returned keyword is represented as a <Tag> node. And each has a Name, Scale, and Link. The Scale indicates the prevalence for an individual keyword (more of those keywords equals a larger value). The Link is a URL where you can view the items that were matched using that keyword.

All we need to do now is to create a tiny Flash application that will fetch that information and display it.

Creating the Flash Application

I have included 6 files in this tutorial source code release (see source code link for this article). You can either rifle through those files to learn on your own, or follow along to get things up and running, and explained how the application works.

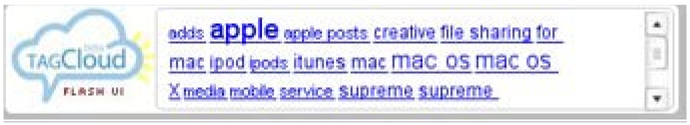
I'll talk about that proxy file in a bit.

Open the FLA or you can start from scratch by recreating this simple layout:

All you really need to be concerned with is having a textfield set to HTML, multiline. To the right of that is a UIScrollbar component - set to scroll that text field instance. Give the dynamic text field an instance name of "links" for now. All we are going to do is load the XML, parse the data, format the data, and present it as hyperlinks.

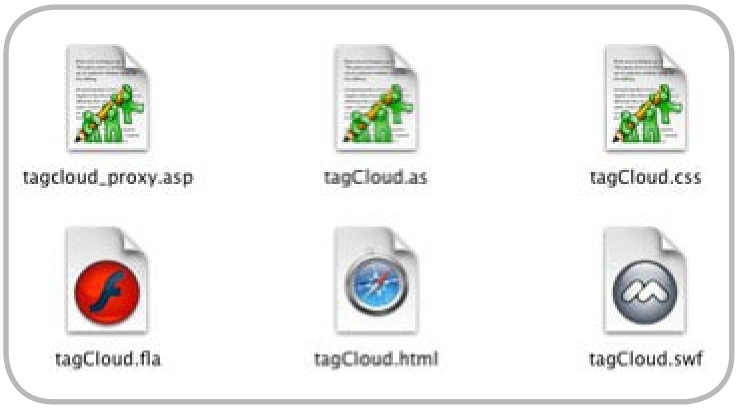
We're going to make parsing the XML a lot easier than what you may have become accustomed to with typical looping techniques. Flash 7 comes

figure 1



adds apple apple posts creative file sharing for mac ipod ipods itunes mac mac os mac os X media mobile service supreme supreme

figure 2



with a basic implementation of XPath. Xfactorstudio has some great XPath libraries, but for this application I think it would be overkill. If you haven't heard of XFactorStudio's XPath implementations for ActionScript, you should check them out.

Anyway, we need to drop a DataBinding component on the Stage so it can be used. Where can you find it (it's not listed with your normal components)? Look here (the Classes Library):

When you open that Library, you should see something like this:

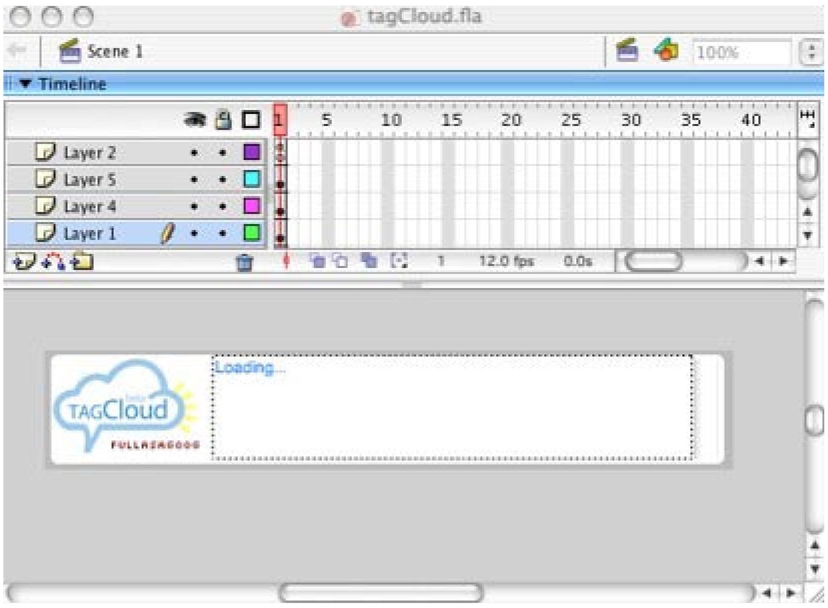
Drag that DataBindingClasses component to your Stage, then delete it. This places the component in your own FLA's Library - ready for use.

ActionScript

We'll jump right into the ActionScript that gets the data, and then displays it. I'll walk you through it below the code block. Not much code, is there?

```
import mx.xpath.XPathAPI;

var myCSS = new TextField.
StyleSheet();
var cssURL = “http://www.ericd.net/
new_css/sections/TagCloud.css”;
myCSS.onLoad = function( success ){
    if (success){
        links.styleSheet = myCSS;
    }
};
myCSS.load( cssURL );
var rssfeed_xml = new XML();
rssfeed_xml.ignoreWhite = true;
rssfeed_xml.load(“http://www.ericd.
net/new_css/sections/TagCloud_proxy.
asp”);

var statement:String = “”;
var sc:Number;

rssfeed_xml.onLoad = function(success)
{
    if (success) {

 var namePath:String = “/Cloud/Tags/
Tag/Name”;
 name_array = mx.xpath.XPathAPI.
selectNodeList(this.firstChild, name-
Path);

 var scale:String = “/Cloud/Tags/Tag/
Scale”;
 scale_array = mx.xpath.XPathAPI.
selectNodeList(this.firstChild,
scale);

 var titlePath:String = “/Cloud/Tags/
Tag/Link”;
 title_array = mx.xpath.XPathAPI.
selectNodeList(this.firstChild, title-
Path);

 links.text = “”;
        for (var i = 0; i<title_
array.length; i++) {

            sc = Number(scale_
array[i].firstChild.nodeValue);

            switch( sc ){

            case 1:
                sc = 12;
                break;
            case 2:
                sc = 14;
                break;
            case 3:
                sc = 15;
                break;
            case 4:
                sc = 16;
                break;
            case 5:
                sc = 18;
                break;
            case 6:
                sc = 19;
                break;
            case 7:
                sc = 20;
                break;
            case 8:
                sc = 22;
                break;
            case 9:
                sc = 24;
                break;
            }

            // The line below should
be continuous... it is on
            // multiple lines here in
the tutorial for ease of display

            statement += “<a href='”
+ title_array[i].firstChild.nodeValue
+
“' target='_blank'><font size='” + sc
+ “'>” + name_array[i].firstChild.
nodeValue + “</font> “;
statement += “ “;
        }

        links.htmlText = statement;

    } else {
        trace(“error loading XML”);
    }
};

                /* Note: instead of
using a switch (I used a switch so it
might be easier
                    to follow
along visually for those newer to
ActionScript), you could do
                    this outside the
for loop below:
```

figure 3



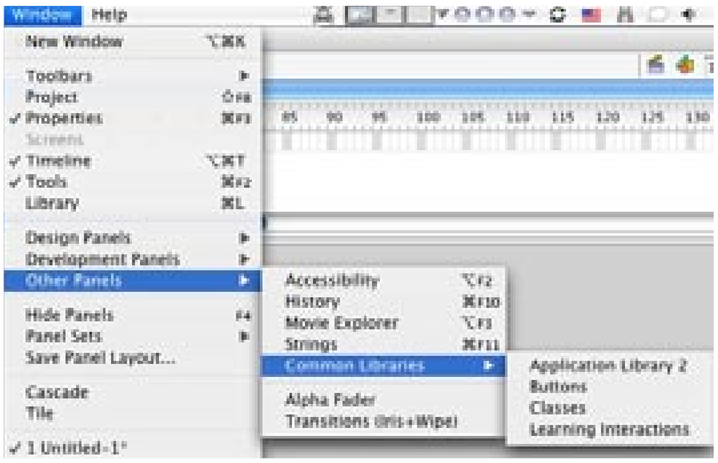


figure 4

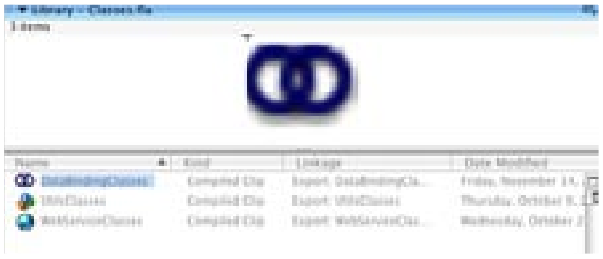


figure 5

Eric E. Dolecki is currently an Interactive Developer/Designer at Convoq, Inc in Boston, MA. He has personally won Macromedia Site of the Day twice, has been published in several prominent magazines, has co-authored numerous books on Flash-technology, and has won several international Flash awards. He maintains his personal site (www.ericd.net) where he showcases experiments, shares information, and offers help to those seeking to learn more about Flash and interactive technology. You can subscribe to Eric's new podcast within iTunes by using this URL:http://feeds.feedburner.com/Version60.

ericd@ericd.net

```
var scales = [0,12,14,15,16,18,19,20
,22,24];

                                and then
inside the loop:

                             sc =
Number(scale_array[i].firstChild.node-
Value);
                             sc = scales[sc];
// redefining the scale to our liking
(pixel size)

                             We could also
use object properties to increase the
lookup speed,
                             but there aren't
enough possible scale values to worry
too much about that.
                             *This comment
block is not in the source release.
                         */
```

Set Up XPath

The first thing we do is include the XPathAPI Class so that we can easily access XML data. You need the DataBindingClasses component in your Library for this to work.

Loading Our CSS

We then want to give the links hover states. We do this through using CSS in Flash 7. You'll see in the code that we load the css file and associate it with the dynamic HTML field set to HTML. You'll see the simple CSS that is loaded in the source file I have included. This takes care of coloring and styling any hyperlinks in that HTML field.

Proxy File XML Load

After the CSS loading code, you'll see where set up our XML load. We have to use what is known as a proxy file to access the data from a different domain than your own. TagCloud does not have a policy file on their server... security in the player keeps you from gaining access to other server's data without consent. So we use a proxy file to trick the player into loading the data we need.

I have included an ASP-flavored proxy file in the source for this tutorial. You can e-mail me at ericd@ericd.net if you have questions.

Parse XML Data

When the XML loads, we grab data and place it into 3 buckets (arrays). namePath will contain an array of the data found in the Name tag for each Tag node.

```
var namePath:String = "/Cloud/Tags/
Tag/Name";
name_array = mx.xpath.XPathAPI.
selectNodeList(this.firstChild, name-
Path);
```

This is the beauty of Xpath... look how we access that data. We use a path directly to the information we need in the XML (each time its found, it adds to an array which we set up). So every node value for XML elements matching the path are included. You don't have to set up cumbersome loops.

We do the same thing for Scale and for Title. After we build those three arrays, we clear the HTML textfield just to be safe.

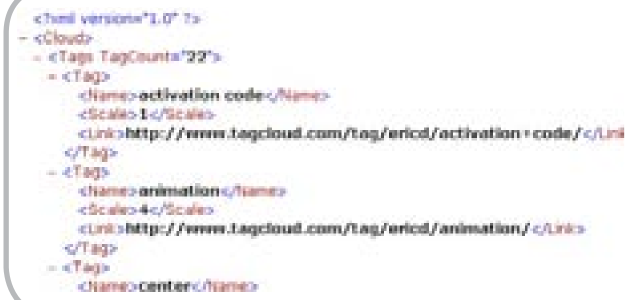
Build String to Display

We now run a loop based on the number of items in the title_array... in order to build up a master string which we will set to the HTML field. For each tag, we take its scale value, and we run a switch on it... setting its font size to whatever we'd like. This is how the links with more emphasis are larger than those with less emphasis. You can see where we build up the string with each loop... each Tag added to the string until we're looped through them all.

We then set the field to the value of that string, the CSS takes over the hover links and appearance for us, and you're done.

Conclusion

I hope you found this tutorial useful - and I hope you noticed that getting a system like this up and running is actually pretty easy when you do it with Flash.



```
  <?xml version="1.0" ?>
- <Cloud>
  - <Tags TagCount="22">
    - <Tag>
        <Name>activation code</Name>
        <Scale>1</Scale>
        <Link>http://www.tagcloud.com/tag/ericd/activation+code/</Lin
      </Tag>
    - <Tag>
        <Name>animation</Name>
        <Scale>4</Scale>
        <Link>http://www.tagcloud.com/tag/ericd/animation/</Link>
      </Tag>
    - <Tag>
        <Name>center</Name>
```

listing 1

Five Flash Tips

Guy Watson shares a few quick hints

by guy watson

Tip #1: Neat Fix - onLoad Bug

I have heard so many people moan on the many community forums and mailing lists about the onLoad event handler not working with loadMovie and although I have already posted a fix for the problem (http://www.actionscripts.org/forums/showthread.php3?threadid=13830), I found this very small and functional fix that does the same thing, a while ago on the Flashcoders Mailing List: http://chattyfig.figleaf.com/mailman/htdig/flashcoders/2002-October/049415.html and so if you see the question, forward people to http://chattyfig.figleaf.com/apache2-default/.

Tip #2: Bug - Sound.loadSound & GetRight

If you have GetRight (http://www.download.com/GetRight/3000-2071_4-10005533.html) installed on your computer, a download management utility, and you view a Flash movie that attempts to load an mp3 sound file into the Flash Player dynamically using the Sound.loadSound method, GetRight stops the loading of the mp3 file and asks you to choose a location to save the mp3 file on your local computer; whether you choose to download the file or not, Flash still does not load the file into the player, which raises an issue.

GetRight is one of the most, if not the most popular download utility and this is the default behaviour of the program, of course these settings can be changed, by removing the hook for mp3 files but Flash alone cannot do this automatically for you, this is a manual process that viewers of your site will have follow if they want to listen to dynamically loaded sounds on your site...

To fix the problem, I assume that by changing the file extension of your mp3 files to something which is not registered as a download hook in GetRight will do the trick, but of course, if GetRight actually checks the MIME -type of the file instead of using the files extension, then this will not solve the problem. Changing the mime-type is not only a complex process for novices, it will not solve the problem, because the Flash Player will not play the file if its MIME-type is not set to type mp3.

Tip #3: Stage.showMenu

It is all in the name, Stage.showMenu is a Boolean property of the Stage object that defines whether to display the menu or not, when the user right-clicks on your Flash movie.

A boolean value is either true or false, 0 or 1, on or off, the Stage.showMenu value can either be true or false:

```
true - Show the menu on right-click
false - Down show the menu on right-
click
```

Please bear in mind, that this isn not the answer to all our prayers, when you turn the right-click menu off, the 'About Flash 6 Player' and 'Settings' options are still displayed, so no, you cant 'fully' remove the right-click menu. The actionscript:

```
//turn off the right-click menu
Stage.showMenu=false;
```

Tip #4: System.security.allowDomain

This undocumented method grants permission for the specified domain(s) to access objects and variables in the Flash movie that calls the allowDomain command. This property had to be implemented because of the changes made to the Security Sandbox in the Flash 6 Player...

I have two movies, 'a.swf' and 'b.swf', 'a.swf' is located on www.flashguru.co.uk server and 'b.swf' is located on www.macromedia.com server, for 'a.swf' to be able to access the contents of 'b.swf', 'b.swf' has to make a call to the allowDomain command as follows:

```
System.security.allowDomain("http://
www.flashguru.co.uk");
```

The domain can be formatted in numerous ways, the same different formats that the loadVariables, loadVariablesNum, xml.load and loadvars.load methods accept: http://www.domain.com

```
http://domain.com
domain.com
http://IpAddress
```

You can pass multiple domains to the method as arguments. This method has been documented in the Macromedia Flash MX ActionScript Dictionary Errata (http://www.macromedia.com/support/flash/documentation/flash_mx_errata/flash_mx_errata03.html).

Tip #5: Test Player - System Keys

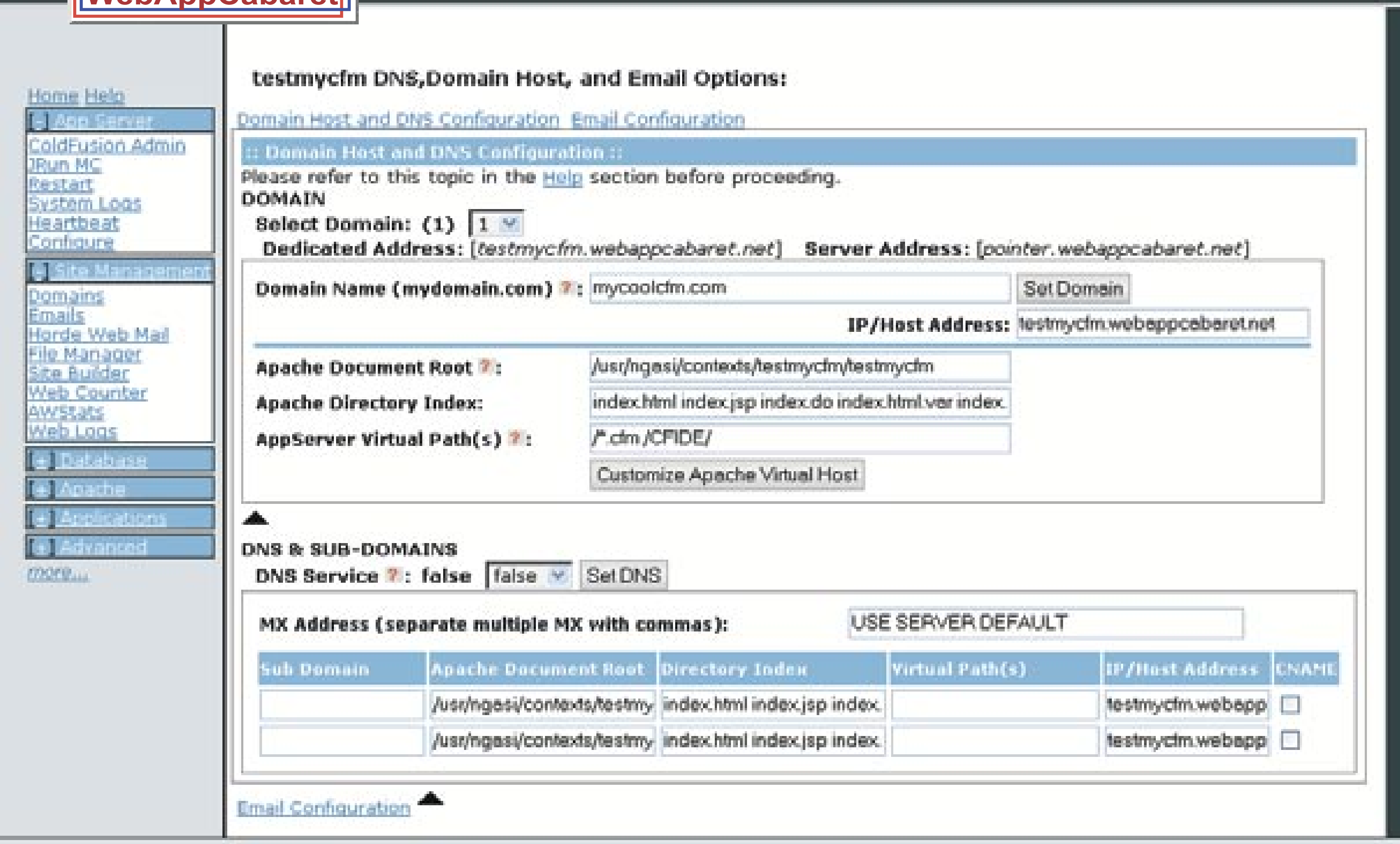
By default, in the local test player(Control > Test Movie), if you try and press the enter key, to test some code that uses it, your code will not work because the player over-rides the key press as a keyboard shortcut to pause and play the movie, the same goes for the tab key and other important system keys, such as Ctrl, Shift etc... So if you want to test your tabIndexes in the test player, so you can use trace and the debugger, simply select Disable Keyboard Shortcuts from the Control menu (Control > Disable Keyboard Shortcuts) and your code will now recieve the key presses. Saves me a lot of faffing around! Think this is something else I was slow on, these little fine details just fly over my head.

Guy Watson, known to some as FlashGuru, is an active member of the Flash Community and a well respected industry leader. He has won numerous industry awards for his work and is regularly invited to hop around the world and deliver presentations at various industry events.

guy@flashguru.co.uk

WebAppCabaret sm

ColdFusion MX 6.1 Enterprise Hosting made easy.



testmycfm DNS,Domain Host, and Email Options:

Domain Host and DNS Configuration Email Configuration

:: Domain Host and DNS Configuration ::

Please refer to this topic in the Help section before proceeding.

DOMAIN

Select Domain: (1) 1

Dedicated Address: [testmycfm.webappcabaret.net] Server Address: [pointer.webappcabaret.net]

Domain Name (mydomain.com) ?: mycoolcfm.com Set Domain

IP/Host Address: testmycfm.webappcabaret.net

Apache Document Root ?: /usr/ngasi/contexts/testmycfm/testmycfm

Apache Directory Index: index.html index.jsp index.do index.html.var index.

AppServer Virtual Path(s) ?: /*.cfm /CFIDE/

Customize Apache Virtual Host

DNS & SUB-DOMAINS

DNS Service ?: false false Set DNS

MX Address (separate multiple MX with commas): USE SERVER DEFAULT

Sub Domain	Apache Document Root	Directory Index	Virtual Path(s)	IP/Host Address	CNAME
	/usr/ngasi/contexts/testmy	index.html index.jsp index.		testmycfm.webapp	☐
	/usr/ngasi/contexts/testmy	index.html index.jsp index.		testmycfm.webapp	☐

Email Configuration

Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with ColdFusion MX. At WebAppCabaret our standards based process and tools make deploying ColdFusion MX applications as easy as a point-and-click.
We call it **Point-and-Deploy Hosting**.
Our advanced NGASI Web Hosting management Control was designed for the hosting and management of ColdFusion web sites and applications thus cutting down on maintenance time.

Backed by an experienced staff as well as a **Tier 1 Data center** and network. Our network is certified with frequent security audits by reputable security firms.

All ColdFusion hosting plans have separate and individual installation AND instances of ColdFusion MX 6.1 Enterprise with **full access to ColdFusion Administrator** and JRun Management Console; so there is virtually no restriction or customization required for your application.

Complete power and control at the tip of your fingers. We take care of the system and hosting infrastructure so you can concentrate on development and deployment of your application. That is **real ROI**.

Log on now at http://www.webappcabaret.com/cdj.jsp or call today at 1.866.256.7973

NGASI POWERED

WebAppCabaret sm

http://www.webappcabaret.com/cdj.jsp 1.866.256.7973

Point-and-Deploy J2EE Hosting

SUBTLE
MULRITHREADING

No matter how powerful a computer you or your end user has, faster software is always desirable. For one, we live in an era where time is very valuable and technology is here to empower us, not to set limits on what we can or wish to do. Furthermore, the responsiveness of an application to its user is a crucial benefit and increasingly important differentiator between ordinary software and superior software. Responsiveness of software is generally defined as its ability to react fast, to provide guidance and feedback and to allow users to cancel or redirect time-consuming operations. Conversely, an application that temporarily freezes and does not provide progress information while it is processing tasks that take as little as a few seconds, does not convey the sense of responsiveness, care and respect users expect, and in addition interrupts the uer's concentration when performing tasks such as learning, selecting a product from an electronic catalog, or just playing a game .

by laurent brigaut

Modern operating systems, Windows and Mac OS alike, implement a mechanism called multithreading that allows multiple tasks to take place simultaneously , thus interacting with users while processing other CPU or network intensive tasks.

This article will show how you can improve the responsiveness of your Director based applications, through the use of multithreaded Director Xtras. Implementing multithreading in a Director Xtra has been a commonly asked question for the seven years I have been on Macromedia's XDK forum.

To quote Albert Einstein "For every problem there is a solution which is simple, obvious, and wrong". Time and again, I've seen poor solutions suggested on the XDK forum to address the multithreading issue. These solutions aren't the best and in some cases, will even crash your project. This is what has inspired me to write this article.

Example 1: A Board Game Application

Let's assume you are building a board game such as chess, or Othello-Reversi, or any other board game that requires your opponent (the computer) to "forward think" multiple scenarios. Such an application typically requires CPU-intensive and time-consuming dynamic programming or backtracking algorithms. Let's further assume you want your game to stand out from the crowd: it must provide an animated progress indicator providing the user with feedback on how long it will take before the computer's next move. Ultimately, you may also want to give users the ability to force the computer to "Play Now" with the best move it has at the moment. To do so, your best bet is to code the game's "intelligence" in a multithreaded Xtra. Better yet, assuming that you can reuse an existing game engine (a library or application) that implements the required intelligence, you only need to bridge your Director user interface with the engine using a multithreaded Xtra.

Any time-consuming computation problem can be addressed in the same way as the game example. A few similar scenarios could be:

- Scientific calculations involving large matrices or complex algorithms
- Optimization problems in logistics, manufacturing, transportation and other areas
- Visualization applications with intensive graphic processing
- Real-time data compression / decompression

Example 2: Networking in Macromedia Director

Another context where multithreading plays a key role is networking in Macromedia Director. Assume that your Director movie must asynchronously send and receive information from a remote server, say your corporate ERP. Asynchronous means that the said process can accept a request, return control to the caller and then notify the caller again when it completes. In contrast, in a synchronous call, the said process would hold the calling request, perform its processing, get the result and send it back to the originator by way of the calling function. The user interface would look frozen in the meantime.

Although NetLingo is able to perform asynchronous network operations, your Director code must keep looping and checking the status of such an operation so it can provide the user with feedback and perform an action once the operation has been completed. The code would also need to properly handle error and timeout scenarios. This mechanism is called polling. It is further discussed below.

Any task that can be delegated to an external CPU falls in this category:

- Reading and writing data to a corporate server, such as an ERP, CRM, LMS, etc.
- Tapping into custom-made enterprise servers built with Java, .NET, etc.
- Integrating your project to a chat server, a database server, a web service, etc.

What Is Multithreading?

One can easily understand how a computer linearly executes a sequence of instructions, one after another. Any commercial-grade CPU can only sequentially process one such instruction at a time. However, multithreading implies that multiple actions simultaneously take place on the same computer. How can this be?

Both of the above statements are true as long as you agree to abstract the concept of time. In the wonderful world of multithreading, your application and the CPU have different concepts of time. Your program only deals with the Operating System and thus assumes that its threads are executed simultaneously. However, while the Operating System makes you believe this is true, down under, it juggles with setting priorities on threads and has the CPU process short sequences of each so they appear to be simultaneous to you.

Polling: Simple but Inefficient

A simple solution to providing users with an interface that remains active while performing other tasks is to have your Lingo code continuously poll the status of the external tasks, possibly displaying progress information, and performing a completion action in due time.

For example, if the computer must calculate the next move of a board game, your Lingo code would call your Xtra, which would create a new thread (let's call it the "secondary thread"), and call the game engine with the appropriate parameters. Your Xtra would immediately resume in the main thread and return control to Director. Then, your Lingo code would need to continuously loop On EnterFrame (or On Idle, or something similar) and check a status method (let's call it GameBusy()). As long as GameBusy() returns true, your Lingo keeps looping, updating progress information, managing timeouts and error conditions, and trying to respond to any other user requests. In the mean time, the secondary thread provides enough CPU time to the game engine to compute the next move. Whenever GameBusy() returns FALSE, you exit the loop, call another method to retrieve the computer's next move and display it to the user.

Although polling works fine in simple projects, it has two major drawbacks:

1. It wastes valuable CPU time.
2. It makes your code harder to write, to read, to debug, to update and to reuse in future projects.

Thus, the polling solution is detrimental to both your users (who experience slower software execution) and you (who has to spend more time coding and debugging).

Brutal Multithreading: Simple but Wrong
An obvious way to optimize the above-mentioned polling mechanism is to replace it by a notification mechanism.

For a real-world example, imagine that you place an order at your local electronic store for the latest super-cool MP3 player, but the product is currently out-of-stock. The sales associate however promises they will receive it within the next 10 days. One possible -- but inefficient -- way for you to get your hands on the product is to call the store every hour or so to find out whether or not it has arrived. This would waste a lot of time for both you and the store and is what you would define as a polling mechanism.

Another and better mechanism would be for the sales associate to phone you once they receive the product. This is a notification mechanism and as you can see, is far more efficient. We will take this mechanism and adapt it to our game board example.

Imagine the following context: It is the computer's turn to compute the next move. Your Lingo code calls your multithreaded Xtra, which creates a new thread (the "secondary thread"), which calls the game engine with the appropriate parameters, and returns control to Director and your Lingo code in the main calling thread. While the game engine is computing its best next move in the secondary thread, you are free to make your Lingo code do whatever you feel is appropriate, for example you could display an animation, respond to user interactions, etc. Your Lingo code wouldn't need to be different than any other ordinary Lingo code and thus can be architected exactly the way you want it. Director and your Lingo code would obviously run in the main thread.

When the game engine completes its computation, it would then call a handler in Director which you would have designated to take over after the next move's result is delivered. The game engine would do this in the secondary thread as it would not have access to the main thread at this point.

Unfortunately, this approach would inevitably lead to a crash in Director. This is because Director is not protected against reentrancy: you cannot call a Director method from anywhere else other than the main thread. You would have to find a way to catch its main thread first.

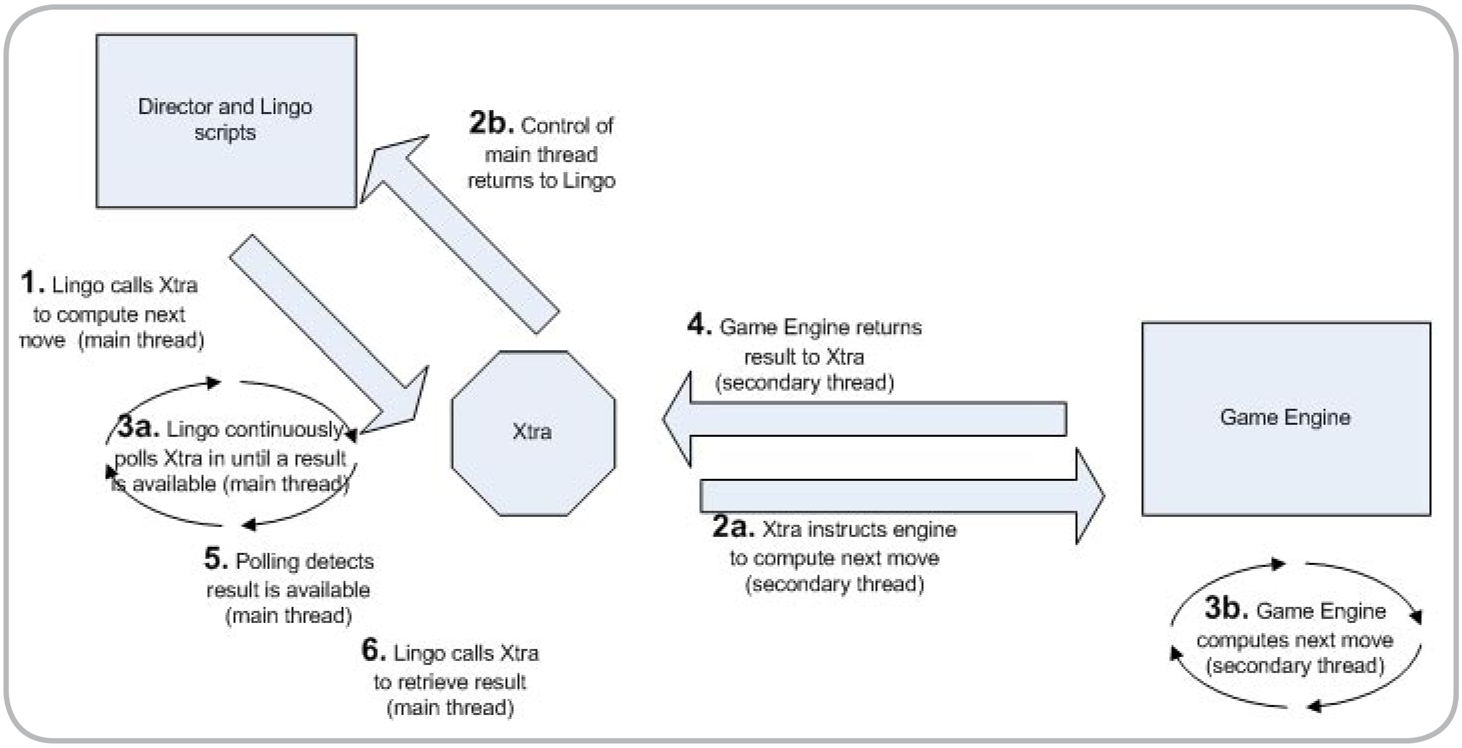
To illustrate this mechanism with our real-world example, having the secondary thread directly call the completion handler in your Director movie would be equivalent to the sales associate calling to notify you that your MP3 player has arrived, and expecting you to take delivery right away even though you may be in the shower at that time of the call and not in a position to instantaneously respond to his or her call.

Subtle Multithreading: the Producer/Consumer Model

Let's improve the notification mechanism introduced above. Let's have the sales associate call and leave a voicemail informing you that your MP3 Player has arrived. This would allow you to retrieve and respond to the voicemail in the order that best suits your schedule.

To do the above in a software program, you would need to implement a Producer/Consumer communication model. The main thread (Director) would be the Consumer. The secondary thread would be the Producer. There would be a FIFO (First-In First-Out) message queue to coordinate their efforts. The message queue works exactly like the answering machine in our real-world example between the sales associate (the Producer) and you (the Consumer). When the Producer has something to signal to

figure 1


Director and Lingo scripts
2b. Control of main thread returns to Lingo
1. Lingo calls Xtra to compute next move (main thread)
4. Game Engine returns result to Xtra (secondary thread)
Game Engine
3a. Lingo continuously polls Xtra in until a result is available (main thread)
Xtra
2a. Xtra instructs engine to compute next move (secondary thread)
5. Polling detects result is available (main thread)
3b. Game Engine computes next move (secondary thread)
6. Lingo calls Xtra to retrieve result (main thread)

the Consumer, it adds it to the message queue. Every time the Consumer has spare time, it checks the queue to see if something is waiting for it. When it finds a message, it retrieves it and processes it.

In our board game example, this translates to the following sequence: When it's the computer's turn to make a move, your Lingo calls the Xtra which immediately returns control to your Lingo. While your Lingo code is free to interact with the user, the Xtra spawns a secondary thread and sends it to the game engine for processing. When the game engine is ready with its next move, it posts the result to the message queue using the secondary thread (the only thread it has access to). The main thread then picks up the message and processes it.

While this mechanism is only explained in the context of the completion of a computation, it equally applies to the retrieval of the progress of a task or its error condition.

Making It Bullet-Proof

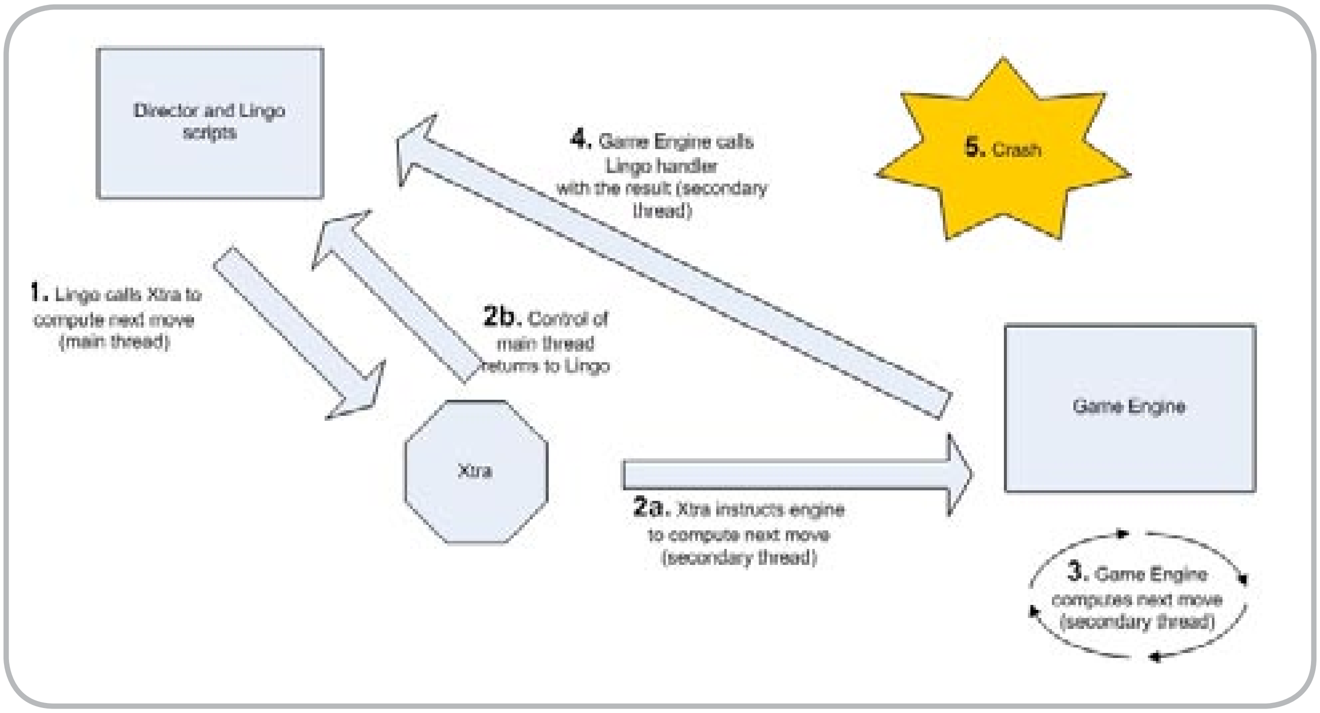
To make sure your threads work harmoniously together, you have to make sure the following conditions are met.

- ***Memory Sharing:*** The queue is a memory zone shared by multiple threads. You must make sure that the threads don't modify the queue at the same time. Luckily, operating systems implement the concept of Critical Section to protect you against this problem. A Critical Section is a portion of the program that is executed continuously by a thread without being interrupted by any other thread. Thus, to make sure that one thread is not retrieving a message from the queue and reorganizing memory while the other thread is still posting the same message – and probably reorganizing memory too –, you must include each message addition and retrieval call in a Critical Section.
- ***Controlling Queue Size:*** If the Producer adds messages to the queue faster than the Consumer can process them, your queue will indefinitely grow. That would lead to uncontrolled memory consumption, excessive memory reallocations, excessive virtual memory use resulting in excessive stress on hard disk usage and CPU. The immediate symptom would be the extreme slow down of your computer with a complete crash not too far away. If you suspect that such a problem may arise in your specific application, you should consider implementing another multithreading mechanism called Semaphore. A Semaphore is a way for multiple threads to share a limited number of resources. In this case, the resource is the memory used by the message queue and a Semaphore can be used to limit the number of items contained in it.
- ***Avoiding Deadlocks:*** A deadlock occurs when two threads wait for each other and make it impossible for each other to resume activity. Deadlocks are often caused by a misuse of Semaphores and Critical Sections. For example, if the queue is full and the Producer tries to add a message, it first enters a Critical Section to acquire exclusive access to the queue. Then it checks the Semaphore to make sure it can indeed add a message. Since the queue is full, the Producer exits without adding the message, hoping to be able to do so in a later iteration, after the Consumer has removed one or more messages from the queue. A fraction of a second later, the Consumer tries to retrieve a message from the queue: It first tries to acquire exclusive access to the queue through a Critical Section but fails because the Producer already has acquired exclusivity through its own Critical Section. Consequently, the Consumer is unable to remove a message from the queue. This obviously creates a deadlock between Producer and Consumer.

A Sample Implementation

This section introduces a sample

figure 2


Director and Lingo scripts
4. Game Engine calls Lingo handler with the result (secondary thread)
5. Crash
1. Lingo calls Xtra to compute next move (main thread)
2b. Control of main thread returns to Lingo
Game Engine
Xtra
2a. Xtra instructs engine to compute next move (secondary thread)
3. Game Engine computes next move (secondary thread)

Are you using PDF technology in your daily business?

Do your requirements go beyond the standard features offered by your current PDF products?

PDF Tools AG has spent years addressing your needs and is proud to offer the newest line of PDF tools designed for customers relying on PDF technology for their critical business processes:

The 3-Heights™ family of PDF programming components

PDF Tools AG is a world leader in PDF programming technology, delivering reliable PDF products to international customers in virtually all market segments.

PDF Tools AG provides server-based software products designed specifically for developers, integrators, consultants, customizing specialists and IT-departments. Thousands of companies worldwide use PDF Tools AG's products directly and tens of thousands more use the technology indirectly via our growing network of OEM partners. The tools are easily embedded into application programs and are available for a multitude of operating system platforms.

pdf-tools.com
Premium PDF Technology

Contact:
pdfsales@pdf-tools.com

or visit us at:
http://www.pdf-tools.com

implementation of a multithreaded Xtra through a simplified model. In this Xtra, the Producer is a simple thread that counts from 1 to 1,000, and pushes the value of the counter into the message queue. The Consumer retrieves the value and calls a method in Lingo and displays it in Director.

We only show the Windows version of this Xtra here, but the Mac OS version would be almost identical. Also, for the sake of clarity, the error checking code has been stripped out so that the essential part of the code is more apparent.

Let's first look at the queue manager. This manager is responsible for adding (pushing) and retrieving (popping) messages. We use STL (C++'s Standard Template Library) to represent the queue. The elements in this queue are objects of type CEvent, which is a virtual class and therefore can contain any type of structure.

Adding a message to the queue:

```
// Make sure that the queue did not
reach its maximum size
// This call will block until the
queue has some space available
WaitForSingleObject(mSemaphore,
INFINITE);

// Request exclusive access to the
queue
// This call will block if the queue
is already being accessed elsewhere
EnterCriticalSection(&mCriticalSecti
on);

// Add the event to the queue
mEventQueue.push(theEvent);

// Signal that we no longer require
exclusive access
LeaveCriticalSection(&mCriticalSecti
on);
```

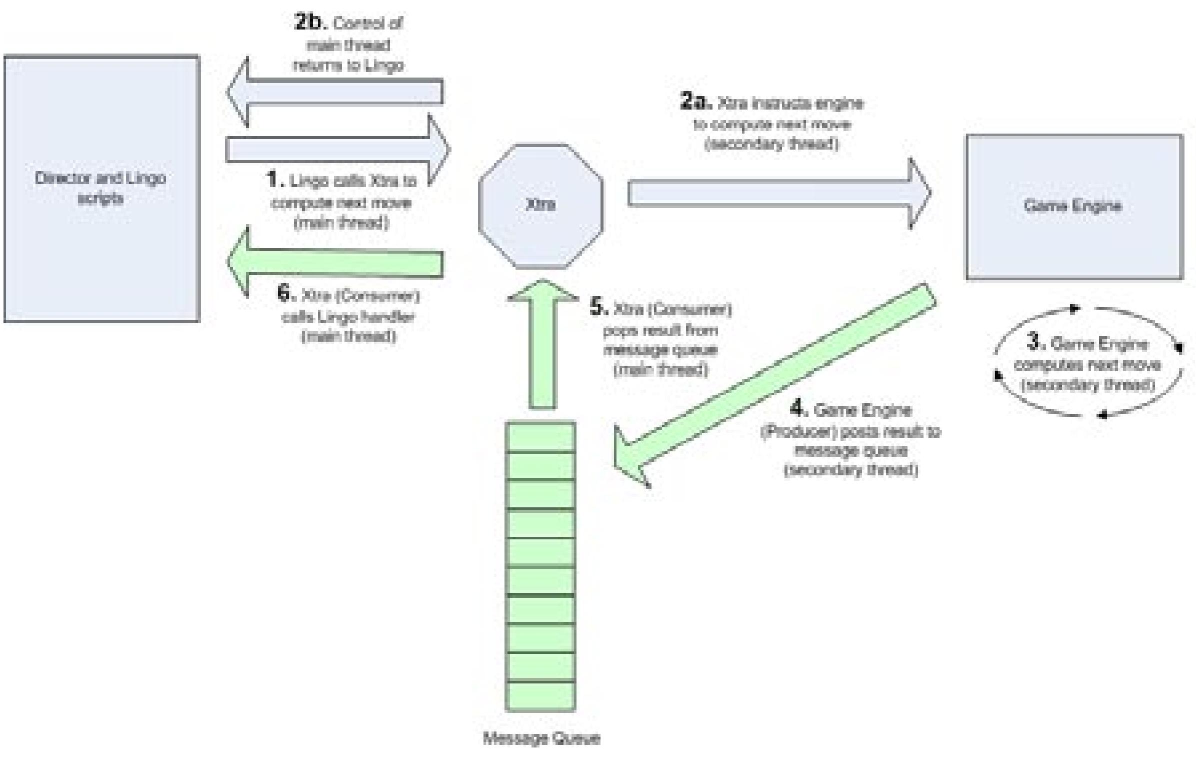
Retrieving a message from the queue:

```
// Make sure that nobody else is
accessing the queue
// This call will block if the queue
is already being accessed elsewhere
EnterCriticalSection(&mCriticalSecti
on);

// Pop the event
CEvent* event = 0;

if (!mEventQueue.empty()) // Check if
the queue contains at least one item
{
 event = mEventQueue.front();
 mEventQueue.pop();
}

// Signal that we no longer require
exclusive access
LeaveCriticalSection(&mCriticalSecti
on);

// Release the Semaphore acquired dur-
ing the push
// This call will eventually unblock a
waiting push
ReleaseSemaphore(mSemaphore, 1, 0);
```

The Producer thread part is very simple. It will only call the message adding method shown above when it needs to communicate with the Consumer.

The Consumer thread will poll the queue manager on idle time and call Lingo every time it finds a message to

figure 3



Director and Lingo scripts
2b. Control of main thread returns to Lingo
1. Lingo calls Xtra to compute next move (main thread)
6. Xtra (Consumer) calls Lingo handler (main thread)
Xtra
2a. Xtra instructs engine to compute next move (secondary thread)
Game Engine
5. Xtra (Consumer) pops result from message queue (main thread)
3. Game Engine computes next move (secondary thread)
4. Game Engine (Producer) posts result to message queue (secondary thread)
Message Queue

process. It would do this with MOA's notification mechanism, more precisely through the IMoaNotificationClient interface. Briefly stated, your Xtra would register to this interface so it can be notified of events of type NID_DrNIdle (which are just plain idle events).

```
PIMoaNotificationClient pNotification-
Client;
PIMoaNotification pMmNotification;
// Acquire MOA notification interface
pObj->pCallback->QueryInterface(&IID_
IMoaNotification, (PPMoaVoid) &This-
>pMmNotification);
if (pObj->pMmNotification != 0)
{
 // Instantiate a notification client
object
 pObj->pCallback->MoaCreateInstance(&
CLSID(CScript), &IID(IMoaNotification
Client), (PPMoaVoid)&pObj->pNotifica-
tionClient);
 if (pObj->pNotificationClient)
 {
 // Register the notification client
object to receive IDLE events
 pObj->pMmNotification->RegisterNotif
icationClient(pObj->pNotificationCli-
ent, &NID_CustomNotificationID , 0,
this);
 }
}
The Notify function below is the core
of the code
STDMETHODIMP CScript_
IMoaNotificationClient::
Notify(ConstPMoaNotifyID nid, PMoaVoid
pNData, PMoaVoid pRefCon)
{
/* variable declarations */
 MoaError err = kMoaErr_NoErr;

 // We ask the notification inter-
face to pass a pointer to our CScript
interface along with the notification
 CScript_IMoaMmXScript* pThis =
(CScript_IMoaMmXScript*)pRefCon;

// Pop the event from the queue man-
ager
CEvent* event = (CEvent*)pThis->pObj-
>pEventManager->PopEvent();

 if (0 != event)
 {
 // Get the value of the counter
pushed by the Producer
MoaLong tick;
 event->GetEvent(tick);

 // Call Lingo
MoaMmSymbol LingoHandlerSym;
MoaMmValue TickValue;

// Set the name of the Lingo handler
to be called as a Symbol
pThis ->pObj->pMmUtils->String
ToSymbol("ThreadSampleEvt", &
LingoHandlerSym);

// Convert the counter vaue to a
MoaValue
pThis ->pObj->pMmUtils-
>IntegerToValue(tick, & TickValue);

// Call the Lingo Handler
pThis ->pObj->pDrMovie->CallHandler(Li
ngoHandlerSym, 1, & TickValue, 0);

// Release the MoaValue
pThis ->pObj->pMmUtils->ValueRelease(&
TickValue);

 // Free the memory used by the event
 delete event;
 }

 return(err);
}
```

If you would like to see the complete script, it can be downloaded from: is http://www.INM.com/services/xtras/sample/.

Conclusion

If you consider the user to be an important stakeholder in your project, then you may be very sensitive to offering them with good-looking and superior user interfaces. However, most of the time, superior user interface is perceived by multimedia developers as being attractive and visually consistent interfaces. But what about functionality?

User interactions with your program must be as appealing and consistent as the user interface, and multithreading plays a key role in letting you achieve this in the context of projects that require time-consuming computation or network communications. This is one of the most commonly requested feature I see now from clients for whom we are developing custom Xtras, or derived versions of one of our commercial Xtras.

Among the different variations of multithreaded code implementation, from our experience, the Producer/Consumer model is by far the most robust and efficient one. It is also very flexible in the sense that it allows you to write properly architected Lingo code and it can itself be properly architected which makes it easier for you and your colleagues to debug, to update and to reuse the code in future projects. It can also be easily extended to accommodate projects that require multiple Producers and Consumers. We have used this solution numerous times and it has consistently proven to be the most stable and efficient option and is something I would definitely recommend to other Xtra developers.

The benefits of the Producer/Consumer model is threefold. Not only does it delivers benefits to the end users of your project, but to the developers of the project and as well, and, not to be discounted, the sponsors of your project who are the people who pay for its development and expect robust software delivered on time at the least possible cost.

In addition to having hopefully successfully brought you to the above conclusion, I hope I was persuasive in demonstrating that multimedia is more than ever multidisciplinary. A few years ago, combining various media together in a simple application with simple navigation was considered to be an immersive user experience.

Indeed it was, compared to the alternatives available back then. But this base line, as well as users' expectations, is constantly increasing. To remain competitive, one has to enhance every aspect of their multimedia projects, including the software portion. This requires the implementation of advanced though proven software architecture concepts such as multithreading, component architecture, object-oriented design and C++ templates.

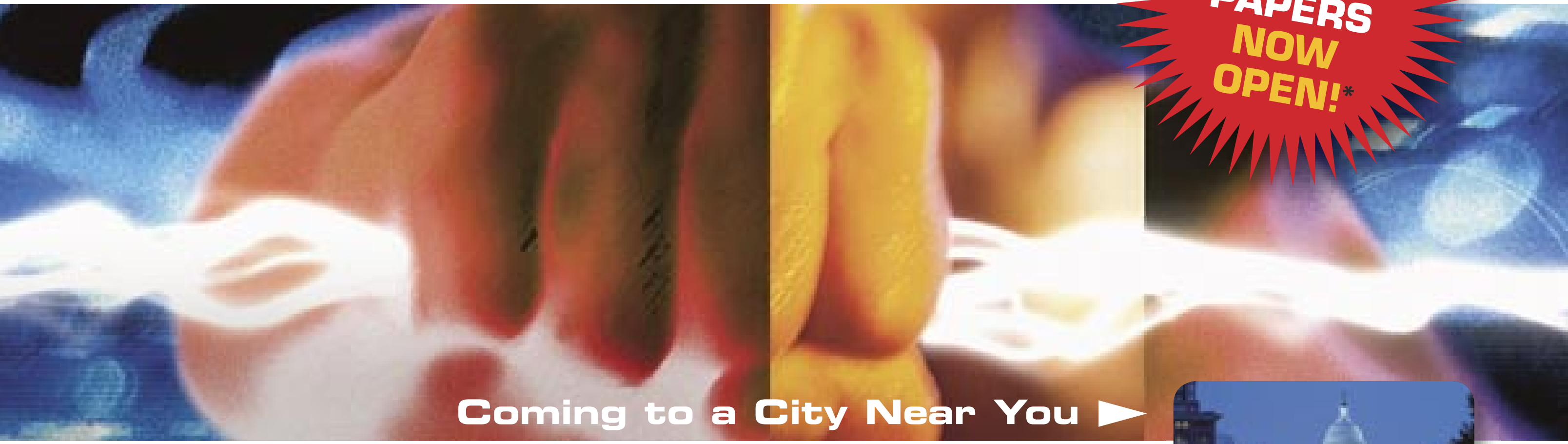
Today's standards for delivering a superior multimedia projects require, in addition to superior artistic and content development skills, superior software engineering skills. A requirement, we are likely to see evolve further still in the years to come.

[Engage and Explore] the technologies, solutions and applications that are driving today's **Web services** initiatives and strategies...

www.sys-con.com/edge2005

web services EDGE conference & expo
FALL SERIES

CALL FOR PAPERS NOW OPEN!*



Coming to a City Near You ►

Web Services Edge

Fall Conference Series

3 Dynamic Conference Programs

Targeting Major Industry Markets

20+ seminars within 5 tracks will address the hottest topics & issues:

- ► Web Services: The Benefits and Challenges
- ► Web Services Security
- ► SOA (Service-Oriented Architecture) and ESB (Enterprise Service Bus) Strategies
- ► Interoperability, Incremental Integration, & Open Source
- ► The Management Process in Developing a Web Services Strategy

Program Features:

- ► Keynotes
- ► Tutorials
- ► Panel Discussions

Why Attend:

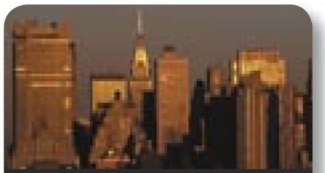
- ► Improve the return on your technology investment
- ► Develop & sharpen your strategy and identify key action steps
- ► Find new ways to reach and impress customers with Web services
- ► Maximize the power of your enterprise
- ► Protect your business from security threats
- ► Assess Web services as a viable option

Attention Exhibitors:

► An Exhibit-Forum will display leading Web services products, services, and solutions

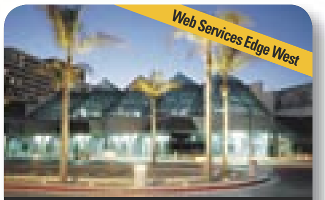
1

The Westin Washington, D.C.
Washington, D.C.
September 7-8, 2005



2

Helmsley Hotel
New York, NY
September 19-20, 2005



3

The Westin Santa Clara Convention Center
Santa Clara, CA
October 24-25, 2005

Register Today! www.SYS-CON.com/Edge2005

Sponsored by

WebServices Journal · XML Journal · .NET Developer's Journal · eclipse developer's journal · WebSphere Journal · Information Storage+Security Journal · wldj · JDJ · LinuxWorld Magazine · MX developer's journal · asp.netPRO · SD Times · CoDe · Software Test & Performance

*Call for Papers email: grisha@sys-con.com

For Exhibit and Sponsorship Information ► Call 201 802-3066

Produced by SYS-CON EVENTS

Delivering Captivate Content on the Web

With Macromedia Captivate, you can create interactive tutorials with recorded narration, built-in testing, with an easy, painless, and quick process. But now that authors can so easily create tutorials, they have time to think about the best way to deliver their content to end users, and to think about what works well with the Captivate workflow.

by jesse randall warden

ith Macromedia Captivate, you can create interactive tutorials with recorded narration, built-in testing, with an easy, painless, and quick process. But now that authors can so easily create tutorials, they have time to think about the best way to deliver their content to end users, and to think about what works well with the Captivate workflow.

When I started using Macromedia Captivate early last year, I quickly learned that Captivate authors did not have a clear deployment path for Captivate content they wanted to make available on the web. This problem does not stem from Captivate – the tool has a plethora of publishing options. Captivate can publish to SWF file format, optionally wrapped in HTML, to Macromedia Breeze, a stand-alone executable with an auto-run option for CD-ROM delivery, e-mail, Microsoft Word, and you can even send your content to a server through FTP functionality from within Captivate. While heavily testing Captivate for my own use and collaborating with others in the Captivate community, I quickly found many Captivate authors that are experienced, organized, and extremely talented in delivering documentation, tutorials, and other various e-learning content.

To support effective and easy-to-use delivery of Captivate content on the web, I developed a utility, called CaptivatePlayer. In this article, you will learn how to deploy Captivate content with CaptivatePlayer.

Introduction

Captivate content is most effective when you create many small, concise Captivate demonstrations or simulations. This is because Captivate has to manage all of the media it creates: the screen-capture images, the audio narration, the custom text, cursor movements, and so forth. That is a lot of activity and assets to track. Even with a fast computer, it is best if you create many small Captivate demonstrations and simulations because this affects your end user. If the Captivate content is smaller, it is easier for an end user to follow and understand. Moreover, the actual SWF file size is smaller and results in quicker downloads, which ensures a better playback experience for the end user.

Following this best practice creates a challenge, however: You suddenly have a bunch of SWF files, and you might not be sure what to do with them. If you are like most authors, you probably want your users to access your Captivate content in a web browser on the Internet, your local intranet, or as a companion executable. Since Captivate creates an HTML file or an executable for each Captivate demonstration or simulation, does this mean you have to create a website just to manage all of your content? Do your users need to keep track the content (which, by the way, isn't the best experience for them)? What about the executables? Will the user see each piece of Captivate content as a separate "program"?

Using CaptivatePlayer

I recognized the need for Captivate authors to have a simple way to play multiple Captivate SWF files and to make it easy for them to set up since there are a wide range of skill sets and backgrounds in the Captivate community. In addition to making CaptivatePlayer easy to implement for authors, it needed to be easy to access and use for the audience. Creating a new HTML page that loaded the Captivate SWF files into the HTML files, per project was unappealing; equally unappealing was having multiple EXE files grouped together in a folder. Linking to SWF files directly in a web browser does not take into account various player version checks, control over how your SWF displays in the browser, or provide your user with any sense of continuity and navigation between each SWF file. You might be able to use the Captivate MenuBuilder for creating the necessary web content, but doing this for each project could be time consuming and redundant work. Captivate is fun and easy to use for end users; I wanted an easy way to quickly deploy created content to the web.

Figure 1 shows an example of a Flash tutorial I wrote that displays in the CaptivatePlayer (highlighted in green), through the Firefox browser. As you can see, CaptivatePlayer frames the Captivate content unobtrusively, and takes up very little room.

How CaptivatePlayer Works

With CaptivatePlayer, your users can view many Captivate SWF files through one consistent interface. CaptivatePlayer provides navigation for users to access all Captivate SWF files. Users access SWF files through a pop-up menu that contains all of the Captivate SWF file names.

By using a menu, you maximize valuable screen real estate. CaptivatePlayer also has controls for scaling Captivate content in case your users have smaller viewing areas. It also includes a mute option for audio.

CaptivatePlayer does not include playback controls because Captivate already provides these for you when you create Captivate content. CaptivatePlayer displays in 100% of the browser window, again maximizing screen real-estate. CaptivatePlayer is one SWF file that dynamically loads the SWF or EXE files published from Captivate.

You can deploy Captivate content easily in CaptivatePlayer with the following steps:

1. Edit the captivate_playlist.xml file by adding your Captivate SWF file names to it.
2. Edit the playback options in the captivate_playlist.xml or in the index.html file.
3. Upload your Captivate SWF files, the captivate_playlist.xml, the index.html, and CaptivatePlayer.swf file to your web server or to the locally accessible folder that you're using. If you're using an executable, you just need your Captivate SWF files, the captivate_playlist.xml, and the CaptivatePlayer.exe. All files must be in the same folder to correctly operate.

If you change or update your Captivate projects, just upload the new SWF files. If you've added or deleted SWF files from your presentation, modify the new list of SWF files in the captivate_playlist.xml file and upload the updated XML file.

Configuration Options

There are two levels of configuration that you can specify for CaptivatePlayer. The web and EXE versions of CaptivatePlayer read captivate_playlist.xml (a required file), to know which Captivate content to play. This is due to the way security is implemented in Macromedia Flash Player. A SWF file cannot read the contents of a local folder; you must manually specify it in the captivate_playlist.xml file. To do so, type the names of your Captivate SWF files into the XML file.

The XML file has a few configuration options that you can edit. These options control the way CaptivatePlayer plays your Captivate content. The configuration options specify:

- auto play
- the starting sound volume
- whether or not to scale the size of the Captivate content

Advanced authors can specify these options in the index.html file that calls the CaptivatePlayer.swf file, but the XML file takes precedence.

The following example is a typical web deployment.

How to Use the Play List XML File

Before you get nervous, remember that XML files are just text files. You can edit them in Notepad, WordPad, Microsoft Word, or any other text editor. I prefer Macromedia Dreamweaver because it color codes my XML nodes, aiding readability. The following is the XML file in Dreamweaver:

```
<?xml version=”1.0” encoding=”iso-
8859-1”? >

  <captivate_playlist
     autoplay=”true”
   “volume=”50”
    content=”true”>
<swf src=”demonstration1.swf”
name=”Introduction”>
      <swf src=”demonstration2.swf”
name=”How To Do”>
      <swf src=”demonstration3.swf”
name=”Conclusion”>

  </captivate_playlist>

</xml>
```

When CaptivatePlayer uses the XML file excerpted in Figure 8, CaptivatePlayer performs the following actions:

- Automatically starts playing the first SWF file
- Sets the volume to 50%
- Scales the Captivate SWF file to fit to the size the CaptivatePlayer
- Puts demonstration1.swf, demonstration2.swf, and demonstration3.swf into the menu, but uses the names Introduction, How To Do, and Conclusion

Listed below are the options for each of the attributes.

Attributes for the captivate_playlist tag	Value
autoplay	true or false. Defaults to “true.”
volume	0-100
scalecontent	true or false. Defaults to “true.”

In the code sample above, the SWF source tag has a specified name attribute. You could have used the SWF file name, but “Introduction” is more intuitive to the user than demonstration1.swf. If you want to add more Captivate SWF files, you simply add another SWF tag in the XML file, just like the examples provided. You can add as many SWF tags as you like, but you must have at least one SWF tag always present.

How to Use the index.html File

I included a default index.html file for authors with either a Flash or ColdFusion background. First, for those in a hurry,

figure 1

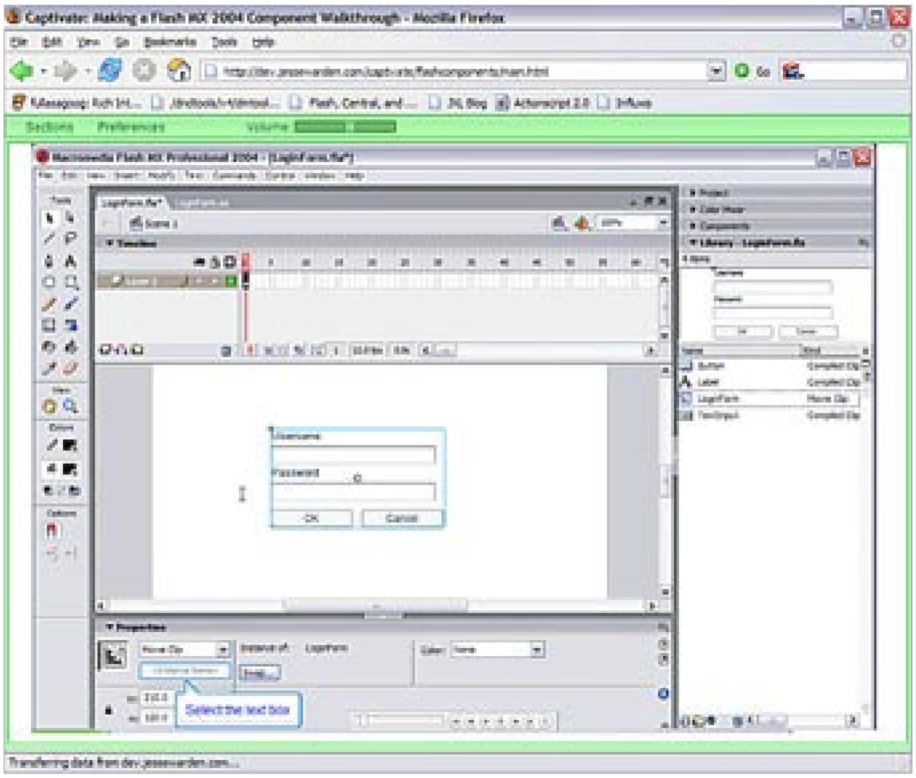

Captivate: Making a Flash MX 2004 Component Walkthrough - Mozilla Firefox
Sections
Preferences
Volume
Macromedia Flash MX Professional 2004 - [LoginForm.fla*]
Username
Password
OK
Cancel
Select the text box
Transferring data from dev.jesseswarden.com...

figure 2

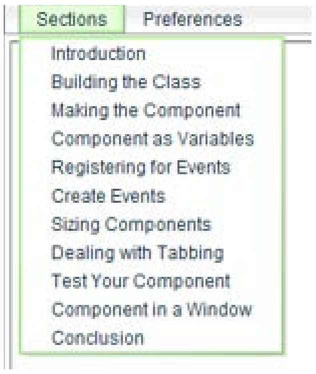


figure 3

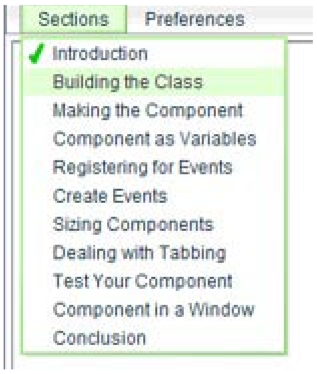
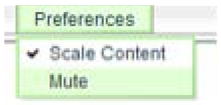


figure 4



Reach Over 100,000

Enterprise Development Managers & Decision Makers with...

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

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

Get Listed as a Top 20* Solutions Provider

For Advertising Details Call 201 802-3021 Today!

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

SYS-CON MEDIA

The World's Leading i-Technology Publisher figure 5

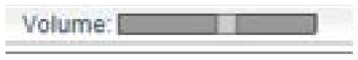


figure 6

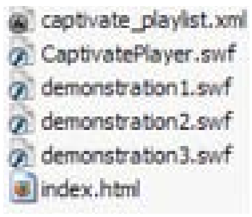

captivate_playlist.xml
CaptivatePlayer.swf
demonstration1.swf
demonstration2.swf
demonstration3.swf
index.html

figure 7

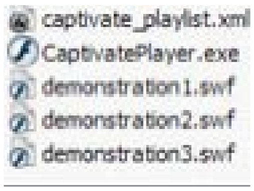

captivate_playlist.xml
CaptivatePlayer.exe
demonstration1.swf
demonstration2.swf
demonstration3.swf

figure 8

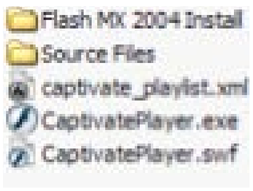

Flash MX 2004 Install
Source Files
captivate_playlist.xml
CaptivatePlayer.exe
CaptivatePlayer.swf

figure 9

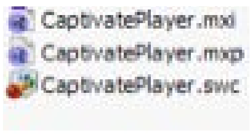
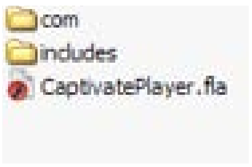

CaptivatePlayer.mxi
CaptivatePlayer.mxp
CaptivatePlayer.swc

figure 10


com
includes
CaptivatePlayer.fla

the HTML is already written for you; the CaptivatePlayer is embedded into the HTML with the necessary code to make it play in the full screen in the browser. Second, for those incorporating CaptivatePlayer into their own website design, you can copy and paste the code for embedding CaptivatePlayer.

For more details, refer to readme_index.txt, included in the ZIP file you downloaded in the Requirements section. It explains in detail how to customize and use the index.html file.

The CaptivatePlayer Source Files

There are many ways you can use CaptivatePlayer (as an embedded SWF file in HTML or as an EXE file) and it can be confusing understanding all the files included in the ZIP file that accompanies this article.

After unzipping the archive, the base level directory has two folders and three files (Figure 10).

For Flash Developers who need to use the CaptivatePlayer in an existing Rich Internet Application, the necessary files are in the "Flash MX 2004 Install" folder. Included is the MXP, which installs the CaptivatePlayer as a component in the Components Panel. I've included the MXI and SWC for Flash Developers who need those instead.

For Flash Developers who have more specific needs to customize CaptivatePlayer, the "Source Files" folder contains the FLA file, created in Flash MX 2004 and source AS (ActionScript) files. This folder contains everything a Flash Developer needs to customize the design of the CaptivatePlayer, tweak the way it works, add functionality, and/or compile a customized version. A default setup file, used to initialize the CaptivatePlayer, is included in the includes folder.

Back to Top

Where to Go from Here

Macromedia Captivate is the easiest and most flexible way to create interactive demonstrations and software simulations. It is fun to use, does the hard work for you, and lets you spend more time focusing on polishing up your tutorial, simulation, demonstration, or test to make it more effective. Because I have prior experience creating these types of presentations manually in Flash, I know that Captivate significantly reduces the amount of time and work required to create these demonstrations and simulations. I hope that CaptivatePlayer complements your workflow by providing a final, easy step in deploying your finished work for your users online, on an intranet, or even running locally, off of another user's machine.

In talking to Captivate authors from all over the USA, from Canada to Germany, by phone and e-mail, I can definitely say there is room to grow and improve the way the CaptivatePlayer deploys Captivate movies. One author has requested that I add a global control bar instead of having Captivate generate one for each demonstration or simulation. This would help end users quickly access other Captivate content, not just locations inside one demonstration or simulation. Other authors, who have lot of Captivate content, called modules, also have sub-modules. They prefer the current menu to contain nested menus, like most context menus do today (File > New or Right click > Properties for example). This is especially ideal for users, because a volume of 50-100 Captivate demonstrations or simulations does not fit well into a one-level menu.

Creating these demonstrations or simulations is only part of the process. I created CaptivatePlayer to fulfill a need to deliver Captivate content quickly to users. There is still room for improvement as I have mentioned, so if the CaptivatePlayer does not satisfy your deployment needs, please do send me feedback.

Notes from the Author

1. You can see CaptivatePlayer in action. The following is a tutorial I created for creating a Flash MX 2004 component: dev.jessewarden.com/captivate/flash-components/main.html.
2. To request changes in the next version of Captivate, go to: www.macromedia.com/go/wish/.

Jesse Warden is a professional multimedia developer working at Roundbox Media. He specializes in Flex and Flash Development. He has spoken in Sydney Australia at MXDU 2003 and MXDU 2005, multiple Atlanta Macromedia User Groups, Georgia Tech, and other venues about various Macromedia products and technologies. Jesse runs a blog at jessewarden.com where he contributes coding techniques, plug-ins, and sample projects to the community. jesse@jessewarden.com

FREE RESOURCE CD INCLUDED!
information STORAGE+SECURITY journal
www.ISSJournal.com
In this issue:
IP Storage
SAN NAS
Disaster Recovery
Enterprise Security
Infosecurity
Microsoft RSA Security
SecureID for Windows
SAN NAS
Emerging technology trends and market maneuvers

Using Runtime Shared Libraries

Improving download performance

by roger gonzalez

With Macromedia Flex 1.5 you can build runtime shared libraries (RSLs) that can be individually loaded, cached, and used by multiple applications. This article demonstrates how easily you can integrate RSLs into your Flex applications. It also addresses the performance tradeoffs that you must consider when building dynamically linked applications.

The Basics

The best way for you to learn about RSLs is to build several copies of the same application with and without various permutations of RSL settings. Follow these steps:

1. Create Info.mxml, a small MXML component that displays its initialization time:

```
<?xml version="1.0" encoding="iso-
8859-1"?>
<mx:Button xmlns:mx="http://www.macro-
media.com/2003/mxml"
          backgroundColor="#ffffff"
          width="250" height="150"
          initialize="go()">
  <mx:Script>
    var name:String = "?";
    function go()
    {
        var start:Number = 0;
        if (_global.startTime !=
undefined)
          start = _global.start-
Time;
        var dt:Number = (getTimer() -
start) / 1000;
        label = name + " initialized
in " + dt + " s." ;
    }
   </mx:Script>
  </mx:Button>
```

Tutorials and sample files can be download at http://download.macromedia.com/pub/developer/rsl_examples.zip

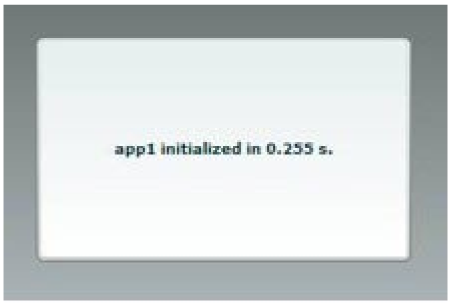


figure 1

app1 initialized in 0.255 s.

2. Create app1.mxml, a trivial little application that you use as the basis for the experiments:

```
<?xml version="1.0" encoding="iso-
8859-1"?>
<mx:Application xmlns:mx="http://www.
macromedia.com/2003/mxml"
                xmlns="*"
                width="300"
height="200">
  <Info name="{className}" />
</mx:Application>
```

3. Load app1.mxml in your browser. You will see a button similar to Figure 1.

The size of the generated SWF file is fairly large, because simply using the Application and Button classes causes much of the Flex application model to be linked into your application. If you have the <keep-generated-swfs> option enabled in your Flex configuration file, Flex writes a copy of the SWF file to the application directory as app1.swf (not to be confused with the URL used to download the SWF file, which is actually app1.mxml.swf):

```
% ls -l app1.swf
-rw-------    1 rg       129899 Oct
18 07:14 app1.swf
```

Each Flex application starts from a similar baseline size, because each contains most of the same components. Clearly, if you have multiple Flex applications on your server, it is wasteful (of users time and your bandwidth) for users to have to download so much redundant information!

Wouldn't it be nice to factor shared components and assets in your application into libraries that could be loaded at runtime? You can!

The following steps show you how to convert your baseline application to use RSLs:

1. Specify what goes into the library. Create a file named shared.sws with the following content:

```
<library>
  <component name="Application"
             uri="http://www.macrome-
dia.com/2003/mxml" />
  <component name="Info" uri="*" />
</library>
```

2. Copy app1.mxml to app2.mxml. Edit app2.mxml to look like this:

```
<?xml version="1.0" encoding="iso-
8859-1"?>
<mx:Application xmlns:mx="http://www.
macromedia.com/2003/mxml"
                xmlns="*"
                width="300"
height="200"
                rsl="shared.sws">
  <Info name="{className}" />
    </mx:Application>
```

3. Load app2.mxml in your browser, and if you haven't made any typos, it should look virtually identical to app1 (Figure 1).

Congratulations, you've now converted a statically linked Flex application (all components and assets defined internally) to a dynamically linked Flex application (some components and assets loaded at runtime). Observe the file sizes:

```
-rw-------    1 rg       129899 Oct
18 07:14 app1.swf
-rw-------    1 rg         9127 Oct
18 07:23 app2.swf
```

You've decreased the SWF file size by 117K – quite an improvement! I'll explain where the bits went in the next section. On the surface, it might appear from your SWS file that you were only going to import the Application and Button components. However, you are actually also importing all their dependencies. In general, you will see an immediate improvement in download size by building a SWS file that just references Application, but you get the best improvement by carefully setting up your SWS specification to include all referenced shared components.

Currently, there is no straightforward way to know the optimal set of components to include in your SWS file. However, you can get some hints by perusing the compile report generated when you build your application (and have the appropriate setting enabled in your Flex server configuration file). Any symbol definition marked "external" is imported from a RSL. All other symbols are linked into the application itself. Don't worry too much if a few stray classes get linked in; if you just list the MXML elements that you use, you are guaranteed to get a highly effective RSL.

Download Performance

Although Flash Player expects an RSL to be a SWF file, the file generated from a SWS library specification is actually a SWC component library. A SWC file is a compressed archive (in ZIP format) containing an implementation SWF file and additional metadata for use by the compiler. (This enables you to quickly switch between dynamic linking with the rsl property and static linking with the lib property!)

The library you built from shared.sws is created in the Flex server's generated/libs directory hierarchy. The Flex server knows how to extract the implementation SWF file (often named Library.swf) from the SWC file by the URL; in this case the URL is shared.swc.swf. You can use any ZIP tool to peek at the size of the SWF inside the library:

```
% unzip -l $FLEX_APP/WEB-INF/Flex/generated/libs/*/shared.swc | grep '.swf'
   140628  10-18-04 07:23   Library.swf
```

Notice that the aggregate size of the dynamically linked application app2.swf plus the RSL shared.swc.swf is larger than the size of the statically linked app1.swf file. This is due to the overhead that is part of a SWF library as well as some lost opportunities for code merging in the optimization phase of the Flex compiler. This overhead should make it obvious that RSLs do not help if you only have a single Flex application. The real download performance benefit occurs when you have multiple applications sharing the same RSL.

Imagine users downloading two statically-linked applications similar to app1.mxml compared to downloading two dynamically-linked applications similar to app2.mxml:

```
2x (app1.swf @ 129899 ) = 259798
2x (app2.swf @ 9127) + (shared.swc.swf @ 140628 ) = 158882
```

The aggregate download size improvement using dynamic linking is clearly a big win here (That's about 18 seconds for a user on a 56K modem, if those still exist; but perhaps more importantly it reflects nontrivial cash savings in network usage if you have lots of users.). The download size improvement only improves more so as you leverage the same RSL! Thus, although the download size of a single dynamically-linked application is slightly worse than the download size of a comparable statically-linked application, the amortized download performance across multiple applications is much better.

Runtime Performance

Unfortunately, the runtime cost of using RSLs is not that simple. It has several dimensions.

To download the application and its RSL(s), Macromedia Flash Player needs to perform multiple HTTP transactions where previously one was sufficient. This adds latency (potentially one second per additional TCP connection if HTTP KeepAlive isn't supported) and an additional failure mode (a network glitch after application load but before RSL load).

Since not all applications are identical, it is quite likely that an RSL built with the union of all components needed by several applications will (when considered on a per-application basis) always contain more components and assets than the "pruned-down" list possible with a statically-linked application. These extra unused definitions have a small but measurable overhead.

To experience these issues first-hand, follow these steps:

1. Create allmx.sws with the entire Flex application model included:

```
<library>
  <namespace uri="http://www.macromedia.com/2003/mxml" all="true" />
   </library>
```

2. Copy app2.mxml to app3.mxml, and change it to use your newly created SWS file:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml"
                xmlns="*"
                width="300"
height="200"
                rsl="allmx.sws">
  <Info name="{className}" />
   </mx:Application>
Compiling this version results in a very large RSL:
% unzip -l $FLEX_APP/WEB-INF/Flex/generated/libs/*/allmx.swc | grep '.swf'
   457111  10-17-04 20:14   Library.swf
```

Observe that the aggregate download of this RSL plus two app3 SWF files would be much larger than even the statically-linked app1 SWF file! Unless you can leverage this RSL more effectively, overall download performance will be worse than in the case of a statically linked app1 SWF file.

Also, unless you are using a much faster computer than mine, compiling, downloading, and even initializing this version took quite a bit of time. Before you can address the amount of time taken, you need to be able to differentiate the time spent on compilation download time and initialization time. Compilation time is a one-time amortized cost, and you can eliminate it by using precompilation; download time is another amortized cost, described earlier. Initialization time would appear to be a fixed cost, but you will discover later on that it can also be improved.

Without changing any files, clear your browser cache, and observe how long it takes to load app2.mxml compared to app3.mxml. Since neither application needed recompilation, you should see only the time necessary for download and initialization. (On a fast computer with a local server, you may not see much of a difference; if you have a network debugger tool that enables you to simulate a slow connection, now would be a great time to use it! If not, you might try connecting remotely or embedding large images into the application and using RSLs to slow things down.)

Because RSLs are normally loaded by Flash Player in a manner that blocks the application and provides no mechanism for error recovery, Flex RSLs are "pre-fetched" by the application preloader to monitor the download status and report errors, and the "real" load of the RSL is a quick fetch out of the cache. However, this extra pre-fetch adds some overhead. You can disable this feature for any RSL by adding preload="false" to your RSL's <library> element. (Disabling the application preloader will disable all RSL pre-fetching.)

Hierarchical RSL Loading

While you can build a single custom RSL that contains a mix of your own components and the Flex application model, you might also find it valuable to partition-related groups of components and assets into a hierarchy of multiple RSLs. For example, create a new file named applib.sws with the following library specification:

```
<library rsl="allmx.sws">
  <component uri="*" name="Info" />
</library>
Then, copy app3.mxml to    app4.mxml,
with the following change:
<?xml version="1.0" encoding="iso-
8859-1"?>
<mx:Application xmlns:mx="http://www.
macromedia.com/2003/mxml"
                xmlns="*"
                width="300"
height="200"
                rsl="applib.sws">
  <Info name="{className}" />
</mx:Application>
```

This is probably the slowest version of all the applications you have built so far in this tutorial, but it allows unrelated applications to use the Flex application model in its own RSL without the overhead of other unnecessary components. Again, you will need multiple large applications that use most of the Flex components in order to effectively leverage this configuration.

Test Harness

So far, you have created several dynamically linked applications. However, so far, you've only relied on how long it "feels" for an application to load, and the amount of time it takes for the application to initialize—a crude measurement that depends on the ActionScript getTimer() call starting at zero when Flash Player initializes. It also suffers from the problem that Flash Player may be started up asynchronously with the actual network download of the SWF file, and isn't necessarily seeing the true startup time from the first bit of data. The results of this crude measurement are probably a bit optimistic.

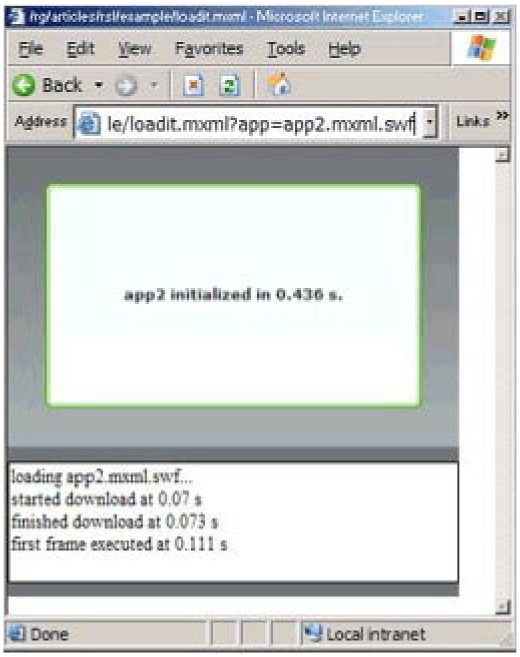
The best way around this problem is to start Flash Player ahead of time with a bootstrap loader application, and have the bootstrap loader measure the actual load time of your applications. Describing how this works is beyond the scope of this article, but a simple (and fairly self-explanatory) version called loadit.mxml is included here: http://www.macromedia.com/devnet/flex/articles/rsl.html. You use this application by passing it the name of an SWF file as the value of the app query parameter in the URL (see Figure 2).

The URL I use to load the app is http://localhost:8100/flex/rslarticle/loadit.mxml?app=app1.mxml.swf.

Note the use of the .mxml.swf extension to get the loadit app to load your application's SWF data rather than the application's HTML shell, which would just confuse Flash Player.

Try loading app1.mxml through app4.mxml, clear the cache, and then load each application again. Observe the increase in startup time with each increasingly complex use of dynamic linking.

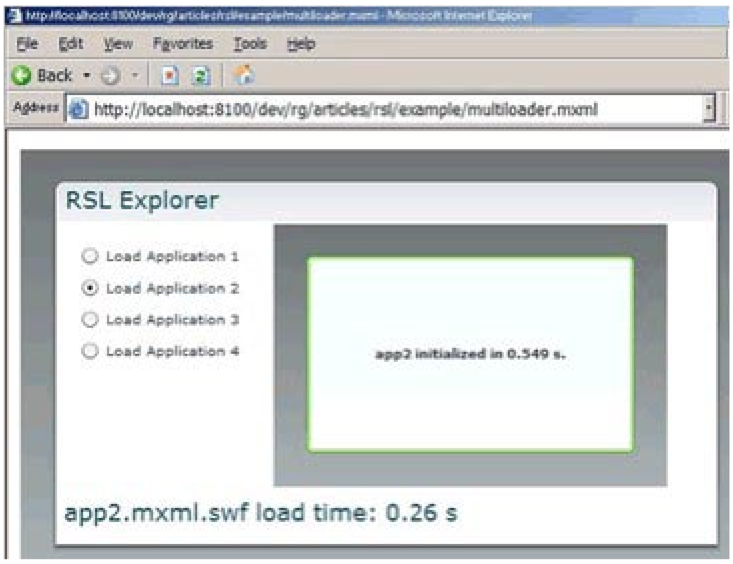
figure 2


rg/articles/rsl/example/loadit.mxml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address le/loadit.mxml?app=app2.mxml.swf
Links
Done
Local intranet

app2 initialized in 0.436 s.

```
loading app2.mxml.swf...
started download at 0.07 s
finished download at 0.073 s
first frame executed at 0.111 s
```

figure 3


http://localhost:8100/dev/rg/articles/rsl/example/multiloader.mxml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://localhost:8100/dev/rg/articles/rsl/example/multiloader.mxml
RSL Explorer
Load Application 1
Load Application 2
Load Application 3
Load Application 4

app2 initialized in 0.549 s.

app2.mxml.swf load time: 0.26 s

Applications, Loaders, and "Mini-Apps"

If you look at loadit.mxml, you will notice that it is mostly pure ActionScript and does not use of any of the Flex application model. This is necessary because as it turns out, using the Flex components in the loader application has a direct effect on the initialization time.

To demonstrate how significant, load the multiloader.mxml application included in the samples.zip file mentioned above. This application is similar to the loadit version, except that it is built using the Flex application model; it does its work using the Loader component instead of low-level ActionScript, and app1 through app4 can be loaded with a click.

As you load the different applications, observe that the times are uniformly lower than the times when using loadit.mxml, even if you clear the browser cache while multiloader.mxml is running.

This is because the ActionScript class prototype setup only occurs once for a given class, even if the class is loaded again later. Although the timing data displayed by loadit.mxml is a more accurate measurement of loading and initializing an application from scratch, you can exploit the initialization time improvements demonstrated by multiloader.mxml as a technique for incrementally loading very large applications.

By partitioning your application into an outer shell and a number of individually loaded modules, and optimizing the download by making all the modules and the outer shell share common RSLs, you get the best of both worlds. While this takes a bit more work for the developer, it pays for itself with significantly faster interactivity compared to a monolithic application.

Compilation Performance

Although compilation performance is nowhere near as important as download and runtime performance, it's still very nice to have a quick rebuild cycle.

If you use RSLs with an application that is frequently changed on a live server, you will see significant compilation and download improvements by moving components and assets that do not change into an RSL – only the application itself will be recompiled. The RSL will be up-to-date and available for fast relinking and will not need to be downloaded again by users if it has already been cached by their browsers.

You can also invert this process if you have an RSL undergoing frequent changes. Just set the rebuildClients="false" attribute in your SWS file, and the RSL can be rebuilt without triggering the application to recompile. This is dangerous if you change the component source code, however, because you must make certain that you don't change the interface to any of the classes in a way that will break the application. However, for certain RSLs that contain only assets (for example, imagine an RSL that is rebuilt every morning from an autogenerated SVG diagram), it will significantly improve performance.

If you set up your RSL correctly, you can even use it as a substitute way of dynamically loading image types that would normally need to be embedded because their format isn't natively supported by Flash Player – the RSL compiler is acting as an automatic transcoder!

With sufficiently tricky library configuration (using the rebuildClients="false" and preload="false" attributes), it is possible to get into a situation where a compilation error in a library component cannot be reported anywhere (normal errors require the application to be rebuilt in order for them to be reported in the HTML shell). If it seems like your RSL isn't loading, fall back to the default settings to ensure that you're not missing some important warnings or error output.

In conclusion, runtime shared libraries are easy to use, but may impose some noticeable runtime overhead. This cost is generally acceptable if you can leverage dynamic linking to significantly improve your applications' amortized download performance. By understanding the tradeoffs of different RSL configurations, you can choose the most effective architecture for your applications.

Roger Gonzalez is a Principal Engineer on the Flex compiler team. Prior to Macromedia, he has done everything from working on autonomous underwater robots to running the engineering group at a 3D game development studio. Roger is an avid motorcyclist, and recently relocated to California in order to pursue year-round riding. rgonzalez@macromedia.com

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

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

PRODUCTION
Production Consultant
Jim Morgan, 201 802-3033
jim@sys-con.com
Lead Designer
Louis F. Cuffari, 201 802-3035
louis@sys-con.com
Art Director
Alex Botero, 201 802-3031
alex@sys-con.com
Associate Art Director
Tami Beatty, 201 802-3038
tami@sys-con.com
Assistant Art Directors's
Andrea Boden, 201 802-3034
andrea@sys-con.com
Abraham Addo, 201 802-3037
abraham@sys-con.com
Video Production
Frank Moricco, 201 802-3036
frank@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
Vincent Santaiti, 201 802-3054
vincent@sys-con.com

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

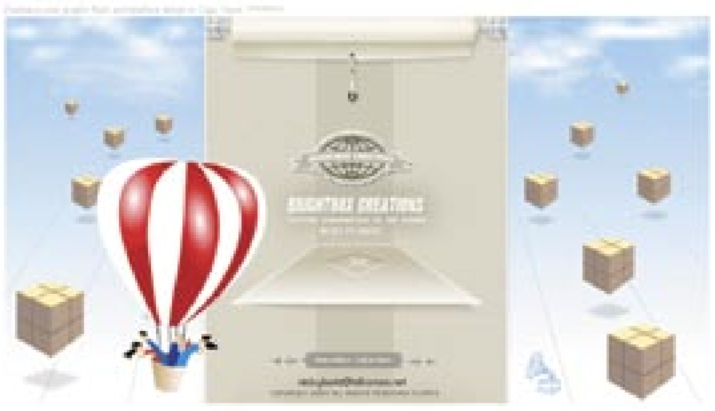
EVENTS
President, SYS-CON Events
Grisha Davida, 201 802-3004
grisha@sys-con.com
National Sales Manager
Jim Hanchrow, 201 802-3066
jimh@sys-con.com

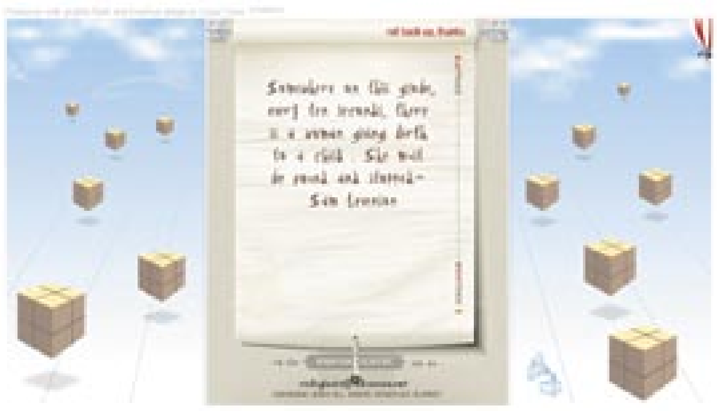
CUSTOMER RELATIONS
Circulation Service Coordinators
Edna Earle Russell, 201 802-3081
edna@sys-con.com
Linda Lipton, 201 802-3012
linda@sys-con.com
JDJ Store Manager
Brundila Staropoli, 201 802-3000
bruni@sys-con.com

SYS-CON MEDIA

Brightbox

brightbox is my 2005 portfolio Web site – it's a Flash movie inspired by my favorite design styles all rolled into one movie. I wanted to create something completely out of the ordinary – something people would remember, and I wanted users to want to "figure out" how to use the site and discover its little surprises along the way. www.brightbox.co.za/





Drown your competition with Intermedia.NET

Looking for a hosting company with the most cutting-edge technology? With **Intermedia.NET** you get much more than today's hottest web & Exchange hosting tools – you get a decade of experience and an unmatched reputation for customer satisfaction.

Thousands of companies across the globe count on us for reliable, secure hosting solutions...and so can you.

In celebration of 10 successful years in the business, we're offering a promotional plan for just $1. Visit our website www.intermedia.net to find out more.

Our premier hosting services include:

- Windows 2003 with ASP.NET
- ColdFusion MX with Security sandboxes
- Linux with MySQL databases
- E-Commerce with Miva Merchant Store Builder
- Beneficial Reseller Programs
- ...and much more!

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

INTERMEDIA.NET

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

macromedia®

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

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

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