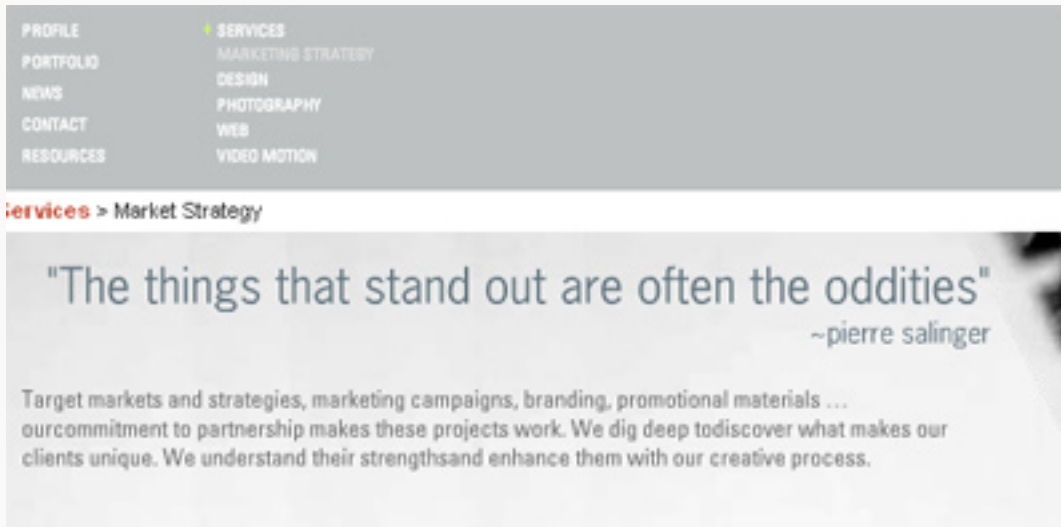


FLASH + CONTENT MANAGEMENT SYSTEMS WITH SEARCH ENGINE OPTIMIZATION



Above is an example of a flash navigation and a flash body content in action. All the text is in the html generated by the CMS, and thus retains all the SEO. This allows for embedding fonts and styles along with all your regular flash media.



About the Author, Steven Stark:

Steven received his first job as a Junior Web Developer in 2000, and within the first year I was managing a small web development team after running the company solo for 6 months. By 2002, he attended the Center of Digital Arts and Technology where he completed the 3 "year" X-Media certification.

After college he worked for another design company that dealt primarily with Car Dealers, producing over 20 high end websites in just

under 6 months. Afterwards, he then moved to Vancouver and worked in Affiliate Management for a large Online Sports company. Soon after, however, he started focusing on his own contracts and soon decided to turn that into a business. Being a jack of all trades allows Steven the unique ability to manage and teach a large staff what he knows and how to do it properly and efficiently.

This experience and position makes it so Steven is able to keep on the cutting edge of new technologies like Flash and Flex Actionscripting, and come up with some new solutions to year old problems.



Appendix

CHAPTER #1 - Introduction

- Method to the Madness
- Action Script and Flash
- XML
- CMS / Joomla!
- Fonts
- Javascript

CHAPTER #2 - Preparation

- Joomla!
- Class' and Prototypes
- Forums

CHAPTER #3 - Customization

- Templates and CSS
- CMS Customization
- Strict XHTML

CHAPTER #4 - Action Script

- Fonts
- Javascript
- I.E. VS F.F.
- Add HTML and Parse Results
- Render HTML
- Filters and Gradients

CHAPTER #5 - Rinse and Repeat

- Troubleshooting and Testing
- Different .swf for each block
- Search Engine Optimization
- Credits

A large, stylized letter 'I' in a light green color, positioned behind the title 'Introduction'. The 'I' is composed of several overlapping rectangular blocks of varying shades of green.

Introduction

Method to the Madness

This process is very long and nerve racking, and should not be tried at home by any sane person. That said, I am slightly crazy and like to push things to the edge, and I came across a problem. The problem was that I wanted to have flash animations behind the content generated by my content management system, and you are not allowed to put anything overtop of a flash element. Also, things under the flash element will, of course, lose all functionality.

What is the solution? Send all the HTML data to flash and have flash, not the browser, render the html to the screen. And guess what, this actually does work! And not only does it work, but you can add more functionality than regular html and at the same time have search engines read all the info, as it is right inside the .html as any other page.

Ok, if you are not currently jumping up and down with joy, this document is not for you and you should leave now. If you have an uncontrollable, slightly maddening smile right now, then you will be happy to go through the purgation needed to make this happen. This is not for amateurs, and you will need to know all the following technologies discussed in this introduction.

Also, you should keep in mind that I am only going to provide you with the most basic of information required to do this, and I expect that you will be able to fit in the blanks. This is a very large and maddening puzzle.

And, finally, I am a self taught programmer, and just an intermediate one at that. I am sure that some of my code can be improved, and I would love it if you added a comment below showing everyone a superior method.

Action Script and Flash

You need to be very comfortable with Action Script in order to have any hope of succeeding with this. If not you will get lost, frustrated, and will throw your computer out the window, trust me! Action Script is not fun and is very picky and sometimes finicky: thus patience is very much required.

I am not going to teach you every little thing every step of the way. I expect that you can fill in the dots as so you can customize this idea to reflect your situation. I am using Flash 8, however I see no reason this will not work with Flex or even MX 2004 for the most part.

XML

For this project, only a moderate knowledge of xml is required. You must understand that the XHTML is going to be sent to flash, turned into a string, modified, then converted into an XML object, and rendered. In short, XHTML is a form of XML.

Joomla!

Joomla! Is currently my favorite Open Source Content Management System. We require an open source CMS because we need to modify some of the root code. Joomla! Runs on PHP and MySQL, and has a very large community for support. I use it with 90% of my websites.

There are many benefits of using a CMS, however the most important is the ease of modifying the content, the extra functionality and the modules and components you can easily add to it and modify.

Fonts

If you have ever worked with a print designer, you will know that they really want to use their fancy font, however in the world of websites we can only use a few basic 'universal' fonts like verdana and arial. This is no more!

Flash is able to embed fonts, and we can link to the fonts in our .CSS file that we will later create. Also, now that we have flash 8 we can add filters and gradients to our fonts in action script, as if we were in photoshop.

Javascript

You will only need a basic understanding of javascript for this. You need to know how the Document Object Model (DOM) works in order to locate the correct blocks. `getElementById()` will be a big help for us here. That is about it, but I could see this soon leading into building your own popup windows full of content from within flash using java script, which will be the subject of the next document I will write.

There is alot you can do with Javascript and Flash, so it cannot hurt to know a thing or two about Javascript.

Preparation

Joomla!

For this documentation I will be assuming that you are using Joomla! (Mambo should be the same). Install Joomla! to your server just like any other website. Set it so be 'SEO Friendly' if possible. Fill it up with a some content but keep things loose. We need to keep the content simple html, only using things like <p> and ect, as defined in the FP XHTML Render documentation, as discussed later. This is the time to setup and needed modules and components. Get everything working with plain html first before moving to flash.

Class' and Prototypes

You will need to install the FP XHTML Render (currently version 1.2.2) Sourceforge project from <http://sourceforge.net/projects/fpxhtmlrender> created by MAKO at <http://www.flashpushers.net> . Alternatively, you could always write your own XHTML render script, however this one is very well done. Thank you MAKO! You must read his documentation very closely to know exactly which tags work, and how. Some are required!

Another couple great resource for prototypes and class' are <http://www.sephiroth.it/> and <http://proto.layer51.com/> . These two websites have saved me many times.

Forums

If you are stuck, web forums are a great resource, but you need to pick carefully. I find good results from <http://www.actionscript.org> and <http://kirupa.com/> .

Customization

Templates and CSS

There is actually nothing you need to do for the template, except keep in mind that the content is going to be hidden until the flash sees it. In Joomla!, the content is defined by the id 'body_outer', and if it is not then set it to that. In your .CSS, define '.body_outer' to have 'visibility: none;', making it invisible.

When you embed your flash .swf, give it the name 'my_content_swf'.

CMS Customization

Now we have to customize the output of our CMS to be that which flash will be expecting. I will go over this using Joomla!. Open /components/com_content/content.class.php and /components/com_content/content.php in dreamweaver. In content.php navigate to where it says // show/hides the intro text, for me it is line 1255. You will have to edit this to what works for you, but for me I changed it to read:

```
if ( $params->get( 'introtext' ) ) {
    $row->text = '<div id=\'to_flash\'><span class=\'real_
header\'>' . $row->introtext . ( $params->get( 'intro_only' ) ? '' : chr(13)
. '</span><span class=\'real_content\'>' . $row->fulltext . '</span></
div>');
} else {
    $row->text = $row->fulltext;
}
```

What I am doing here is making sure to use div's and span's in the right spots. the render only can see span's, but the parent 'to_flash' tag can be a div. I am also making sure that the id's are setup properly. Only the elements inside the 'to_flash' div will be sent to flash. You may find other places you need to edit the content output in these two files.

Strict XHTML

You must make sure to read the documentation for the FP XHTML RENDER very closely, as it only allows certain tags. Also, all content must be setup with basic tags in XML format, so you can expect a certain header format followed by a series of <p> tags, so we can properly parse the string.

This is very important and will cause most of your problems. I cannot stress this enough! Read the documentation that comes with FP XHTML closely because as versions of FP XHTML change, so will the compatability. I am hoping that the future versions will be alot less strict, allowing for a much smoother CMS to Flash integration.

Action Script

Fonts

Flash is great, it can embed any font you want. There is plenty of documentation about this online. You import the font into your library, give it a linkage id and leave set the size and bold and that. Then leave it alone! Every .swf will have a .css associated with it, and that .css is where you define the font styles and the linkage id to the font you wish to use.

JavaScript

This is a very important step. Frames #1 and #2 will be dedicated to javascript and you flash will start on frame #3. in frame #1 put the following:

```
_root.getURL("javascript: top.document.my_content_swf.  
SetVariable(\"_root.html_content\",unescape(top.document.  
getElementById(\"to_flash\").innerHTML));");
```

This script is calling the HTML from the 'to_flash' div we defined earlier, and sending it to the 'my_content_swf' .swf on the page (this flash file), as the string _root.html_content.

On frame two, include this code:

```
if(_root.html_content == undefined){  
    this.gotoAndPlay(1);  
};
```

This creates a slow frame based loop. No need to use up the computer with a really fast loop or listener, so just use a frame based loop. You will thank me when you have a lot of these going at once!

I.E. vs F.F.

Internet Explorer acts differently than Fire Fox when using this javascript .innerHTML method. For this reason we have to change the whole string to lower case. We then have to parse this string using this function:

```
String.prototype.replace = function(a,b) {return this.split(a).join(b);}
```

For my content my I.E. fix parsing looks like this:

```
_root.html_content = _root.html_content.replace("  ","");
_root.html_content = _root.html_content.replace("class=re","class=\
"re");
_root.html_content = _root.html_content.replace("er>","er\>");
_root.html_content = _root.html_content.replace("ent>","ent\>");
_root.html_content = _root.html_content.replace(" size=24>","
size=\"24\">");
```

This removes the tabs (that is not spaces it is the tab character), and adds the quotes where they should be. You have to be creative here, and know what html you are expecting. Here I am expecting a font tag with the size of 24 set. Yes, this is a touch restricting, however it should be workable in most cases.

Add HTML and Parse Results

Since we had to set everything to lower case to appease I.E., now we have to parse the results... but we cannot do that properly as a string. We have to convert the string into an XML object. But wait! There isn't enough HTML code in our string! The FP XHTML RENDER script requires a few tags and a link to our .css file. Mine looks like this:

```
_root.test_xml = new XML();
_root.test_xml.ignoreWhite = true;
_root.test_call = "<html><head><link href=\"flash_css.css\"
rel=\"stylesheet\" type=\"text/css\"/></head><body><table><tr><td
width=\"500\" border=\"0\">" + _root.html_content + " </td></tr></
table></body></html>";
```

I used the following prototype from <http://proto.layer51.com/> from Nosferatu:

```
/* String.prototype.encase
A simple and quick way to ensure your strings
are being displayed with the proper capitalization,
no matters how their content is capitalized.
Concerning the pattern parameter, every alphabetical
char is mapped as "capitalization mask",
while a non-alphabetical char is mapped as
"keep the source capitalization for this char".
The last pattern char is wrapped till the end of string.
```

```
Ex: trace("hELLO WoRLd".encase("A."));
```

```
Displays: HELLo WoRLd
```

```
Ex: trace("hELLO WoRLd".encase("Aa"));
```

```
Displays: Hello world
```

```
Ex: trace("hELLO WoRLd".encase("A"));
```

```
Displays: HELLO WORLD
```

Have fun!

- Nosferatu -

```
*/
```

```
String.prototype.encase = function(pattern)
{
  if (typeof pattern != "string") return this;
  var a="a", _a="A", z="z", _z="Z", str="", i, j, pt, ch;
  String.prototype.__c = String.prototype.toString;
  for (i = 0, j = Math.min(this.length, pattern.length); i < j; i++)
  {
    ch = this.charAt(i);
    pt = pattern.charAt(i);
    String.prototype.__c = pt >= _a && pt <= _z ? String.prototype.
toUpperCase : pt >= a && pt <= z ? String.prototype.toLowerCase : String.
prototype.toString;
    str += ch.__c();
  }
  str += this.substring(i).__c();
  delete String.prototype.__c;
  return str;
}
```

I then have to go through my XML and find the proper nodes and change the first characters in each line and paragraph to be capitalized. I can only do this by knowing the exact format for the html I am collecting. My script looks like this:

```

    _root.test_xml.parseXML(_root.test_call);
    //capitalize quote
    _root.test_xml.childNodes[0].childNodes[1].childNodes[0].
childNodes[0].childNodes[0].childNodes[0].childNodes[0].childNodes[0].
childNodes[0].nodeValue = "\"" + _root.test_xml.childNodes[0].
childNodes[1].childNodes[0].childNodes[0].childNodes[0].childNodes[0].
childNodes[0].childNodes[0].childNodes[0].nodeValue.encase("Aa") + "\"";
    //capitalize author
    _root.test_xml.childNodes[0].childNodes[1].childNodes[0].
childNodes[0].childNodes[0].childNodes[0].childNodes[1].childNodes[0].
nodeValue = "~ " + _root.test_xml.childNodes[0].childNodes[1].
childNodes[0].childNodes[0].childNodes[0].childNodes[0].childNodes[1].
childNodes[0].nodeValue.encase("Aa");
    //capitalize body
    //ph1
    for(a=0; a<_root.test_xml.childNodes[0].childNodes[1].childNodes[0].
childNodes[0].childNodes[0].childNodes[1].childNodes.length; a++){
        test = _root.test_xml.childNodes[0].childNodes[1].
childNodes[0].childNodes[0].childNodes[0].childNodes[1].childNodes[a].
childNodes[0].nodeValue.split(" ");
        for(b=0; b<test.length; b++){
            test[b] = test[b].encase("Aa");
            trace(b+"<>" + test[b]);
        }

        _root.test_xml.childNodes[0].childNodes[1].childNodes[0].
childNodes[0].childNodes[0].childNodes[1].childNodes[a].childNodes[0].
nodeValue = test.join(" ");
    }

```

With this script I am allowing for as many paragraphs as there is. I am not, however, searching for question marks as the end of a sentence, only periods. This parsing method only works when you know the exact format of the XHTML that the flash should receive. In most instances, you do know the exact format.

Render HTML

Now that we have our properly formatted XML object ready, we can render ourselves some HTML inside Flash! This is where the magic happens. You can put the render code right on frame #3 at the end, or you can be like me and put it in another movieclip in the library, with a linkage id, so you can call it into your scroll box. My render code is close to the same as the examples (mine updates my scrollbars, and uses javascript errors) from FP XHTML RENDER, and looks like this:

```

//!-- UTF8
//#####
//enable tooltips
import FastProtoZoo.FPhtmlRender.htmlRender;
//attach on the fly
this.attachMovie(htmlRender.symbolName, "pgr", 1);
//set box dim
this.pgr.setDims(666,257);
//-----
//-----
//-----
this.pgr.setHTFCallback(this, "received");
//page rendered callBack
this.pgr.setCallback(this, "rendered", "error");
//enable html fixes
this.pgr.useHTMLFix=true;
//
this.received = function(value:String) {
    trace("asfunction###"+value);
    //parse if it is a close command
};
//if rendered get page parameters
this.rendered = function(value:String) {
    trace("rendered###"+value);
    _root.scroll.bar.update();
};
//error while loading
this.error = function(value:String) {
    trace("error###"+value);
    _root.getURL("javascript:alert(`error!>"+value+"`);");
};
//#####parse page
this.pgr.preparse(_root.test_xml);

```

Filters and Gradients

Now that we have the HTML content rendered inside flash and in its own movieclip, we can do anything we want to it! Personally, I enjoy adding filters or gradients to the text. I will leave you to discover gradients on your own, however here is an example of a basic glow filter:

```

// import the filter classes
import flash.filters.*
var drop:DropShadowFilter = new DropShadowFilter(1, 135, 0xf4f4f4,
70, 1, 1, 1, 1, false, false, false);
this.pgr.filters = [drop];

```

A large, stylized letter 'R' in a light green color, positioned to the left of the main title. The 'R' is partially overlaid by a horizontal yellow bar that extends across the page.

Rinse and Repeat

Troubleshooting and Testing

There are too many things that can go wrong with this method still, and a lot of that has to do with the fact that FP HTML Render is made for .html files, and requires some very strict formatting. IE breaks this formatting when it uses the unsupported javascript DOM .innerHTML property, yet FF handles this properly. If anyone knows how to get around this problem in a simple manner, or when FP HTML RENDER does not require me to lowercase and parse my strings to include quotations, then I will update and simplify this process.

Different .swf for each block

Until that time, however, you must endure the troubleshooting and slowly make this work for each individual module you wish to have flash in. In some designs, your flash may just be a cropped element and only showing a corner of the rendered XHTML, and another flash is showing the rest. The possibilities are endless!

Search Engine Optimization

This is my favorite part of all of this: Flash now has just as much SEO capabilities as a flat HTML page! All the text content is actually in the .HTML document, the only thing is that the style sets the visibility to hidden. The text is still there, and if hidden is not an option for you then you can just position it under the flash for browsers without flash support with some quick javascript. SEO with animation and filters! RSS readers in flash! This is an internet marketing person's wet dream.

Credits

Written By Steven Alan Stark

This file is allowed to be published on VancouverOnlineMedia.com and Digitopolis.com only.

Please leave comments and and ideas you may have to improve this procedure.

You can email me directly at steven@stevenstark.com

~Thank You