SIDEdev

Flash Front-End Project

Flash y XML: Un ejemplo practico

Autor: Joseba Alonso Perez
Internet: http://www.sidedev.net/articulos/xml
Publicacion Original: Revista e-site Nº16
Version de Flash: 5

Indice

Introducción

Hace ya algún tiempo cuando descubrí Flash se abrió un nuevo mundo de posibilidades para innovar en la creación de websites. Desde el principio me convertí en un adepto a este programa y lo comencé a utilizar para introducir efectos y animación en las webs que diseñaba combinándolo con HTML. A la hora de introducir los contenidos dinámicos como noticias o texto procedente de una base de datos se introducían mediante HTML. Pronto la necesidad de que esos datos llegara a introducirse dentro de la película flash se hizo cada vez mas patente y comencé a estudiar las posibilidades, pronto me di cuenta que eran muchas: Javascript, ASP, PHP, ColdFusion… De todas ellas Javascript era la única que no necesitaba de procesamiento en el Servidor. Normalmente este tipo de procesamiento en servidor implica un coste en el hosting del website y las capacidades de Javascript para manejar con facilidad datos externos dejaban a este tipo de proyectos con necesidad de grandes tiempos de desarrollo para el website. Pero con la llegada de Flash 5 y su renovado actionScript también nos llego un regalo de Macromedia en forma de objeto XML. Ahora ya no se hacia tan necesario ese procesamiento de servidor…..

Sobre XML

Pero lo primero que se pregunta casi seguramente el lector es: ¿Qué es eso del XML? Bueno, XML es un estándar de la industria para la creación de lenguajes de marcado. Lo que significa que provee de una serie de normas para que tú puedas crear tu propio documento de marcas y que sean portables a otras aplicaciones en cualquier tipo de sistema. ¿Suena bonito verdad? Ahora pensemos en nuestro viejo conocido HTML. Este nos provee de una serie de marcas que son interpretadas por el navegador para mostrar la información al usuario. Por ejemplo si necesitamos que determinado texto salga en negrita solo tendremos que "encerrar" el texto deseado entre las marcas <B> y </B> para que el navegador lo interprete y nuestro texto salga en negrita. Es realmente sencillo y seguramente es algo que has hecho muchas veces. ¿Pero te has puesto ha pensar que es realmente lo que estas haciendo? Estas dando información sobre la información. Esto es lo que se denomina metainformación. Piensa en el contenido de tu pagina web como en la información real que tu quieres mostrar a tus visitantes. Y separa mentalmente el contenido de la forma. Son 2 cosas totalmente distintas pero a la vez necesariamente unidas. El contenido es el texto que quieres que lean, las imágenes que están ilustrando ese texto, incluso esa pequeña animación en Flash en la que muestras como se realiza el complejo proceso industrial de el producto de tu cliente. Luego esta la forma. El tipo de la letra, su tamaño, colocación, alineamiento. Pero ahora piensa que realmente estas marcas de HTML no esta aportando mas que un formato. Realmente el navegador difícilmente sabrá si estas hablando de matemáticas, ciencias o estas contando un chiste. Todo lo que puede hacer es operaciones con cadenas. Buscar un patrón, mirar si coincide con otro… etc. Ese es el gran problema que existe hoy en dia para almacenar información en HTML. Por eso precisamente se creo XML, para que se pudieran crear múltiples lenguajes de marcas que proporcionasen una información extra sobre esa información. Así una marca que fuese algo parecido a: Hola, mi nombre es <NOMBRE>Joseba Alonso</NOMBRE>. Seria muy fácil de saber por cualquier navegador, aplicación o motor de Búsqueda que esas letras en concreto dentro del texto
Identifican un nombre de persona. Esto es la metainformación, información sobre la información y es la finalidad máxima del XML. La finalidad de un documento XML jamás será el proporcionar una forma de presentación para ese contenido. Para eso podemos utilizar Hojas de Estilo, XSL o nuestro viejo conocido Flash.

XML es una recomendación de el **World Wide Web Consorcium (W3C)**. Este organismo se formo para estandarizar los lenguajes de marcado que estaban apareciendo, como el HTML. En realidad cuando el W3C habla de que es una recomendación se quiere decir que es un standard. La recomendación completa la podéis encontrar en **http://www.w3.org/TR/2000/REC-xml-20001006** aunque puede ser una documentación un tanto compleja para los no iniciados. Podéis encontrar una traducción al castellano en **http://mipagina.euskaltel.es/gsagarduy/rec-xml-es.html**. También encontrareis un ejemplo de XML en la figura 1.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE raiz[
    <!ELEMENT raiz ( #PCDATA | tipo1 | tipo2)*>
    <!ELEMENT tipo1 (#PCDATA)>
    <!ELEMENT tipo2 (#PCDATA)>
]>
<raiz>
    <tipo1>Engranajes de maquinaria pesada</tipo1>
    <tipo2>Engranajes de maquinaria ligera</tipo2>
</raiz>
```

La muerte del HTML

No quiero hacer saltar las lágrimas a nadie. Pero HTML esta oficialmente muerto. La versión actual 4.01 convertida en recomendación en
Diciembre del 1997 es la última versión que tendrá HTML. El sucesor de HTML se llama XHTML y es una recomendación desde Enero del 2000 (**http://www.w3.org/TR/xhtml1/**) . Todo esto del XML y la muerte del HTML es la consecuencia directa del caos que sufre hoy en día la web. En parte causado por las deficiencias del HTML y en parte causado por la permisividad de los navegadores a la hora de interpretar un documento HTML. La búsqueda de información es muy poco eficaz en HTML. Lo que parece la solución a los males de hoy en día es separar el contendido de la presentación totalmente. Esta es la finalidad máxima de XML. Recordemos que XML no provee de ningún tipo de mecanismo para representar el documento sino que solo se encarga de separar y estructurar la información. La tarea de la presentación se ha trasladado completamente a lenguajes como CSS (Conocidas como hojas de estilo) y XSL. Nosotros utilizaremos Flash para representar esa información. Pero lo bueno de todo esto es que esa misma información que vamos a crear en forma de documentos XML podría ser utilizado junta a CSS para ser mostrada en el navegador o por cualquier otra aplicación. Una sola fuente de información, bien estructurada y un montón de formas de visualizarla. Todo un avance si lo comparamos a como solemos trabajar normalmente.

Estructura de un documento XML

XML nos deja crear nuestras propias marcas para ordenar la información. Por ejemplo podríamos crear unas marcas <NOMBRE> y </NOMBRE> para separar dentro de un carta el nombre de una persona. De nuevo volvemos al concepto de metainformación. Estamos dando una información adicional sobre ese texto. Esta información puede ser utilizada por una aplicación para poner el nombre del destinatario de la carta por ejemplo. Es lo que se llama un elemento. Hay 2 tipos de elementos, los contenedores y los vacíos. Los contenedores son los que llevan 2 marcas para delimitar la información y los vacíos solo llevan 1 marca, son elementos que proporcionan algún tipo de información pero que no tienen una relación directa con el texto. Por ejemplo piensa en la marca <IMG> o <HR> de HTML. Son elementos vacíos. La diferencia entre HTML y XML se encuentra en que en XML los elementos vacíos acaban con un símbolo de / al final de la marca. Por ejemplo <IMAGEN />.

Un ejemplo de XML

Tomemos el siguiente documento XML

```
<?xml version="1.0" encoding="UTF-8 ?>
<CARTA>
        <DESTINATARIO>Joseba Alonso Perez</DESTINATARIO>
        <CUERPO>
                Hola Joseba, necesitaría que me enviases de nuevo las imágenes.
                Pero las necesito en formato TIFF ya que mi ordenador no es capaz de leer
                el formato que me has enviado. Un saludo.
        </CUERPO>
        <IMAGEN />
</CARTA>
```

También podemos incluir atributos dentro de cada marca para añadir información adicional sobre el propio contenido. De hecho esto es tremendamente útil. Piensa que utilidad tan nula tendría el elemento <IMG SRC="imagen.gif"> en HTML si no tuviera el parámetro SRC. Volvamos al ejemplo anterior:

```
<?xml version="1.0" encoding="UTF-8" ?>
        <CARTA fechada="18/10/2001">
        <DESTINATARIO>Joseba Alonso Perez</DESTINATARIO>
        <CUERPO>
                Hola Joseba, necesitaria que me enviases de nuevo las imágenes.
                Pero las necesito en formato TIFF ya que mi ordenador no es capaz de leer
                el formato que me has enviado.Un saludo.
        </CUERPO>
        <IMAGEN fichero="maqueta.psd" />
</CARTA>
```

En XML, al contrario de HTML, un documento que tenga una estructura incorrecta o con fallos se considera como inservible y todo procesador XML, incluido Flash, reconocerá dicho error y no procesara el archivo. Esto es lo que en XML se llama error de mala formación. Hay que tener cuidado antes de enviar un fichero XML a Flash de que sea correcto, o lo que es lo mismo, de que este bien formado. Una forma sencilla de comprobar si el documento esta bien formado es abrirlo con el Internet Explorer (versión 5 o superior). Si el documento esta mal formado este nos dará un error y nos dirá donde ha encontrado el problema (al contrario de flash que simplemente nos da el error)

Puede ser necesario conocer también que un documento XML puede llevar asociado una declaración de tipo de documento (De ahora en adelante DTD) que lleva una marca especial dentro del documento XML: <!DOCTYPE ….]>. La DTD sirve para mantener al documento dentro de unas normas. Si el documento además de estar bien formado, cumple con las normas de su DTD se dice que el documento XML es de tipo válido. No profundizaremos en este tema ya que Flash no trabaja con las DTD aunque si las reconoce y las separa del documento XML en si.

Internet Explorer y XML

Muchas veces, para comprobar que un documento XML es correcto se utiliza el navegador Microsoft Internet Explorer (IE) para mostrarlo. Hay que tener en cuenta las 3 cosas que puede hacer IE para mostrarlo en pantalla:

1. Si el documento XML esta mal formado nos mostrará un mensaje de error.
2. Si el documento XML esta bien formado nos mostrara el documento fuente en pantalla. Tal y como lo hemos escrito pero con unos símbolos de - al lado de los nodos que expanden y contraen el árbol por debajo de ese elemento.
3. Si el documento XML esta bien formado y además tiene asociada una hoja de estilos (CSS o XSL) la aplicará y procesara internamente para mostrarnos un documento HTML resultado de la transformación que aplica la hoja de estilos a este documento XML. Pero este caso no se va a dar a lo largo de este artículo.

Lo que es decir. Un documento XML nunca contiene ninguna información referente a la forma en que tiene que mostrarse. Solo contiene información, datos, de una manera muy bien estructurada. Para mostrarse en un navegador puede usar una hoja de estilos. Lo que hacemos en estos ejercicios es precisamente coger esa formación estructurada y representarlo en pantalla.

Se podría decir que es una analogía de lo que hace el navegador.

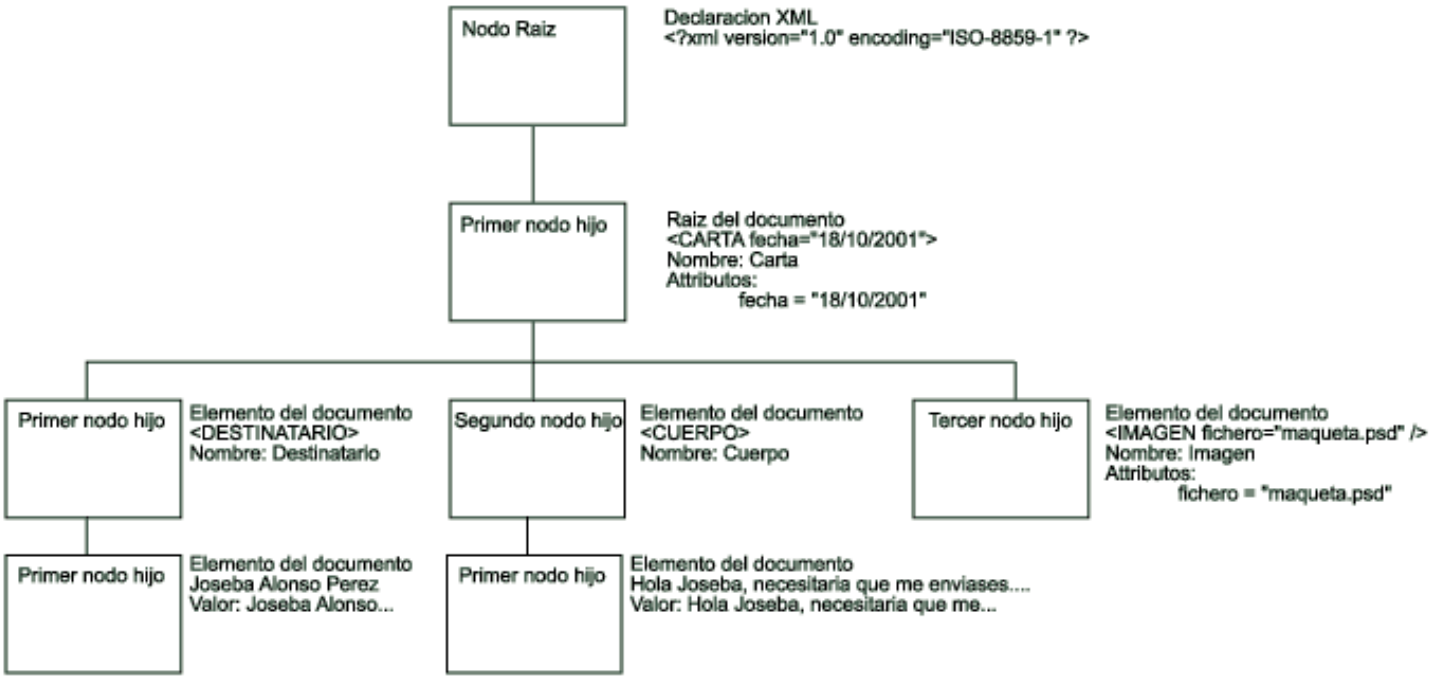
Internet Explorer:
Documento XML + Hoja de estilos = Pantalla maquetada

Macromedia Flash:
Documento XML + ActionScript = Pantalla maquetada

Es importante tener muy en cuenta que XML se pensó expresamente para poder separar la información en si, de la forma en que tiene que visualizarse. Que es precisamente el gran defecto de HTML.

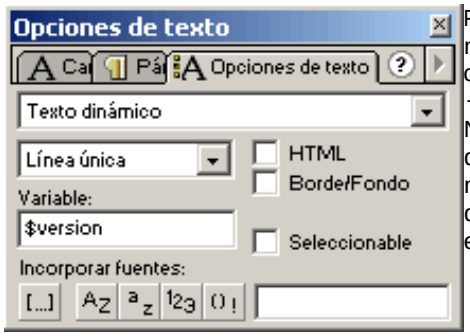
El DOM y parser de Flash.

El DOM (Document Object Model) es una forma de interpretar un documento a través de un modelo de Objetos. El parser (programa o subprograma encargado de interpretar el documento) carga el documento XML en la memoria del ordenador y forma un árbol de elementos que nos permiten interactuar con el documento. Cada rama de ese árbol esta formado por nodos que representan los elementos del documento XML. En la figura 2 podemos ver un esquema de cómo es representado el ejemplo anterior por el parser de Flash.



El primer nodo corresponde a la declaración de documento XML (<?xml version="1.0"…). Siendo su primer (y único) nodo hijo el elemento raíz del documento anterior (<CARTA fechada="…). Este nodo tiene dos propiedades, una de ellas es el nombre del propio nodo y otra es un array que contiene los atributos. En este caso solo es 1, el atributo fecha, que tiene un valor de *"18/10/2001"*. Este nodo *"CARTA"* tiene 3 nodos hijos. Siendo el primero de ellos el que correspondería a el elemento <DESTINATARIO> el segundo a <CUERPO> y el tercero a <IMAGEN>. Todos ellos tienen una propiedad con el nombre de el nodo e imagen también cuenta con otra para sus atributos (el atributo fichero). *DESTINATARIO* y *CUERPO* cuentan además con un nodo hijo cada uno de ellos. Con una propiedad valor para contener el texto de los elementos. Por tanto podríamos decir que Flash obtiene 2 tipos de nodos diferentes al procesar un documento XML: Nodos que representan a elementos y nodos que representan a los valores de esos elementos. No te preocupes si no entiendes todo. Poco a poco iremos viendo con detenimiento cada uno de los conceptos.

También hay que tener muy en cuenta que existen varias versiones del denominado Flash Player. La ultima es la Versión 5r42 pero la que lleva instalada el Flash por defecto el la versión 5r32. Las diferencias mas grandes entre estas versiones de Flash es precisamente el tratamiento que hace de XML. De hecho la versión r32 tiene grandes problemas al tratar el XML así que nosotros usaremos la versión r42 por ser la más estable.



Para averiguar la versión que tienes del player crea una caja de texto en un documento nuevo. Vas a la paleta opciones de texto y seleccionas tipo de texto dinámico. En la casilla variable escribes *$version* y pruebas la película en el reproductor flash (*CTRL +INTRO*). Te aparecerá en el cuadro WIN 5,0,32,0 (Si estas en un ordenador Windows) No te preocupes, esa es la versión del Flash Player que tiene incluida el programa, desgraciadamente no se puede cambiar. Ahora prueba esa misma película en el navegador (*Tecla F12*), si te sale WIN 5,0,42,0 todo correcto, tienes la ultima versión del Flash player instalada en el navegador. Si no es así, tendrás que actualizar el plugin en el navegador como se explica en el siguiente apartado.

Instalación de la versión r42 del Flash Player

Vamos a fijarnos en el HTML que crea Flash cuando publicamos el archivo (*Menú archivo/Publicar o tecla F12*). Junto a nuestro archivo .fla ahora tendremos otros 2, uno de ellos el .swf (la película flash) y otro .htm o.html. Abrimos el HTML con un editor de HTML cualquiera y nos fijamos en las líneas de código que incluyen el Flash dentro de la película. (figura 3). Si nos fijamos la propiedad codebase (indicado en rojo en la figura 3) esta diciendo que esa película flash que va a reproducir necesita la versión 5,0,0,0 de flash para poder ser mostrada correctamente. (Curiosamente no existe tal versión :) Pues tendremos que sustituir esos números por 5,0,42,0 para que si el navegador no tiene esta ultima versión del plugin, se la descargue automáticamente (En Internet Explorer). Con eso nos aseguramos que cuando alguien visite nuestra página no tenga problemas a la hora de mostrar los datos XML. Esto es muy importante ya que ninguno de los ejercicios que mostraremos a continuación funcionarían en una versión inferior.

```
1   <HTML>
2   <HEAD>
3   <TITLE>ejemplo1</TITLE>
4   </HEAD>
5   <BODY bgcolor="#000000">
6   <OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,0,0"
7    WIDTH=600 HEIGHT=400>
8    <PARAM NAME=movie VALUE="ejemplo1.swf">
9    <PARAM NAME=quality VALUE=high>
10   <PARAM NAME=bgcolor VALUE=#000000>
11   <EMBED src="ejemplo1.swf" quality=high bgcolor=#000000  WIDTH=600 HEIGHT=400 TYPE="application/x-
    PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash
12  </OBJECT>
13  </BODY>
14  </HTML>
```

Primer archivo XML

Vamos a plantear un sistema de noticias para una web íntegramente en Flash. Lógicamente un sistema de este tipo necesita ser alimentado externamente con los datos para no tener que estar modificando los archivos flash continuamente. Además tener las noticias fuera de la película flash en si nos ayudaría a que pudieran ser alimentadas por personas totalmente ajenas al entorno del desarrollo web. La idea de meterlos en archivos de texto y cargarlos vía loadVariables es una solución que puede funcionar con casos puntuales y datos concretos como texto pero el hacerlo en XML nos ofrece una serie de ventajas:

1. La información esta mejor estructurada. Con lo que manejarla será más fácil.
2. Los archivos pueden ser creados con cualquier aplicación externa por ser un Standard.
3. La lectura de los archivos en modo texto es mas clara
4. Nos beneficiamos de un montón de herramientas que provee el Objeto XML de Flash para cargar la información y procesarla.

Todo el ejercicio que vamos a realizar se va ha hacer sin requerir de ningún proceso en el servidor lo que nos permite que pueda ser usado con cualquier tipo de hosting (Alojamiento Web).

Se asumen también una serie de conceptos básicos en Flash como la Duplicación de MovieClips, el uso de Arrays y de funciones.

Se van a utilizar dos tipos de documentos XML que crearemos nosotros mismos según las necesidades. Uno de ellos será el principal que guardara algunos datos generales. Solo abra un documento XML de este tipo. El otro tipo de documento será el que contenga la noticia en si. De este tipo crearemos uno por noticia y lógicamente tendrá toda la información referente a esta.

Documento Principal

Este documento lo utilizaremos a la hora de mostrar la lista de noticias. Va a contener 2 tipos de elementos uno de ellos el nodo raíz que llamaremos *NOTICIAS* y varios nodos secundarios que llamaremos *REFERENCIA*. Necesitaremos un editor de texto o algún otro editor de código como Macromedia Homesite. Si vas a utilizar un editor de texto plano escoge uno que sea lo mas sencillo posible, como el notepad. No necesitaremos nada más que texto puro para escribir los archivos XML. Creamos un documento en blanco con el y escribimos:

```
 <?xml version="1.0" encoding="ISO-8859-1" ?>
```

Con esto indicamos al procesador (en este caso flash) que el archivo es de tipo XML versión 1 (no hay otra de momento) y que el tipo de codificación que vamos a usar para el texto es el de la normativa ISO-8859-1 lo que nos permitirá meter casi todo tipo de caracteres sin problemas, incluidos acentos. Esta marca debe ir la primera en cualquier documento XML, sino tendría un error de mala formación.

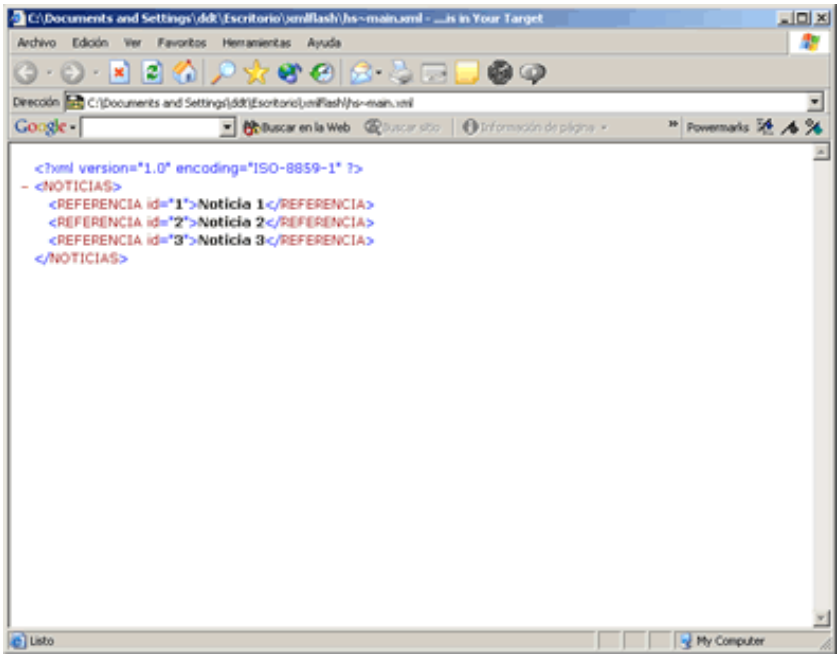
A continuación escribimos:

```
<NOTICIAS>
        <REFERENCIA id="1">Noticia 1</REFERENCIA>
        <REFERENCIA id="2">Noticia 2</REFERENCIA>
        <REFERENCIA id="3">Noticia 3</REFERENCIA>
< /NOTICIAS>
```

De esta manera creamos el nodo principal *NOTICIAS*. Todo documento XML debe tener un nodo principal y único, si no, estaría mal formado. Piensa que en HTML es lo mismo, todo documento comienza con *<HTML>* y termina con *</HTML>*, aunque los navegadores no son tan meticulosos con este tipo de cosas como los procesadores XML.

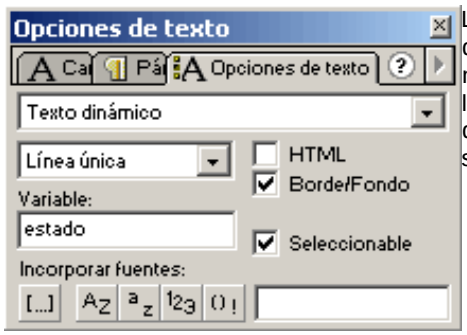
Dentro de la marca (o nodo) *NOTICIAS* nos encontramos con 3 nodos *REFERENCIA*, cada uno de ellos conteniendo un texto. Los usaremos mas tarde para guardar los titulos de las noticias que nos aparecerán en el listado. Es importante a partir de aquí tener en cuenta los siguientes conceptos, probablemente mas adelante necesitaremos volver a consultarlos:

1. El nodo principal es *NOTICIAS*. También se dice que es el primer (y único) hijo del documento XML
2. El nodo principal solo contiene nodos secundarios
3. Los nodos secundarios son *REFERENCIA*. Tienen una relación de hermandad entre ellos.
4. Cada nodo *REFERENCIA* tiene un atributo común llamado ‘id’ que les da un identificador único sobre los demás.
5. El primer nodo hijo de *NOTICIAS* es la referencia 1, no por llevar el id="1" sino por estar escrito el primero después del comienzo de la marca.
6. Todos los nodos *REFERENCIA* tienen un nodo padre común: *NOTICIAS*.
7. Cada uno de los nodos *REFERENCIA* tiene un nodo hijo del tipo Texto ("noticia 1","noticia 2" y "noticia 3" respectivamente)
8. *NOTICIAS* y *REFERENCIA* son nodos de tipo XML (tipo 1) y los textos "noticia 1", "noticia 2" y "noticia 3" son de tipo Texto (tipo 3)



Ya tenemos nuestro primer documento XML listo. ¿Fácil verdad? Lo guardamos en una carpeta con el nombre **main.xml** y lo abrimos con el Internet Explorer 5 o superior. Nos mostrará una página como la de la figura 4. En caso de que de algún error revisa el documento XML y verifica que todo esta correctamente escrito.

Primer contacto



Llega el momento de abrir el Flash. Creamos un archivo nuevo y lo guardamos al lado del documento XML. Esto es importante a la hora de manejar las urls. Si tenemos nuestra película flash al lado del documento XML no nos tendremos que preocupar por la URL a la hora de cargar el fichero. De todas maneras se puede dejar en cualquier directorio pero tendrás que acceder a el a través de rutas URL Standard, como las que se usan en HTML.

Creamos un campo de texto y en las opciones de texto lo marcamos como texto dinámico. En el campo variable escribimos "estado". Ahora hacemos clic sobre el primer fotograma, abrimos la ventana acciones y escribimos lo siguiente:

```
function cargaXML(){
      estado = "documento XML Cargado";
}
var docXML=new XML();
docXML.ignoreWhite = true;
docXML.onLoad = cargaXML;
docXML.load("main.xml");
```

Lo primero que hemos escrito es una funcion (*cargaXML*) que será llamada cuando el documento haya sido cargado. Mas adelante veremos como lo hace. Dentro de esa función simplemente le damos a la variable estado, que esta enlazada con el campo de texto, el valor de *"documento XML Cargado"* de esta manera sabremos cuando la operación de carga ha terminado.

```
var docXML = new XML();
```

En esta línea creamos el objeto donde vamos a almacenar el documento XML. *docXML* es un nombre de variable como cualquier otro. La clave de la creación de este objeto esta en la palabra new que crea un objeto XML nuevo (y vacio) que es almacenado en la variable *docXML* para poder tratarlo.

```
docXML.ignoreWhite = true;
```

En esta línea estamos poniendo la propiedad del documento XML ignoreWhite a verdadero. Con esto lograremos que el objeto XML ignore los espacios en blanco entre marca y marca del documento XML Esta característica es especial del player 5.0.41.0 y posteriores, por tanto no funcionará en ninguna otra versión anterior de el player. Tampoco se encuentra documentada en la primera versión de la documentación del Flash (la que te instala el Flash). Si quieres conseguir la ultima versión puedes buscarla en **http://www.macromedia.com/support/flash/documentation.html** . Si esta opción puesta en true los documentos XML que contengan espacios entre marca y marca (como el que hemos realizado nosotros) serán interpretados incorrectamente.

```
docXML.onLoad = cargaXML;
```

Con esta línea le estamos diciendo a Flash que cuando haya cargado el documento XML (onLoad) ejecute la función cargaXML () que hemos definido antes.

```
docXML.load("noticia.xml");
```

Por ultimo llenamos el objeto XML que acabamos de crear con el documento XML que hemos estrito antes: noticia.xml

Listo, le damos al botón *F12* para previsualizar la película Flash y en el navegador. Si todo ha ido correctamente veremos la caja de texto que hemos colocado en la película con la frase *"Documento XML cargado"*. Lo tienes completo en el fichero **xmlfla0.fla**. Perfecto, muy bonito, pero de momento no hemos hecho nada con él. Aun así ya hemos dado un paso importante. Hemos creado nuestro primer objeto XML y lo hemos llenado de datos a partir de un fichero exterior. A partir de ahora deberíamos comenzar a tratar esos datos para conseguir algo más útil.

Extrayendo los datos

Abrimos el archivo **flaxml1d.fla** que nos va a proporcionar un entorno para mostrar los datos. En realidad no son más que cajas de texto con una variable asociada como la que acabamos de hacer en el ejercicio anterior.

Hacemos clic sobre el primer fotograma de la capa ActionScript y sacamos la paleta de acciones. En ella escribimos lo siguiente:

```
function cargaXML(){
      primerNivel = this.firstChild;
      noticias = primerNivel.childNodes;
      n1 = noticias[0].firstChild;
      n2 = noticias[1].firstChild;
      n3 = noticias[2].firstChild;
      titular1 = n1.nodeValue;
      titular2 = n2.nodeValue;
      titular3 = n3.nodeValue;
      limpiaXML();
}
```

Es la función que habíamos visto antes. Es llamada al cargar el documento XML. En este caso le hemos añadido unas líneas de código para tratar el documento XML.

```
primerNivel = this.firstChild;
```

En este caso la palabra *this* se refiere al documento XML del que hemos cargado, del cual recogemos su primer nodo hijo (*NOTICAS* en nuestro caso) y lo almacenamos en la variable *primerNivel*. Si no te queda claro esto, vuelve a leerte las conceptos que se han explicado al crear el documento XML.

```
noticias = primerNivel.childNodes;
```

Ahora lo que hemos hecho es coger todos los nodos hijos del nodo *NOTICIAS* y los hemos almacenado en la variable *noticias*. En este caso los 3 nodos *REFERENCIA* han quedado almacenados en esta variable que automáticamente se convierte en un Array de 3 posiciones, una con cada nodo de noticias.

```
n1 = noticias[0].firstChild;
```

Ahora accedemos al primer (y único) hijo del primer nodo del Array noticias. Teniendo en cuenta el documento XML creado, *noticias[0]* correspondería al siguiente fragmento del XML:

```
<REFERENCIA>Noticia 1</REFERENCIA>
```

por tanto el primer (y único) hijo de ese nodo se trata de:

```
Noticia 1
```

Piensa que todavía no hemos cogido el valor del texto que es lo que nos interesa sino que tenemos colocado en la variable *n1* el nodo que contiene dicho texto. Luego lo extraeremos el texto de él. De momento vamos a realizar la misma operación con

la segunda y tercera noticia

```
      n2 = noticias[1].firstChild;
      n3 = noticias[2].firstChild;
```

Ahora ya tenemos los 3 nodos de texto localizados en las variables *n1*,*n2* y *n3*. Vamos a extraer el valor de cada una de ellas:

```
          titular1 = n1.nodeValue;
      titular2 = n2.nodeValue;
      titular3 = n3.nodeValue;
```

Para extraer el texto se utiliza la propiedad nodeValue que solo poseen los nodos de tipo texto. Colocamos dicho texto en las variables *titular1*, *titular2* y *titular3* que están relacionadas con los campos de texto dentro de la película.

```
 limpiaXML();
```

Todo perfecto, pero nos queda una ultima tarea, la limpieza. Algo esencial. Este proceso se define en las líneas siguientes que escribimos después de la función *cargaXML()*:

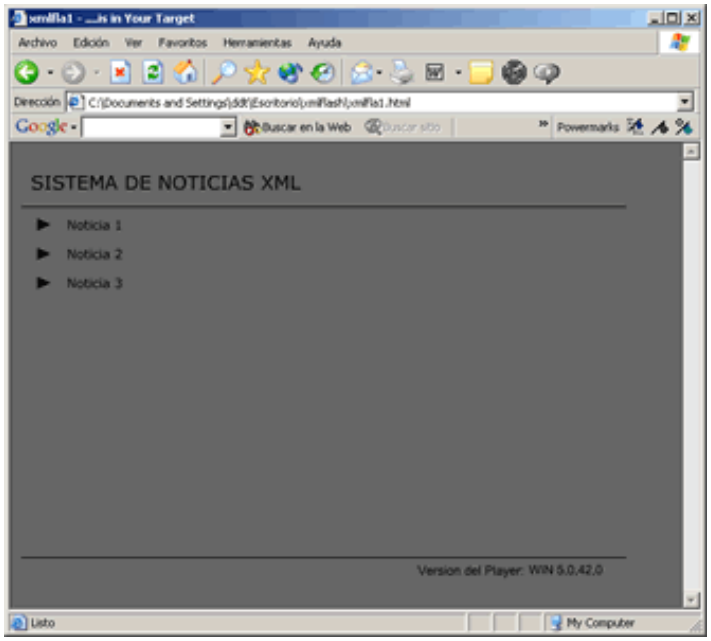
```
function limpiaXML(){
      delete primerNivel;
      delete noticias;
      delete n1;
      delete n2;
      delete n3;
      delete docXML;
}
```

Es algo sencillo de realizar pero esto es necesario para liberar a la memoria de todo el objeto XML. Se trata de eliminar todas las variables involucradas en el procesado del documento XML descrito anteriormente con la instrucción delete. Si no hiciéramos esto las variables se quedarían en la memoria con lo que el rendimiento general de la película Flash se vería afectado. Es algo con lo que hay que tener cuidado porque una vez extraídos los datos que nos interesan del documento XML no tenemos ninguna necesidad de que este siga en la memoria.

Por último escribimos las mismas líneas que en el ejercicio anterior para cargar el documento XML:

```
var docXML=new XML();
docXML.ignoreWhite = true;
docXML.onLoad = cargaXML;
docXML.load("main.xml");
```

Guardamos el archivo Flash junto con el documento *main.xml* y pulsamos *F12*. Ahora ya podemos ver los efectos de haber procesado el documento XML y extraído los titulares de las noticias. El fichero con el codigo completo es **xmlfla1.fla**. Sino me equivoco una sonrisa se debe de estar dibujando en tu cara...



Lo siguiente que deberías hacer es probar ha cambiar el documento XML fuente e ir mirando los resultados con el Flash.

Multiplica y vencerás

Bueno, ya nos encontramos nadando en el interesante mundo del Objeto XML. Pero nuestro sistema de noticias aun tiene graves carencias. Solo esta preparado para almacenar 3 titulares ni más ni menos. Vamos a arreglarlo ahora mismo para que sea un poco mas flexible. Abrimos el fichero **flaxml2d.fla** que nos volverá a ofrecer un entorno donde colocar nuestros datos. En esta película hay un *MovieClip* que utilizaremos como plantilla para duplicarlo y colocar en la escena tantos titulares como nos hagan falta (siempre y cuando haya espacio en la pantalla, claro). Dentro de él solo hay un botón y un campo de texto asociado a la variable titular. Hacemos clic de nuevo sobre el fotograma de la capa ActionScript y escribimos la siguiente función:

```
function cargaXML(){
      var primerNivel = this.firstChild;
      var noticias = primerNivel.childNodes;
      var posicionY = 65;
      noticias.reverse();
      for (x in noticias){
        noticia = noticias[x].firstChild;
        duplicateMovieClip("plantilla","titular"+x,x);
        _root["titular"+x].titular = noticia.nodeValue;
        _root["titular"+x]._y = posicionY;
        posicionY+=25;
      }
      limpiaXML()
}
```

La diferencia al anterior se encuentra en el bucle *for* que hemos insertado en el código. La variable noticias es un Array que se puede recorrer con el bucle *for* … in como cualquier otro Array, esto nos da la ventaja de pasar por todos y cada uno de ellos almacenando en *x* la posición en el Array que de en cada vuelta. Curiosamente ese *for* recorre la estructura de final al principio así que si queremos tener las noticias ordenadas como en el fichero XML tendremos que escribir la instrucción:

```
 noticias.reverse();
```

que le dará la vuelta a el Array.

```
noticia = noticias[x].firstChild;
```

Con esto almacenamos en la variable *noticia* el nodo hijo de la noticia "x", o lo que es decir, la noticia que este procesando en ese momento el bucle *for*.

```
duplicateMovieClip("plantilla","titular"+x,x);
```

Duplicamos el MovieClip plantilla y le damos un nombre y un nivel.

```
 _root["titular"+x].titular = noticia.nodeValue;
```

Con esto sacamos el valor del nodo de texto *noticia* y lo insertamos en la variable *titular* que contiene el *MovieClip* que acabamos de duplicar.

```
 _root["titular"+x]._y = posicionY;
posicionY+=25;
```

Por ultimo usamos la variable *posicionY*, que aumentamos en cada vuelta de bucle para ir colocando las noticias en la zona visible de la pantalla.

Guardamos la película Flash al lado del documento XML y la probamos dando al *F12*. Todo deberia aparecer como la vez anterior. Pero ahora la diferencia radica en que podemos añadir y quitar noticias en el archivo XML y el Flash se acomodaría a ello. El ejercicio esta completo en el fichero **xmlfla2.fla**.

El momento perfecto para que pruebes a modificarlos.

Creación de los enlaces

Llega el momento de volver a plantear el proyecto. Vale, ya sabes sacar los datos del documento XML pero ahora necesitas enlazar cada titular con cada noticia que posteriormente escribiremos en otros ficheros XML.

Modificamos el documento XML y lo dejamos de esta manera:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<NOTICIAS>
        <REFERENCIA url="noticia1.xml">Noticia 1</REFERENCIA>
        <REFERENCIA url="noticia2.xml">Noticia 2</REFERENCIA>
        <REFERENCIA url="noticia3.xml">Noticia 3</REFERENCIA>
</NOTICIAS>
```

Lo que hemos hecho ha sido añadir a las marcas *REFERENCIA* un atributo url que va a contener la dirección del archivo XML que contiene la noticia a la que hace referencia. Esto tiene las siguientes tres repercusiones en el objeto XML.

1. Ahora los nodos REFERENCIA tienen una propiedad más llamada attributes.
2. Esta propiedad atributes contiene una colección de atributos.
3. En esta colección de atributos se encuentra un solo atributo llamado url que contiene un texto.

Volvemos al fichero que estabamos trabajando antes en flash y añadimos la siguiente linea dentro del bucle for en la funcion *cargaXML()*:

```
      _root["titular"+x].direccion = noticias[x].attributes.url;
```

Lo que hace es coger el elemento url dentro de la colección *attributes* que he comentado antes y lo almacena en la variable direccion de el MovieClip que acabamos de duplicar. Con lo que al final obtenemos 2 variables por noticia, guardadas es sus correspondientes *MovieClips*: El titular de la noticia y la dirección URL donde esta almacenada.

Ahora vamos a editar el *MovieClip* plantilla que se encuentra en la película Flash. Abrimos la Biblioteca (*CTRL + L*), y hacemos doble clic sobre el símbolo llamado *titular*. Ahora seleccionamos el botón en forma de flecha, abrimos la paleta acciones y escribimos el siguiente código:

```
 on(release){
      _root.cargarNoticia(direccion);
}
```

Con esto lo que haremos es llamar a una función en la línea de tiempo principal (que mas adelante escribiremos) llamada *cargarNoticia()* a la que le manda como parámetro la variable direccion que hemos escrito ahí en el anterior apartado. Esta función es la que se encargará de gestionar la llamada a la noticia. Esto se encuentra completo en el fichero **xmlfla3.fla**

Las noticias XML

Llega el momento de ir añadiendo las noticias a este ejercicio. Para esto vamos a usar los ficheros *noticia1.xml*, *noticia2.xml* y *noticia3.xml*.

Estos documentos XML tienen esta estructura:

```
        <?xml version="1.0" encoding="ISO-8859-1" ?>
<NOTICIA id="1" publicacion="22/12/2001">
        <AUTOR>Pedro Alvarez</AUTOR>
        <CUERPO>
                ……..
        </CUERPO>
</NOTICIA>
```

En este caso sacamos las siguientes conclusiones derivadas de la estructura del documento:

1. El nodo principal, primer hijo, es *NOTICIA*.
2. Este nodo *NOTICIA* tiene un atributo que es publicación.
3. El nodo *NOTICIA* tiene 2 hijos: *AUTOR* y *CUERPO*.
4. Estos 2 nodos *AUTOR* y *CUERPO* tienen una relación de hermandad.
5. A su vez *AUTOR* y *CUERPO* tienen cada uno un nodo hijo de tipo texto.

Guardamos estos archivos XML junto con el archivo **main2.xml**, y el archivo **xmlfla4d.fla** en una nueva carpeta.

Ahora abrimos el archivo **xmlfla4d.fla** flash y hacemos clic sobre el fotograma 1 de la capa ActionScript y modificamos lo siguiente en las funciones.

1. En la funcion *cargaXML()* después de noticias.reverse() añadimos la línea:

   ```
   total = noticias.length;
   ```

   Que nos va guardar en la variable total la cantidad de noticias que hemos recogido del documento XML.
2. Creamos la siguiente función:

   ```
   function cargarNoticia(direccion){
           url=direccion;
           for (x=0;x<total;x++){
                   removeMovieClip("titular"+x);

           }
           gotoAndStop(2);

   }
   ```

Lo que hace es recoger la llamada que hemos creado antes en los botones. Recibimos un parámetro direccion que es la URL donde se encuentra el archivo XML que queremos cargar. Guardamos esa dirección en la variable *url*. Luego con un *for* y usando la variable *total* que hemos creado antes, borramos todos los *MovieClips* que habiamos duplicado en la función *cargaXML()*. Por ultimo mandamos a la película al fotograma 2 que es donde se encuentra nuestro layout para mostrar la noticia.

Ya lo tenemos todo listo. Solo nos hace falta escribir el código que cargue y muestre las noticias, vamos a ello. Hacemos clic sobre el fotograma 2 de la capa ActionScript y escribimos lo siguiente:

```
function cargaXML(){
```

```
        primerNivel = this.firstChild;
        fecha = "FECHA: "+ primerNivel.attributes.publicacion;
        segundoNivel = primerNivel.childNodes;
        autor = "AUTOR: " + segundoNivel[0].firstChild.nodeValue;
        cuerpo = segundoNivel[1].firstChild.nodeValue;
        limpiaXML()

}
```

Una vez más igual nos hace falta releer las conclusiones sacadas del documento XML de noticias para comprender como extraemos el texto sin ninguna duda.

```
primerNivel = this.firstChild;
```

Extraemos el primer nivel, nodo *NOTICIA*, y lo guardamos en la variable *primerNivel*

```
fecha = "FECHA: "+ primerNivel.attributes.publicacion;
```

Sacamos de la colección de atributos del primer nivel el contenido de publicación y concatenándola a el literal "FECHA: " la colocamos en la variable *fecha*. Esta variable esta enlazada con un cuadro de texto.

```
segundoNivel = primerNivel.childNodes;
```

Cogemos todos los nodos de segundo nivel (*AUTOR* y *CUERPO*) y los almacenamos en la variable *segundoNivel* que pasa a ser un Array.

```
autor = "AUTOR: " + segundoNivel[0].firstChild.nodeValue;
```

Aquí nos referimos a el primer nodo del Array *segundoNivel* y hacemos referencia a su primer nodo hijo (Nodo tipo texto) del cual extraemos su contenido con la propiedad *nodeValue*.

```
cuerpo = segundoNivel[1].firstChild.nodeValue;
```

Lo mismo para el segundo nodo del Array

```
limpiaXML()
```

Llamamos a la función XML que es la que debemos escribir a continuación:

```
function limpiaXML(){
        delete primerNivel;
        delete segundoNivel;
        delete docXML;
}
```

Por ultimo escribimos el código que crea el objeto XML y lo carga. Acuérdate como antes hemos almacenado en la variable url la dirección donde se encuentra el documento XML de noticias.

```
docXML=new XML();
docXML.ignoreWhite = true;
docXML.onLoad = cargaXML;
docXML.load(url);
```

Y ya tenemos listo nuestro sistema de noticias. Se encuentra completo en el archivo **xmlfla4.fla**. Pruébalo con el *F12*, añade noticias, modifícalas… es el momento de experimentar un poco.

Joseba Alonso Perez
www.sidedev.net