

Programación en Castellano

<http://www.programacion.net>

Artículos

Todos los artículos
de la Web
en formato PDF

Contenido

1 -. Office 2000 y XML	4
2 -. El coprocesador matemático	11
3 -. XSLT en el IE5. (I)	25
4 -. XSLT, XSL, XPath: ¿qué es cada cosa?	29
5 -. Enviando passwords de forma segura con MD5 y PHP3	32
6 -. Montar una web con apache+php+postgreSQL en Linux	34
7 -. Acceso a base de datos de Access 2000 usando el ADO Data Control	40
8 -. Cómo obtener el nombre de archivo largo a partir de uno corto en Visual C++	44
9 -. Iconos en los favoritos del Explorer 5	47
10 -. Paso de variables Javascript entre páginas (I)	49
11 -. Un ratón con estela	52
12 -. Barra de menús desplegados (III): En marcos distintos	56
13 -. Protección con contraseña (II)	62
14 -. Barra de menús desplegados (II): Soporte W3C-DOM	64
15 -. Forzar un salto de página en la impresión	67
16 -. Impedir el uso del botón derecho del ratón	69
17 -. Mandar imprimir una página	71
18 -. Redirección con Javascript	72
19 -. Trabajar con mapas	74
20 -. Barra de menús desplegados (I)	76
21 -. Subrayado de enlaces (II)	81
22 -. Conversor pesetas / euros	82
23 -. Convertir XML en HTML utilizando XSLT	84
24 -. ¿Cuánto queda para el año 2000?	89
25 -. Navegación con menús desplegados	91
26 -. Subrayado de enlaces (I)	93
27 -. Protección con contraseña (I)	95
28 -. Cambio de imágenes	97
29 -. Cómo no quedar encerrado en los marcos	100
30 -. Paginar Datos	102
31 -. Conceptos básicos de acceso a bases de datos	106
32 -. Envío de correo utilizando CDONTS	111
33 -. Uso del objeto FileSystemObject	115
34 -. Upload de archivos con el componente W3 Upload	118
35 -. Desarrollo de un libro de visitas usando XML y ASP	122
36 -. El objeto diccionario	127
37 -. Codificación de URL's	129
38 -. Uso de Formularios	132
39 -. Evitar la cache del navegador	136
40 -. Creación de un contador	138
41 -. Uso de las variables de Servidor	139

42 -.	El archivo Global.asa	141
43 -.	Redirección de una página ASP	144
44 -.	Generar WML desde un Servlet	146
45 -.	Descompresion de ficheros ZIP.	149
46 -.	CREACION DE FICHEROS ZIP: Implementación un compresor de ficheros ZIP.	154

Office 2000 y XML

URL: <http://www.programacion.net/articulos/office2000xml.php>

Autor: Ramón Montero

Email: alf@ibium.com

Office 2000 no es solo una buena suite de aplicaciones informáticas, sino que también es una herramienta de gestión de documentos web muy avanzada. Sus opciones de XML permiten generar automáticamente los documentos web más avanzados del momento, además de permitir la interconexión de diferentes documentos provenientes de las distintas aplicaciones que forman Office 2000.

En este artículo se van a comentar algunas de las características de los "esquemas" que utiliza Office 2000 para trabajar con XML.

Notas: Para comprobar los ejemplos que siguen a continuación, es imprescindible disponer de Office 2000 y de Internet Explorer 5.

1. Introduccción

La versión anterior de Microsoft Office, o sea, la versión 97, ya incorporaba muchas opciones y herramientas para generar páginas web, aunque algunas de ellas no estaban muy depuradas, perdiéndose incluso alguna información importante cuando se convertía un documento Office al formato HTML.

La última versión de Microsoft Office, esto es, la versión 2000 (que internamente se define como **versión 9**), no solo ha mejorado sus posibilidades de creación y edición de páginas web, sino que ha incorporado verdaderas opciones avanzadas para la Web, tales como el funcionamiento en intranet, la utilización de CSS o la incorporación de XML y VML.

La idea de Microsoft ha sido utilizar el HTML en Office 2000 de forma similar a como se trabaja en los formatos propios de las diferentes aplicaciones, para lo cual, se ha visto obligada a perfeccionar el código que se genera cuando se trabaja como página web, auxiliándose especialmente de las normas CSS y XML, aunque también aprovecha algunas características propias del entorno que forman Office 2000-Internet Explorer 5. Todo esto hace posible que cuando se pasa del diseño en formato propio al Diseño Web, no se nota el cambio en las herramientas disponibles.

Cualquier usuario de Office 2000 que observe el código fuente (opción **Ver > Código fuente**

HTML) que genera un documento cualquiera de Word 2000 abierto como página web (opción **Archivo > Nuevo... > Página Web**), observará que el código es bastante complejo, aunque no se haya escrito ni una palabra. Existen dos motivos principales.

Uno es que cualquier documento precisa cierta información extra que no tiene relación directa con su contenido, como es el nombre del autor, la fecha de creación, la empresa propietaria, etc. Esta información se guarda en formato XML y se distingue porque se incluye entre las etiquetas `<xml>` y `</xml>`.

El otro motivo es que es necesario definir algunos estilos de trabajo para poder empezar, como son el estilo de texto **Normal** o el estilo de definición de la página (tamaño, márgenes, encabezado y pie de página,...). Estos estilos se definen en formato CSS y se incluyen entre las etiquetas `<style>` y `</style>`.

Según se va creando el documento, no solo se van añadiendo las etiquetas HTML necesarias para soportar el contenido, sino que se van modificando los elementos XML de información extra y las descripciones CSS de los estilos.

2. Espacios de nombres

Lo primero que hay que tener en cuenta para entender el código que genera Word 2000 cuando trabaja como Diseño Web, son los **espacios de nombres**.

De una forma sencilla, se puede decir que los "espacios de nombres" (namespaces) es un sistema normalizado que permite trabajar con varios documentos integrados en un mismo documento. Podemos imaginar que es como si ciertas partes de un código se sitúan en una capa, mientras que otras partes del mismo código están en otras capas diferentes, de tal manera que dichas capas se pueden superponer y mezclar, conformando un único documento.

Esto permite, entre otras cosas, poder trabajar con diferentes especificaciones dentro del mismo documento, sin que las posibles coincidencias entre las distintas definiciones produzcan interferencias entre sí. Por ejemplo, se puede trabajar con una etiqueta `<fin>` que se refiera al final del documento y otra etiqueta `<fin>` que sirva para señalar la dirección del final de una carrera de bicicletas, pues si cada etiqueta pertenece a un espacio de nombre distinto, actuarán sin interferirse mutuamente.

En el código de un documento Office 2000 se declaran los espacios de nombres al principio del código. Si observamos el caso de un documento Word 2000 en blanco, veremos que empieza con:

```
<html xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:w="urn:schemas-microsoft-com:office:word"
xmlns="http://www.w3.org/TR/REC-html40">
```

Podemos ver cómo se aprovecha la etiqueta HTML para incluir los tres espacios de nombre (atributo `xmlns`) correspondientes al espacio de nombre por defecto adjudicado a la

especificación HTML 4 (`xmlns="http://www.w3.org/TR/REC-html40"`), al espacio de nombre correspondiente al esquema de Office 2000 (`xmlns:o="urn:schemas-microsoft-com:office:office"`) y al espacio de nombre relacionado con Word 2000 (`xmlns:w="urn:schemas-microsoft-com:office:word"`).

También hay que observar que a cada espacio de nombre se le adjudica un identificador: una **o** para el de Office 2000 y una **w** para Word 2000. En el caso del espacio de nombre por defecto (la norma HTML), que es el que más se va a utilizar, el sistema se ahorra el identificador.

Estos identificadores van a servir para controlar las etiquetas que se van añadiendo al código según se va creando el documento. Si se añaden opciones de Excel 2000, su identificador será **x**, para las opciones de Access 2000 se utiliza el identificador **a**, PowerPoint se identifica con **p** y los dibujos vectoriales se distinguen con el identificador **v**. Además de los posibles espacios de nombres mencionados, en los documentos de Office 2000 podemos encontrar otros menos usuales, como el correspondiente a "mecanografía de datos" (data-typing), que se identifica con **dt**.

3. Comentarios condicionales

Otra característica que hay que comentar para entender el código generado por Office 2000 cuando trabaja en formato HTML es lo que se conoce como "comentario condicional" o CC (Conditional Comment). Este procedimiento, en el momento de escribir este artículo, solo es soportado por Office 2000 y IE5.

El funcionamiento de los CCs es muy simple, pero muy potente. Se basa en aprovechar el sistema clásico de marcación de comentarios en el código HTML, que utiliza las marcas de principio `<!--` y de final `-->` para ocultar cualquier texto incluido entre ellas.

Si en un código HTML se incluyen la siguiente línea:

```
<!-- Este comentario no se puede ver -->
```

el texto quedará oculto a los ojos del visitante de la página web.

Otra utilización típica de estas marcas es la que las utiliza para ocultar cierto código. Por ejemplo, si entre el código HTML de una página web nos encontramos con las siguientes líneas:

```
<script type="text/javascript">
<!--
document.write("Hola")
-->
</script>
```

Las marcas `<!--` y `-->` sirven para que los visualizadores que no soportan JavaScript ignoren el código existente entre ellas, que será procesado solo por los navegadores que sí admitan esta tecnología.

El sistema Office 2000-IE5 soporta una variación de este proceso, ya que pueden añadir condiciones a estas marcas que les indican si el código que incluyen debe ser procesado o no.

Para utilizar CC hay que añadir `[if x]` a la marca de principio y `[endif]` a la marca de final, donde **X** es la condición que se precisa para activar el código que se incluye entre las marcas de CC.

Por ejemplo, las siguientes líneas:

```
<!--[if IE 5]>
Usted está navegando con Internet Explorer 5
<![endif]-->
```

Permiten mostrar el mensaje solo si el navegador utilizado es el IE5.

Por el contrario, el código siguiente:

```
<![if ! IE 5]>
Usted no está navegando con Internet Explorer 5
<![endif]>
```

Hace que se muestre el mensaje solo si el navegador utilizado no es el IE5.

Office 2000 aprovecha el CC para incluir información especial que solo él puede utilizar. Así, el siguiente código:

```
<!--[if gte mso 9]>
<xml>
<o:DocumentProperties>
<o:Author>Ramón Montero</o:Author>
<o:Revision>1</o:Revision>
<o:TotalTime>1</o:TotalTime>
<o:Created>2000-05-04T18:29:00Z</o:Created>
<o:Pages>1</o:Pages>
<o:Company>RAMON.ORG</o:Company>
<o:Lines>1</o:Lines>
<o:Paragraphs>1</o:Paragraphs>
<o:Version>9.2812</o:Version>
</o:DocumentProperties>
</xml>
<![endif]-->
```

Es interpretado por Office 2000 (que es la versión 9) mediante `[if gte mso 9]` para que pueda trabajar con la información incluida como un documento XML que utiliza el espacio de nombre **o** (de office).

Cualquier otro programa o navegador que abra una página web con este código, ignorará la información XML.

Office 2000 utiliza los CCs en otras ocasiones diferentes a la comentada, siendo corriente encontrar CCs insertados en el código HTML.

4. Organización de recursos

Cuando se salva un documento HTML es necesario controlar perfectamente los archivos adicionales al documento, como los gráficos, los sonidos, los vídeos, los frames de los

diseños con cuadros, etc.

Office 2000 realiza este control automáticamente, de forma similar a como actúa IE5 cuando se da la orden de guardar una página web "completa", o sea, creando una carpeta especial con el mismo nombre del archivo HTML (y el añadido **_archivos**), donde incluye todos los recursos necesarios para construir adecuadamente la página web correspondiente al archivo HTML.

En dicha carpeta, además de los archivos de los recursos, se encuentra un archivo XML denominado **filelist.xml**. En dicho archivo se incluye una relación de datos necesarios para gestionar los recursos de la página web.

Un ejemplo del contenido de **filelist.xml** puede ser:

```
<xml xmlns:o="urn:schemas-microsoft-com:office:office">
<o:MainFile HRef="../prueba.htm" />
<o:File HRef="image001.jpg" />
<o:File HRef="image002.jpg" />
<o:File HRef="image003.gif" />
<o:File HRef="image004.wmz" />
<o:File HRef="image005.gif" />
<o:File HRef="image006.gif" />
<o:File HRef="header.htm" />
<o:File HRef="filelist.xml" />
</xml>
```

No es complicado entender el contenido del archivo **filelist.xml**, y si el lector piensa que no parece tener ninguna utilidad, debe entender que soporta información única y adecuada para que Office 2000 pueda reconstruir perfectamente el documento cuando se vuelva a abrir la página web.

5. Controles ActiveX

En algunas ocasiones, no es suficiente con las restringidas posibilidades del HTML para mantener ciertas características especiales de gestión avanzada de documentos. En estos casos, Office 2000 recurre a la inserción de controles ActiveX, capaces de solucionar cualquier necesidad.

Pongamos el caso de una hoja de cálculo generada con Excel 2000, que se guarda en formato HTML, pero manteniendo la interactividad.

El código que se genera es similar al siguiente:

```
<div id="Libro1_1423" align=center x:publishsource="Excel">
<object id="Libro1_1423_WebCalc"
codebase="file:F:\msowc.cab#version=9,0,0,2710"
classid="CLSID:0002E510-0000-0000-C000-000000000046">
<param name=DisplayTitleBar value=false>
<param name=DataType value=HTMLData>
.....
.....
</object>
</div>
```

Se puede distinguir fácilmente la inclusión del control ActiveX (<object ...>) que controla

la hoja de cálculo, pero ¿qué pasa con los navegadores que no soportan la tecnología ActiveX?

Office 2000 resuelve esta cuestión incluyendo dentro de la etiqueta <object>el código:

```
<object ...>
.....
.....
<param name=HTMLData
value="<html xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:x="urn:schemas-microsoft-com:office:excel"
xmlns="http://www.w3.org/TR/REC-html40">
.....
.....
<p style='margin-top:100;font-family:Arial;font-size:8.0pt'>
Para usar esta página Web interactivamente,
debe utilizar Microsoft® Internet Explorer 4.0 o superior
y Microsoft Office Web Components. Visite el
<a HRef=
"http://officeupdate.microsoft.com/office/redirect/
fromOffice9/MSOWCPub.htm?&HelpLCID=3082">
sitio Web de Microsoft Office</a> para obtener más información.</p>
</object>
```

Otra vez nos encontramos referencias a los espacios de nombres, que si en principio se definieron para complementar el XML, Microsoft los utiliza en muchas y diversas ocasiones.

6. Diseño VML

Office 2000 puede trabajar con gráficos vectoriales desde cualquiera de sus aplicaciones, por lo que, para mantener en el diseño HTML las mismas posibilidades que en el diseño "normal", se ha visto obligado a incluir un lenguaje de diseño de gráficos para la web con capacidades vectoriales, y por supuesto, se ha basado en XML para ello.

VML (Vector Markup Language) es el desarrollo que permite el tratamiento de los gráficos vectoriales en las páginas web. Es un "lenguaje" de marcas con un funcionamiento similar al HTML, o sea, se soporta en una serie de etiquetas prefijadas de antemano capaces de permitir el dibujado en pantalla (rendering) de objetos gráficos, que al contrario de los gráficos de mapa de puntos, se basan en un soporte matemático controlado por fórmulas y sistema de ecuaciones.

Para hacernos una idea muy sencilla de cómo funciona el VML, es suficiente con observar las siguientes líneas de código:

```
<v:rect style="width:200px;height:50px" fillcolor="blue" />
<v:line from="50px 20px" to="350px 35px"
strokecolor="navy" strokeweight="3px"/>
```

En el código se puede intuir la descripción de dos figuras: un rectángulo de 200 pixels de ancho por 50 pixels de alto relleno de color azul y una línea que comienza en el punto de coordenadas 50,20 y finaliza en el punto definido por las coordenadas 350,35. La línea es de color azul oscuro y tiene un espesor de 3 pixels.

También se puede observar la utilización del espacio de nombre correspondiente al VML,

definido por la **v**, y en el caso del rectángulo, el aprovechamiento del CSS para su descripción.

Los lectores interesados en el VML pueden ver el tutorial existente en RAMON.ORG sobre el tema.

Por supuesto que para ver en la pantalla de nuestro ordenador las figuras definidas en VML, hay que disponer de un visualizador que incluya un intérprete del VML, que además de Office 2000, puede ser el navegador Internet Explorer 5, siempre que se haya instalado la opción VML.

Aunque en principio hubo varias empresas que participaron en el desarrollo del VML, en la actualidad Microsoft se ha quedado casi solo en su aplicación, ya que la mayoría de las casas prefiere esperar a que se termine de desarrollar el SVG (Scalable Vector Graphics), también basado en el estándar XML y muy parecido al VML, pero con la ventaja de que está arropado por un grupo de trabajo del W3C.

7. Conclusiones

Más que ser una guía de referencia de la utilización del XML en Office 2000, este artículo pretende ser una introducción al sistema de utilización del XML por parte de Office 2000 que Microsoft ha implementado para conseguir que cualquier usuario pueda trabajar en modo Web-HTML desde las aplicaciones que integran Office 2000, sin perder sus principales características, cosa imposible de realizar sin el apoyo del XML.

Office 2000 es sin duda alguna el primer proceso que hace un uso intensivo y completo del XML, y que combinado con IE5, forma un sistema completo y avanzado de diseño web automatizado, haciendo muy fácil la utilización de sus documentos en las intranets.

La mayor pega que se le puede achacar al código HTML generado por Office 2000 es la gran cantidad de información que no pueden entender otras aplicaciones distintas del propio Office 2000 o IE5, por lo que algunas utilidades especializadas en el diseño web están incluyendo opciones de eliminación automática del código específico no estandarizado, pudiendo obtenerse estos filtros incluso de la propia página web de Microsoft.

El coprocesador matemático

URL: <http://www.programacion.net/articulos/copro.php>

Autor: Astaroth

Email: astha@tlotb.com

Bueno, este documento es una breve iniciación al funcionamiento del coprocesador matemático en los sistemas PC y, más concretamente, a la arquitectura de los procesadores PENTIUM, puesto que hoy por hoy es lo más utilizado. No es mi intención dar una explicación completa de su arquitectura interior, simplemente trato de arrojar un poco de luz donde no la hay, porque cuando decidí meterme con el coprocesador matemático, no encontré apenas documentación y la poca (por no decir nula) estaba en inglés.

1. ¿Merece la pena usar el copro?

Esta pregunta es algo ambigua, porque depende mucho de la aplicación en sí. Si lo que queremos realizar son cálculos intensivos en 3D la respuesta es sí. Esto depende mucho por la incapacidad que tiene el coprocesador de intercambiar datos con los registros de propósito general, lo que obliga a costosas descargas en memoria con conversión a entero y que provoca que, por ejemplo, un cálculo en el que tengamos que utilizar directamente el resultado de la operación como un puntero a una zona de memoria, o como valor para introducir en una zona de memoria, sea algo lento.

Si por ejemplo lo que queremos es realizar un cálculo sobre unos datos dados (por ejemplo una multiplicación de matrices) sí que merece la pena su uso.

2. Estructura interna

El coprocesador trabaja internamente sólo en formato real, por lo que cualquier carga en los registros de coprocesador provocará que dicho valor sea convertido a coma flotante.

Sus registros están estructurados en forma de pila y se accede a ellos por el número de entrada que ocupan en la pila.

Los registros son $R(0)$ hasta $R(7)$, en total ocho registros de 80bits, como han de ser manejados en formato de pila, el coprocesador tiene un puntero de control de pila llamado

, Toda interacción que tengamos que hacer con los registros del coprocesador se realiza a traves del puntero de pila `st`, donde el último valor introducido es `st` o `st(0)` y si hubieramos rellenado todos los registros el ultimo seria `st(7)`... ¿ok? Por ejemplo:

- 1) Cargar en copro dato (1345)
- 2) ahora `St(0)=1345`
- 3) Cargar en copro dato (5431)
- 4) ahora `St(0)=5431` y `St(1)=1345`

3. Tipos de datos

El coprocesador puede obtener y escribir datos en memoria de los siguientes tipos.

Entero	Words(16bits),Dword(32 bits),Qwords(64 bits)
Real	Words(16 bits),Dword(32 bits),Qwords(64 bits),Twords(80 bits)

4. Un poco de materia...

Bien, ya hemos visto como ordena el coprocesador sus registros, ahora vamos a ver alguna operacion sencilla.

Todos los ejemplos que aquí escriba estarán enteramente en ensamblador, puesto que si lo que queremos es velocidad de poco nos sirve la `math387.lib` del C.

```

.386
.MODEL Flat
.STACK 800h

.DATA
Numero1 dd 25 ; Numero en formato entero
Numero2 dd 1.25 ; Numero en formato real
; ¡Ojo! este numero solo puede ser
; accedido correctamente por el copro
; a no ser que nos hagamos una rutina
; conversora de formato de FPU a entero
; cosa que no merece la pena, porque
; es mas facil cargarlo y convertirlo
; con el copro, y mas rapido)

Resul dd ? ; Variable para el resultado
Desca dd ? ; Variable para correccion de pila

.CODE

Start:
fild Dword Ptr ds:[Numero1] ;Cargar en el copro la
;variable Numero1 indicando
;que es un entero
fld Dword Ptr ds:[Numero2] ;Idem pero es un real
fadd St(0),St(1) ;Sumarlos
;St(0)=St(0)+St(1)
fstp Dword Ptr ds:[Resul] ;Descargar el resultado.
fstp Dword Ptr ds:[Desca] ;Descartar el otro valor.

mov eax,4c00h
int 21h
End Start

```

Como hemos visto al funcionar a base de pila siempre tenemos que dejarla tal y como se encontraba al iniciarse nuestra rutina, de ahí la operación de popeo con la variable `Desca`, aunque esto puede realizarse más rápido, siempre teniendo en cuenta varias cosas.

En vez de usar `fadd` podemos usar `faddp` que automáticamente suma `St(0)` y `St(1)` y elimina automáticamente el operando sobrante de la pila quedando en `St(0)` el resultado listo para ser popeado.

¡Ojo, las instrucciones de cálculo con la partícula "p" ej: `faddp` o `fdivp`, sólo trabajan con las posiciones de pila `St(0)` y `St(1)`!

También podríamos haber usado en vez del `fstp` a `Desca` un `ffree St(1)`, esta instrucción automáticamente libera ese componente de la pila, pero sólo es efectiva para dejar la pila tal y como estaba cuando el operando a liberar ocupa la última posición en la pila (esto es, el primero introducido).

```

Start:
    fild    Dword Ptr ds:[Numero1] ;Cargar en el copro la
                                ;variable Numero1 indicando
                                ;que es un entero
    fld     Dword Ptr ds:[Numero2] ;Idem pero es un real
    faddp                               ;Sumarlos y popear el
                                ;operando sobrante.
                                ;St(0)=St(0)+St(1)
                                ;pop st(1) > nul
                                ;Solo opera con estos
                                ;punteros de pila!!
    fstp    Dword Ptr ds:[Resul]    ;Descargar el resultado.
    mov     eax,4c00h
    int     21h
End        Start

```

o bien...

```

.CODE

Start:
    fild    Dword Ptr ds:[Numero1] ;Cargar en el copro
                                ;la variable Numero1
                                ;indicandole que es un entero
    fld     Dword Ptr ds:[Numero2] ;Idem pero es un real
    fadd     St(0),St(1)            ;Sumarlos
                                ;St(0)=St(0)+St(1)
    fstp    Dword Ptr ds:[Resul]    ;Descargar el resultado.
                                ;pop St(0) > Resul
    ffree   St(0)                  ;Descartar el otro valor.
                                ;Free St(0) (stack cleared)
    mov     eax,4c00h
    int     21h
End        Start

```

Como habéis podido observar liberamos con `ffree St(0)` esto es por lo anteriormente comentado. Al liberar el resultado de la pila `St(1)` pasa a ser `St(0)`, `St(2)-St(1)`, etc. Debido a su estructura en forma de pila.

5. Instrucciones

Estas son las instrucciones más comunes, he omitido algunas, por tratarse por ejemplo de

aritmética BCD (cosa poco común en el mundo del tiempo-real) pero para los interesados, Intel tiene a su disposición unos archivos en formato de Acrobat Reader con la lista y la organización interna de la FPU. Los ciclos de reloj de las instrucciones, son los declarados por Intel para el Pentium básico, esto puede verse alterado en las CPU's MMX, PRO y Pentium II. No voy a comentar las instrucciones porque en este momento carezco del tiempo necesario para realizar una documentación sobre las instrucciones de manera seria, pero de todas formas solo hay que ser un poco despierto para con solo los nombres hacerse una idea básica del funcionamiento de cada instrucción. Cualquier instrucción de cálculo (`fadd`, `fsub`, etc) toman por defecto si no se le especifica otros operandos `st(0)` y `st(1)`.

Leyenda: rm=Modo Real, vm=Modo Virtual, pm=Modo Protegido

Instrucción	Ejemplo	Ciclos de reloj
FABS	fabs	1
FADD [reg,reg]	fadd	3, 1
FADD memreal	fadd shortreal	3, 1
FADDP reg,ST	faddp st(6),st	3, 1
FIADD memint	fiadd int16	7, 4
FCHS	fchs	1
FCLEX	fclex	9+
FNCLEX	fnclex	9
FCOM	fcom	4, 1
FCOMP	fcomp	4, 1
FCOMPP	fcompp	4, 1
FICOM memint	ficom double	8, 4
FICOMP memint	ficomp darray[di]	8, 4
FCOS	fcos	18-124
FDECSTP	fdecstp	1
FDIV [reg,reg]	fdiv st(5),st	39
FDIV memreal	fdiv longreal	39
FDIVP reg,ST	fdivp st(6),st	39
FIDIV memint	fidiv warray[di]	42
FDIVR [reg,reg]	fdivr st(5),st	39
FDIVR memreal	fdivr longreal	39
FDIVRP reg,ST	fdivrp st(6),st	39
FIDIVR memint	fidivr warray[di]	42
FFREE ST(i)	ffree st(3)	1
FILD memint	fild quads[si]	3, 1
FINCSTP	fincstp	1

FINIT	finit	16
FNINIT	fninit	12
FIST memint	fist doubles[8]	6
FISTP memint	fistp longint	6
FLD reg	fld st(3)	1
FLD mem32real	fld longreal	1
FLD mem64real		1
FLD mem80real		3
FLD1	fld1	2
FLDZ	fldz	2
FLDPI	fldpi	5,3
FLDL2E	fldl2e	5, 3
FLDL2T	fldl2t	5, 2
FLDLG2	fldlg2	5, 3
FLDLN2	fldln2	5, 3
FLDCW mem16	fldcw ctrlword	7
FLDENV mem	fldenv [bp+10]	37, 16-bit pm=32,32-bit pm=33
FMUL [reg,reg]	fmul st(5),st	3, 1
FMULP reg,ST	fmulp st(6),st	3, 1
FIMUL memint	fimul warray[di]	7, 4
FNOP	fnop	1
FPATAN	fpatan	17-173
FPREM	fprem	16-64
FPREM1	fprem1	20-70
FPTAN	fptan	17-173
FRNDINT	frndint	9-20
FRSTOR mem	frstor [bp-94]	16-bit rm or vm=75;32-bit rm or vm=95;pm=70
FSAVE mem	fsave [bp-94]	16-bit rm or vm=127+;32-bit rm or vm=151+;pm=124+
FNSAVE mem	fnsave [bp-94]	16-bit rm or vm=127;32-bit rm or vm=151;pm=124
FSCALE	fscale	20-31
FSIN	fsin	16-126
FSINCOS	fsincos	17-137

FSQRT	fsqrt	70
FST reg	fst st	1
FST memreal	fst longs[bx]	2
FSTP reg	fstp st(3)	1
FSTP mem32real	fstp longreal	2
FSTP mem64real		2
FSTP mem80real		3
FSTCW mem16	fstcw ctrlword	2+
FNSTCW mem16	fnstcw ctrlword	2
FSTENV mem	fstenv [bp-14]	16-bit rm or vm=50+;32-bit rm or vm=48+;16-bit pm=49+;32-bit pm=50+
FNSTENV mem	fnstenv [bp-14]	16-bit rm or vm=50;32-bit rm or vm=48;16-bit pm=49;32-bit pm=50
FSTSW mem16	fstsw statword	2+
FSTSW AX	fstsw ax	2+
FNSTSW mem16	fnstsw statword	2
FNSTSW AX	fnstsw ax	2
FSUB [reg,reg]	fsub st,st(2)	3, 1
FSUB memreal	fsub longreal	3, 1
FSUBP reg,ST	fsubp st(6),st	3, 1
FISUB memint	fisub double	7, 4
FSUBR [reg,reg]	fsubr st,st(2)	3, 1
FSUBR memreal	fsubr longreal	3, 1
FSUBRP reg,ST	fsubrp st(6),st	3, 1
FISUBR memint	fisubr double	7, 4
FTST	ftst	4, 1
FUCOM [reg]	fucom st(2)	4, 1
FUCOMP [reg]	fucomp st(7)	4, 1
FUCOMPP	fucompp	4, 1
FWAIT	fwait	1-3
FXAM	fxam	21
FXCH [reg]	fxchg st(3)	1

FXTRACT	fxtract	13
FYL2X	fyl2x	22-111
FYL2XP1	fyl2xp1	22-103

6. Hints, o cómo acelerar

Lo primero es que, como ya hemos visto con los ciclos de las instrucciones, el volcado y carga de memoria es lento, con lo cual, la primera regla, es realizar las menores cargas y volcados posibles, aprovechando al maximo las posiciones de la pila para cuantas más operaciones mejor.

Por ejemplo, hemos de multiplicar un array por 5...

```

.DATA
Array dd 200 dup(?) ;Array con datos.. :D
Value dd 5

.CODE
fild Dword Ptr ds:[Value]
mov ecx,200 ;200 datos a multiplicar
mov edi,offset Array

Bucle: fild Dword Ptr ds:[edi] ;fild suponiendo que son
;enteros..
fmul ;Multiplica St(0),St(1)
fistp Dword Ptr ds:[edi] ;descargamos el operando
add edi,4
dec ecx
jnz Bucle
ffree St(0) ;Liberar el dato "5" :)

```

o bien, para ahorrarnos el ffree... :)

```

fild Dword Ptr ds:[Value]
mov ecx,199 ;200 datos a multiplicar
mov edi,offset Array

Bucle: fild Dword Ptr ds:[edi] ;fild suponiendo que son
;enteros..
fmul ;Multiplica St(0),St(1)
fistp Dword Ptr ds:[edi] ;descargamos el operando
add edi,4
dec ecx
jnz Bucle
fmulp ;St(0)=St(0)*St(1)
;pop St(1) > nul
fistp Dword Ptr ds:[edi]

```

Esto en realidad es un ejemplo tonto, pero como podemos observar la enseñanza del mismo es directamente aplicable a muchas rutinas matemáticas de ámbito 3D.

Otra cosa a tener en cuenta es que, si las cargas y descargas ya de por sí son lentas (hay que tener en cuenta `cache hits` en las cargas) es más lento todavía con la inclusión de la partícula "i", ya que la FPU ha de convertir ese dato entero a real, esto provoca que siempre perdamos unos pocos ciclos en la operación. Aun así, lo expuesto arriba ya de por sí es más rápido que lo mismo escrito en código de procesador (osea con `imuls`, etc).

Esto es fácilmente solucionable, si nuestra aplicacion necesita realizar diversos cálculos

matemáticos con un array de vectores, con lo cual lo realmente óptimo es tener almacenados esos datos en real, operar con ellos, y en la ultima operación a realizar para su utilización (una proyección 3D a 2D, por ejemplo) aprovechar para hacer la conversión a entero.

También hay otras optimizaciones, hay algunas instrucciones pairebles, concretamente, el `Fmul` con el `Fxchg`, pero de esto hablaré en una sucesiva actualización de documento (¡maldito tiempo!).

7 Ejemplo Práctico: Una rutina de proyección 3D a 2D

Los arrays son de Dwords en formato X,Y,Z con 12 bytes cada vector. El de entrada es real, el de salida sera entero.

```
;In Ecx=Number of vertex
; Edi=pointer to destination
; Esi=pointer of 3D vertex
;
;Out Action Performed!
;-----
ProyCords Proc Near
          fild ds:[Ycenter]          ;World Centering on screen
          fild ds:[Xcenter]          ;Idem.
          fild ds:[Pvi]              ;Focal Distance.
c3d2d:
          fld ds:[esi+8]              ;Get "Z"
          fldl                          ;Load number "1"
          fdivrp                      ;Calc Inverse and pop
                                      ;the not valuable operand
          fld ds:[esi+4]              ;Get "Y"
          fmul st,st(2)               ;Multiply Y with Focal Distance
          fmul st,st(1)               ;Multiply with the inverse
                                      ;of the "Z"
          fadd st,st(4)               ;Add Y Screen centering
          fistp Dword Ptr ds:[edi+4] ;Store final value
          fld Dword Ptr ds:[esi]      ;Get "X"
          fmul st,st(2)               ;Multiply X with Focal Distance
          fmul st,st(1)               ;Multiply with the inverse
                                      ;of the "Z"
          fadd st,st(3)               ;Add X Screen centering
          fistp Dword Ptr ds:[edi]    ;Store final value
          fstp ds:[Dummy]             ;Free the Z inverse
          add esi,12                  ;Increment pointers
          add edi,12                  ;Increment pointers
          dec ecx                     ;decrement counter
          jnz c3d2d                   ;if not 0 go to other vertex
          fistp ds:[Dummy]
          fistp ds:[Dummy]
          fistp ds:[Dummy]            ;Free all
          ret
ProyCords Endp
```

Como se puede observar en esta rutina, hacemos uso de los dos ejemplos de optimización expuestos anteriormente. Tomar los datos, desde una base de datos de reales, utilizar la conversión 3D a 2D para pasarlos a enteros (para poder realizar luego el pintado) y utilizar al máximo de lo que se pueda las posiciones de la pila para guardar datos que han de ser reutilizados. Otra optimización es el uso de inversos, para lo cual primero calculamos $ZI=1/Z$ y luego para obtener $X'=X*PVI/Z$ realizamos la siguiente operación: $X'=(X*PVI)*ZI$.

Desde luego esto es más rápido que dos divisiones pero, de todas formas, aunque se realizara con dos divisiones de copro, el resultado sigue siendo todavía más rápido que su misma

ecuación programada con el procesador.

Además, los inversos se pueden utilizar hasta en los sitios mas insospechados. Por ejemplo, en nuestro sistema de sonido, en el player de Sb hay una tabla de inversos para eliminar DIV's en la rutina de mezcla.

8. Cuidado con...

En primer lugar, mucho cuidado con la pila, cualquier desestabilizacion de la pila puede provocar resultados insospechados y hasta, en muchos casos, el cuelgue total del computador.

En segundo lugar, con los compiladores. Ignoro si hay algun compilador perfecto para el inline en asm, porque realmente yo no uso ninguno, siempre programo en 100% ensamblador, pero he visto casos como el de un chaval que una noche a través del IRC me comento que un bucle con MUL's y otro con FMUL's le funcionaban a la misma velocidad. Me paso el exe, y me di cuenta que el WATCOM C le habia generado en vez de FMUL's, FMULP's, con lo cual siempre popeaba datos y ocasionaba una excepción de pila, lo que hacia que el codigo de copro se ejecutara más lento. Las excepciones de pila por arriba, esto es por `st(0)` son controlables, pero por abajo (`st(7)`) no lo son tanto.

No sé si esto es un bug declarado del compilador, pero tambien he oido que existen algunos problemas con la conversion de real a entero de la citada Math387 del Watcom C, por lo cual aconsejo, que después de haber escrito la rutina, coger un debugger y asegurarse de que el codigo generado por el compilador es exacto.

Nosotros aqui en TLOTB trabajamos tanto con TASM como con NASM y con ninguno de los dos hemos tenido ningun problema con las instrucciones y sintaxis de copro, con lo cual tambien aconsejo el ensamblar las rutinas con alguno de estos dos ensambladores y despues enlazarlas con el programa principal.

9. ¿Que hay de verdad sobre las transferencias con copro?

Bien, esto es un tema peliagudo, y lo evaluaremos en dos casos particulares.

- **Memoria- >Memoria.**

En este caso se obtiene una ligera mejoria por tres razones. La primera es porque el procesador internamente esta orientado al trabajo con Qwords (si no, mirar la instruccion MOVQ del MMX).

La segunda es el conexionado del procesador a la memoria del sistema, cuya transmisión optima es 64bits (mejor si disponemos de modulos de memoria DIMM). La tercera, el alineamiento a 64bits es lo mas óptimo en una CPU Pentium. Aquí, en transferencia, hay que tener cuidado con los NaN.

- **Memoria- >Bus (SVGA,etc)**

En este caso depende mucho del chipset utilizado en la placa base. Despues de pruebas en varios equipos, se llega a la conclusion de que, aunque gastemos transferencia por copro, en el peor de los casos tarda lo mismo que con Dwords (osea un MOVSD) y en el mejor de los casos se gana en torno a un 5% o 10%. Esto oscila mucho dependiendo de la saturacion del BUS, de la SVGA y su velocidad para tragar datos,etc. Con lo cual, de todas formas es aconsejable su utilizacion, o al menos, probar que tal va! :)

10. Más Hints

Para borrar una zona de memoria (por ejemplo una pantalla virtual) el copro bate records.

Alineamiento... hacer la siguiente prueba...

- 1) Tomar una direccion alineada por ejemplo a Qword
- 2) esperar al retraso
- 3) colocar el color de borde a verde por ejemplo
- 4) volcar la pantalla.
- 5) colocar el color de borde a negro por ejemplo
- 6) goto al principio

Apuntar lo que tarda y luego hacer lo mismo con una direccion sin alinear.. ¡veréis la grandiosa diferencia! :)

11. Palabras de control y estado de coprocesador y registros de flag

Aqui esta la especificación de bits de las palabras de estado y de control, Hay que tener en cuenta que esta documentación la he podido conseguir sólo del 387, y que hay bastantes diferencias con el 287, asi que podría ser que en los actuales cambie alguna cosa.

PALABRA DE CONTROL

Bit	Nombre	Descripción
15-13	--	Reservados en el 387
12	--	Reservados en el 387; en el 287 era control de infinito.

11-10	RC	Rounding Control: <ul style="list-style-type: none"> • 00 = redondear al mas cercano; si es XXX.5 entonces ir al numero par más cercano. • 01 = Redondear siempre por abajo (2.1 - > 2; -2.1 - > -3) • 10 = Redondear siempre por arriba (2.1 - > 3; -2.1 - > -2) • 11 = Quitar decimales (2.1 - > 2; -2.1 - > -2)
9-8	PC	Precision Control: <ul style="list-style-type: none"> • 00 = 24 bits (single precision) • 01 = reservado en el 387 • 10 = 53 bits (double precision) • 11 = 64 bits (extended precision)
7-6	--	Reservados en el 387
5	PM	Máscara de excepción de Precisión
4	UM	Máscara de excepción de Underflow
3	OM	Máscara de excepción de Overflow
2	ZM	Máscara de excepción de Zero Divide
1	DM	Máscara de excepción de Denormalized Operand
9	IM	Máscara de excepción de Invalid Operation

PALABRA DE ESTADO

Bit	Nombre	Descripción
15	B	Flag de Busy
14	C3(Z)	Parte del Condition Code.
13-11	TOP	Puntero de pila (Top of Stack)
10	C2(C)	Parte del Condition Code.
9	C1(A)	Parte del Condition Code.
8	C0(S)	Parte del Condition Code.
7	ES	Error Summary flag
6	SF	Stack Flag
5	PE	Excepción de Precision
4	UE	Excepción de Underflow
3	OE	Excepción de Overflow
2	ZE	Excepción de Zero Divide
1	DE	Excepción de Denormalized Operand
0	IE	Excepción de Invalid Operation

Los Condition Codes son bastante complicados de interpretar, ya que su interpretación depende de la instrucción. Baste decir que tras la comparación (FCOM y familia) resulta lo

siguiente:

C3	C2	C0	Significado
0	0	0	TOP > Operand
0	0	1	TOP < Operand
1	0	0	TOP = Operand
1	1	1	Unordered

C1 es para la comparacion el flag de Zero, aunque no sé si eso significa que debe ser interpretado igual que el flag de Zero del registro Flags.

Si hay un Overflow o Underflow de pila, los bits IE y SF se pondrán a 1, y el flag C1 distinguira entre overflow (1) y underflow (0).

12. Denormals

(Esto no está en la documentacion de intel, y puedo equivocarme en algo).

La representación en coma flotante consiste en un bit de signo, un exponente y una mantisa, como sabéis. Para permitir exponentes negativos, en vez de almacenar el exponente en formato de complemento a dos, se almacena sumándole una cantidad llamada *Bias*, que es 127 para el Single, 1023 para el Double y 16383 para el Extended.

Los numeros Single y Double suponen un 1,... implícito antes de empezar la mantisa, pero en el Extended no es implícito, sino que lo necesita explícitamente.

Parece una tontería, pero en los Single y Double, ¿cómo se almacena un cero? Porque el siguiente caso, que se corresponde a todos los bits a cero (pongamos que de Single):

- Exponente = 0
- Mantisa = (1,)000000000000...
- Signo = 0

significaría $1,0000... * 2^{-127}$, pero no cero (aunque esté cerca).

La solución que se ha adoptado es considerar como caso especial el de Exponente = 0, en cuyo ese caso en vez de haber un 1 implícito en la mantisa hay un 0 implícito.

Así:

- Exponente = 0
- Mantisa = (0,)0001001110001000
- Signo = 0

se corresponde a $0,0001001110001000 * 2^{-127}$, o lo que es igual, $1,001110001000 * 2^{-131}$.

Ademas, de esa manera el cero puede ser representado, ya que $0,000000000000... * 2^{-127} = 0$.

A toda la familia de numeros con Exponente = 0, excepto el cero, se les llama *Denormals*. En el libro de Knuth (*Seminumerical Algorithms*) menciona la utilidad de los *Denormals* para no perder tanta precisión en ciertos cálculos.

El caso de los Extended no es diferente, salvo que el 1 explicito puede dejar de estar ahí cuando el exponente es cero (en el resto de numeros, el 1 era obligatorio que estuviera).

Ahora una nota del cuadernillo:

"FLD single/double cuando el operando es un Denormal convierte el numero a Extended precision y lanza la excepcion de Denormalized Operand. Al cargar un Signaling NaN, FLD single/double lanza una excepcion de Invalid Operand."

NaN significa Not a Number, aunque no sé qué formato tienen esos.

La tabla de excepciones dice lo siguiente:

Excepción	Causa	Acción por defecto (si la excepción está enmascarada)
Invalid Op.	Operación sobre un Signaling NaN, formato no soportado, operacion indeterminada ($0 \cdot \infty$, $0/0$, $\infty - \infty$, etc), o underflow/overflow de pila	El resultado es un Quiet NaN, un entero indefinido o un BCD indefinido.
Denormalized Op.	Al menos uno de los operandos es un Denormal, o sea, tiene el menor exponente pero una mantisa no-cero.	Continua el proceso normal.
Zero Divisor	El divisor es cero mientras el dividendo es un numero no cero y no infinito.	El resultado es Infinito.
Overflow	El resultado es demasiado grande para caber en el formato especificado.	El resultado es el mayor numero posible o infinito.
Underflow	El resultado es no-cero pero es demasiado pequeño para caber en el formato especificado, y si está enmascarada la excepción de Underflow, la denormalizacion causa pérdida de precisión.	El resultado es un denormal o cero.
Resultado Inexacto (Precisión)	El resultado EXACTO no es representable en el formato especificado (p.ej. $1/3$); el resultado es redondeado de acuerdo con el modo de redondeo.	Continua el proceso normal.

13. Para terminar.... de empezar.... :))

Lo aconsejable, es realizar al principio del programa un `fini` para inicializar el copro, despues descargar la palabra de control y enmascarar las excepciones y luego volver a cargarlas, con lo cual, reduciremos el tiempo de proceso de las instrucciones que puedan causar una excepcion.

Espero que esto os ayude en algo, la verdad para mi es suficiente para lo que es el manejo habitual de copro.

14. Agradecimientos y Saludos

Ummm.... a.....

- Cranky/TloTb.. por su apoyo linguistico, y por documentacion.
- Unreal/Pulse.. por incitarme a escribir esto.
- A los habituales en #demos,#coders y en es.comp.demos

Saludos especiales para Jcab/Iguana por la charla que tuvimos sobre los Denormals, los NaN y las transferencias de copro.

Todas las marcas registradas son propiedad de sus respectivos dueños, etc.. :)

Si alguien quiere una mejor revisión de este documento, o simplemente quiere consultarme algo o bien mandarme una amenaza para que deje de escribir...

Podeis dar conmigo en Astha@tlotb.com o a traves de <http://www.tlotb.com>.

XSLT en el IE5. (I)

URL: <http://www.programacion.net/articulos/xsltie5.php>

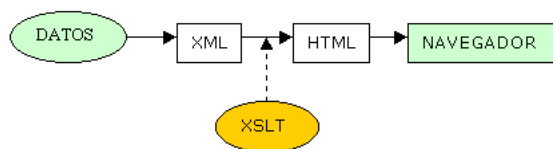
Autor: Joaquín Bravo Montero

Email: jbravo@retemail.es

NOTA: Todos los ejemplos de este artículo sólo se visualizan correctamente en el IE5.

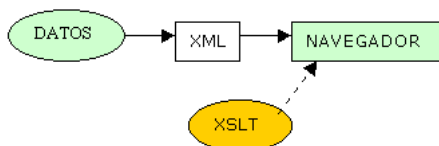
1. Presentación

En un [artículo](#) anterior estudiamos como mediante una XSLT podíamos convertir nuestros documentos XML en HTML y como de esta manera podíamos mostrar nuestros documentos XML en internet en formato HTML.



XML y XSLT

En este artículo vamos un poco más allá, y veremos como podemos saltarnos uno de los pasos anteriores y **enviar la información al navegador directamente en XML y que sea este mediante la XSLT que le indiquemos el que lo presente como si fuese un documento HTML**.



XML y XSLT en el navegador

Las **ventajas** de este método son evidentes:

- **Evitamos el paso de la conversión a HTML** ya que de esta nueva manera es el cliente (el navegador) el que presenta la información en HTML.
- **Ahorramos tamaño** en la información que tenemos que publicar. La primera vez enviaremos al navegador un fichero XML y un fichero XSLT. Pero posteriormente, cuando queramos pintar el fichero XML mediante la misma XSLT el navegador

utilizara la que ya tiene en la cache y por tanto sólo enviara el fichero XML y este siempre ocupa menos que el HTML

- Y además tendremos todas las ventajas que supone presentar nuestra información en XML en Internet en lugar de HTML.
 - Permitir búsquedas más inteligentes.
 - Facilita el intercambio de información con otras aplicaciones.
 - Facilita la personalización de la información.
 - etc.

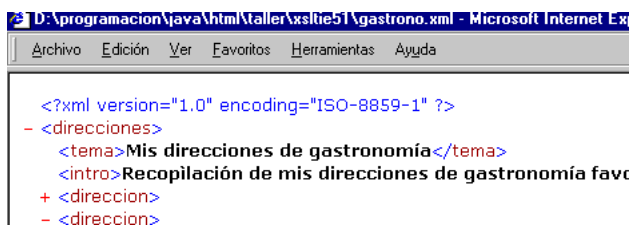
Aunque por el momento este método también tiene sus **inconvenientes**:

- De los navegadores más utilizados:
 - Sólo el IE5 es capaz de trabajar con XML y XSLT.
 - **Opera no lo soporta.**
 - Y el **Netscape tampoco**, aunque:
 - Existe un plugin desarrollado por la empresa [Inlogix](#) que permite visualizar XML sobre el Netscape mediante XSLT.
 - Y que dentro de Mozilla, se está desarrollando un procesador XSLT denominado [Transformiix](#), que permitiera trabajar con XSLTs en Netscape.
- La **implementación XSLT** que traen las diferentes distribuciones del **IE5 es antigua**. La primera versión del IE5 apareció en Febrero de 1999 mientras que la versión 1.0 de la especificación XSLT es del 16 de Noviembre de 1999. Pero este punto va pronto a dejar de ser un problema. Microsoft a lo largo del 2000 ha ido proporcionando nuevas versiones de la MSXML que cada vez se van aproximando más a la actual especificación XSLT.

2. El IE5 es un procesador XSLT

2.1. ¿Como se ve el XML en el IE5?

El IE5 como ya hemos dicho es capaz de mostrar XML. Hay que tener en cuenta que un fichero XML no son más que datos, y por tanto, si no le indicamos de manera alguna como tiene que mostrar esos datos los muestra tal cual como podemos observar por ejemplo en el documento [gastrono.xml](#):



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<direcciones>
<tema>Mis direcciones de gastronomía</tema>
<intro>Recopilación de mis direcciones de gastronomía favo
+ <direccion>
- <direccion>
```

Documento XML en el IE5

Aunque en este caso si nos fijamos, nos lo muestra:

- Formateados de forma más bonita.
- Y nos permite extender y recoger las ramas que forman el arbol del documento.

Esto es así, porque por defecto, le asocia una XSLT, la [defaultss.xsl](#) que lo pinta de esa manera.

Para evitar que nuestros documentos XML se pinten de esta manera y aparezca como a nosotros nos apetece tendremos que indicarle al documento XML con que XSLT queremos pintarlo.

2.2. Asociando una XSLT al XML

Para asociar una XSLT a nuestro XML hay que escribir en la cabecera del documento XML la XSLT que queremos utilizar de la siguiente manera:

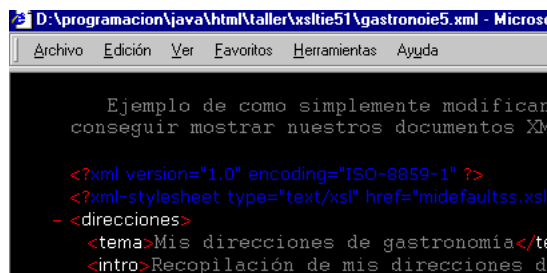
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml:stylesheet type="text/xsl" href="mi.xsl"?>
```

Es decir mediante la instrucción de proceso `xml:stylesheet`

Por ejemplo, podemos crear nuestra propia XSLT: [midefaultss.xsl](#) modificando la [defaultss.xsl](#), para que nuestro código XML se mostrase con un aspecto un diferente, y asociandola en la cabecera del documento XML de la siguiente manera:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml:stylesheet type="text/xsl" href="midefaultss.xsl"?>
```

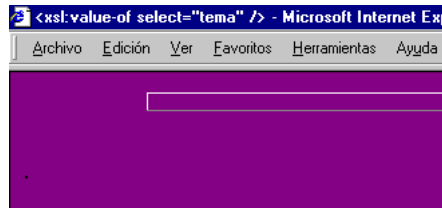
Obtendríamos este aspecto un poco más [psicodélico](#):



Documento XML en el IE5 con el aspecto que no es por defecto.

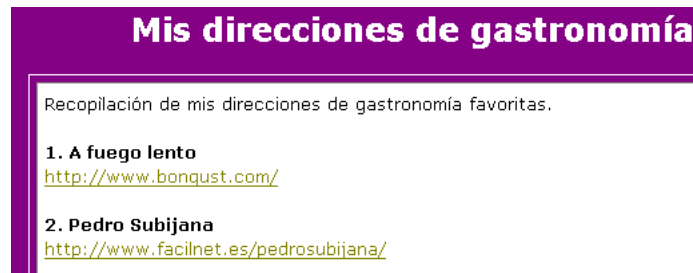
2.3. Estado de la implementacion XSLT del IE5

Como ya hemos dicho la primera versión del IE5 es bastante anterior a la versión 1.0 de la especificación XSLT, por lo que el procesador XSLT del IE5 es obsoleto. Es decir si intentamos utilizar las XSLT que utilizamos en el anterior [artículo](#) observaremos como los resultados son totalmente decepcionantes. Si por ejemplo utilizamos la [gastrono2.xsl](#):



XML en el IE5 con una XSLT estándar.

Cuando el aspecto tendría que haber sido:



Gastrono en formato HTML con una XSLT estándar.

Esto significa que en principio no podemos utilizar nuestras XSLT estándares con el IE5, y que por tanto si queremos utilizar XSLT y XML en el IE5 tendremos que aprender las peculiaridades de las XSLT para este navegador, aunque sinceramente creo que no vale la pena.

Pero no hay que desanimarse. Desde enero del 2000 Microsoft ha ido ofreciendo sucesivas actualizaciones de su MSXML que se han ido adaptando a la especificación definitiva. **Y en la actualidad podemos considerar que existe una nueva implementación de Microsoft que se adapta bastante al estándar.** Pero este sera precisamente el tema del próximo artículo.

3. Direcciones

- [1] Especificación XSLT, <http://www.w3.org/TR/xslt>.
- [2] Especificación XPath, <http://www.w3.org/TR/xpath>.
- [3] Artículo en castellano sobre como convertir XML en HTML utilizando XSLT, <http://html.programacion.net/taller/130699.htm>
- [4] Recopilación de direcciones con tutoriales, artículos, etc. sobre XSLT, XPath y XSL, <http://www.programacion.net/xsl.htm>
- [5] XML en la Web de Microsoft, <http://msdn.microsoft.com/xml/default.asp>.

XSLT, XSL, XPath: ¿qué es cada cosa?

URL: <http://www.programacion.net/articulos/xslt.php>

Autor: Joaquin Bravo Montero

Email: jbravo@retemail.es

1. ¿XSL o XSLT?

La verdad es que es una buena pregunta, ya que actualmente y debido sobretodo a la evolución que ha seguido la especificación **XSLT** se ha llegado a una situación en la que no está muy claro de qué estamos hablando cuando utilizamos el termino XSLT y que relación tienen con las especificaciones **XSL** (Extensible Style Language) y **XPath** (XML Path Language).

La XSL es una especificación desarrollada dentro del **W3c** para aplicar formato a los documentos XML de forma estandarizada. El primer boceto de esta especificación aparece el 18 de agosto de 1998 con el objetivo de implementar los mismos conceptos de las DSSSL, pero simplificados y pre-ajustados para ser utilizados en el web y definidos mediante una sintaxis XML.

Desde un principio, la XSL es un lenguaje para escribir hojas de estilo que consta de dos partes :

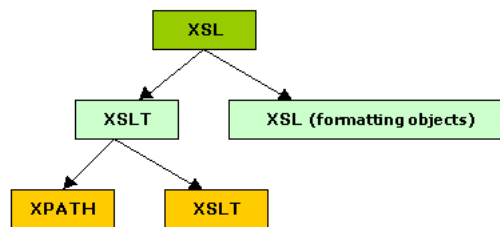
- **Un lenguaje de transformación**, mediante el cual se puede transformar un documento XML en otro XML.
- **Un lenguaje de formateo**, que no es más que un vocabulario XML para especificar objetos de formateo (FO).

Sin embargo, durante su desarrollo, **se decidió "sacar" fuera de la especificación XSL el lenguaje de transformación y desarrollarlo como una especificación "independiente" a la que se denominó XSLT**, centrándose, por tanto, desde entonces la especificación XSL en la definición del lenguaje de formateo. Ponemos las palabras "sacar" e "independiente" entre comillas porque no son del todo ciertas, ya que una especificación es inseparable de la otra debido a que la XSL hace uso de las XSLT, aunque la XSLT esta diseñada para ser utilizada independientemente de la XSL.

Ademas, durante el desarrollo de la especificación XSLT también **se decidió "sacar" fuera**

de ella la parte del lenguaje de transformación que hacía referencia a la forma de acceder y moverse por los nodos de un documento XML, debido esencialmente a que esta sintaxis era muy parecida a la que se estaba desarrollando en la especificación XPointer (enlaces al interior de un documento XML). A esta nueva especificación se le denominó XPath, y ha sido desarrollada no para utilizarse de forma independiente, sino desde las XSLT y desde los XPointer.

Gráficamente podríamos resumir la evolución de la especificación XSL de la siguiente manera:



Evolución de la especificación XSL

Pero, debemos precisar que el **gráfico anterior es bastante simple**. Desde la aparición del XML en 1998 se han ido desarrollando, o están en desarrollo nuevas especificaciones que en teoría enriquecen y añaden nuevas características a la especificación original: XSLT, Xpath, XInclude, XForm, Namespaces, etc.

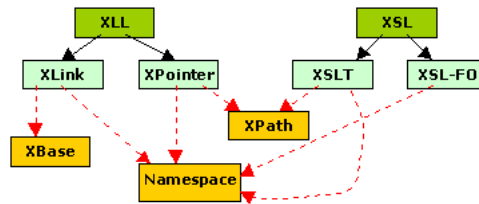
Nos encontramos en la primera fase del desarrollo de muchas de estas especificaciones y por tanto se trata de un momento crítico, ya que podemos llegar a un punto en el que la familia de estándares XML **puede convertirse en un conjunto de especificaciones recargadas e interconectadas a la fuerza**.

Dos factores fundamentales marcaran el futuro del desarrollo de estos estándares y serán la clave de su éxito:

- Su fragmentación.
- Su evolución siguiendo un model común.

Parece ser que la gente del W3c es consciente de este problema y están trabajando por este camino. **Sé estan evitando especificaciones recargadas y complicadas propiciando la fragmentación y la creación de estándares que sean reutilizables desde el resto de especificaciones.**

Por tanto un gráfico más preciso de la evolución de la especificación XSL y XLL (enlaces entre documentos XML) y sus relaciones con otras especificaciones sería el siguiente:



Evolución de la XSL y XLL

2. Resumiendo

Por tanto, y **resumiendo**:

- **XSL es la especificación** que desarrolla el lenguaje de formateo. Hace uso de la especificación XSLT.
- **XSLT es la especificación** que desarrolla el lenguaje de transformación. Hace uso de la especificación XPath. Ha sido diseñada para ser utilizada de forma independiente aunque es utilizada desde la especificación XSL.
- **XPath es la especificación** que desarrolla el lenguaje para acceder a los elementos de un documento XML. Ha sido desarrollada para ser utilizada desde la especificación XSLT y XPointer.

Actualmente la especificación **XSL** se encuentra en el estado de **borrador de trabajo** frente a las especificaciones **XSLT** y **XPath**, que son una **recomendación**, versión 1.0, desde el 16 de Noviembre de 1999.

3. Direcciones

- [1] Especificación XSLT, <http://www.w3.org/TR/xslt>.
- [2] Especificación XPath, <http://www.w3.org/TR/xpath>.
- [3] Borrador de trabajo XSL: <http://www.w3.org/TR/xsl/>.
- [4] XPointer (candidato a recomendación), <http://www.w3.org/TR/xptr>
- [5] Artículo en castellano sobre como convertir XML en HTML utilizando XSLT, <http://html.programacion.net/taller/130699.htm>
- [6] Recopilación de direcciones con tutoriales, artículos, etc. sobre XSLT, XPath y XSL, <http://www.programacion.net/xsl.htm>

Enviando passwords de forma segura con MD5 y PHP3

URL: <http://www.programacion.net/articulos/md5.php>

Autor: Alfredo Reino

Email: alf@ibium.com

Artículo cedido por [La página de Alfredo Reino](#).

1. El algoritmo MD5

El algoritmo MD5 es una funcion de cifrado tipo `hash` que acepta una cadena de texto como entrada, y devuelve un numero de 128 bits. Las ventajas de este tipo de algoritmos son la imposibilidad (computacional) de reconstruir la cadena original a partir del resultado, y tambien la imposibilidad de encontrar dos cadenas de texto que generen el mismo resultado.

Esto nos permite usar el algoritmo para transmitir contraseñas a traves de un medio inseguro. Simplemente se cifra la contraseña, y se envia de forma cifrada. En el punto de destino, para comprobar si el password es correcto, se cifra de la misma manera y se comparan las formas cifradas.

El ejemplo que vamos a ver ahora, se implementa un sistema de formulario de login y un script hecho en PHP3 que comprueba su validez.

Para ello, utilizamos la implementación de [MD5 en JavaScript](#) de [Paul Johnston](#).

2. El formulario de Login

El siguiente documento en HTML contiene un formulario en el que el usuario introduce su nombre de usuario y una contraseña. Cuando se pulsa el botón de enviar, se añade al password un numero aleatorio, y todo ello se cifra usando el algoritmo MD5.

A continuación, el numero aleatorio, y el resultado del MD5, se introducen en elementos tipo "Hidden" en el formulario, y se envia todo a un `script` escrito en PHP3 que verifica el acceso.


```
<html><head><title></title>
<script language="JavaScript" src="md5.js"></script>
<script language="JavaScript">
numero = Math.random().toString();

function calculaMD5() {
var pw = document.forms["login"].elements["password"].value
pw += numero
return calcMD5(pw)
}

function enviaMD5(hash) {
document.forms["login"].elements["cifrado"].value = hash;
document.forms["login"].elements["numero"].value = numero;
document.forms["login"].submit();
}
</script>
</head>

<body>
<form action="auth.php3" method="POST" name="login">
Usuario: <input type="Text" name="usuario"><br>
Password: <input type="Password" name="password"><br>
<input type="Hidden" name="cifrado" value="">
<input type="Hidden" name="numero" value="">
<input type="Submit" value=" Login " onClick="enviaMD5(calculaMD5())">
</form>
</body></html>
```

3. El script en PHP3

PHP3 es un lenguaje de `server-side scripting`, como pueda serlo el ASP, ColdFusion, etc. En este ejemplo se ha usado PHP3 porque ya lleva una implementación del algoritmo MD5.

Además, PHP3 es gratuito y open source. Para más información, lo mejor es visitar <http://www.php.net/>

Igualmente se podría haber utilizado ASP, utilizando la misma implementación del MD5 que vimos antes, y escribiendo el `script` en JScript.

Básicamente, el `script` extrae el password para el usuario correspondiente (la función `passwordUsuario()`, no está detallada en este texto). A continuación, le añade el número aleatorio que recibe del formulario, aplica el algoritmo MD5, y lo compara con el valor del password cifrado que recibe del formulario.

```
<html><head><title></title></head>
<body>

<?
$password = passwordUsuario($usuario);

$serverpassword = strtolower(md5($password.$numero));
$clientpassword = strtolower($cifrado);

if ($serverpassword==$clientpassword)
{ print "<p>Correcto!"; }
else
{ print "<p>Acceso denegado!"; }
?>

</body></html>
```

Montar una web con apache+php+postgresql en Linux

URL: <http://www.programacion.net/articulos/apache.php>

Autor: Rafael Martinez

Email: r.m.guerrero@medisin.uio.no

En este articulo vamos a tratar la instalación y configuración de un servidor web, utilizando **Apache** como servidor, **PHP** como lenguaje interpretado de alto nivel y **PostgreSQL** como base de datos. Con esta combinacion podremos crear páginas dinamicas y obtener informacion de nuestra base de datos para presentarla via web. Por ultimo, daremos unos cuantos ejemplos de como programar nuestras páginas web utilizando estos programas.

Partimos de la base de que tenemos una máquina con **Linux** instalado y funcionando y con todas las herramientas necesarias para la compilación de programas (**gcc,make,..**). Este artículo se basará en **Apache 1.3.x**, **PHP 3.0.x** y **PostgreSQL 6.5.x** y ha sido comprobado en un equipo con **Apache 1.3.6**, **PHP 3.0.12** y **PostgreSQL 6.5.1**. Lo primero que tenemos que hacer es bajarnos los tres paquetes con los programas necesarios y grabarlos en un directorio de nuestro sistema (por ejemplo, `/local/download/`):

- **Apache:**
 - [Principal](#)
 - [Mirrors](#)
- **PHP:**
 - [Principal](#)
- **PostgreSQL:**
 - [Principal](#)
 - [Mirrors](#)

Una vez que tenemos los programas, tendremos que elegir el lugar donde los vamos a instalar. En este artículo suponemos que instalaremos:

Apache en: `/usr/local/apache/`

PHP como módulo de Apache

PostgreSQL en: /usr/local/pgsql/

Catalogo Web: /home/httpd/html/

1. Instalación de PostgreSQL

Lo primero que tenemos que hacer es crear una cuenta que administrará la base de datos:

```
[localhost]$ su
[localhost]$ /usr/sbin/adduser postgres
[localhost]$ passwd postgres
[localhost]$ exit
```

Una vez creada la cuenta Postgres crearemos los directorios que utilizaremos para instalar PostgreSQL con los permisos adecuados:

```
[localhost]$ su
[localhost]$ cd /usr/src
[localhost]$ mkdir pgsql
[localhost]$ chown postgres:postgres pgsql
[localhost]$ cd /usr/local
[localhost]$ mkdir pgsql
[localhost]$ chown postgres:postgres pgsql
[localhost]$ exit
```

Empezamos con el proceso de compilación/instalación:

```
[localhost]$ su postgres
[localhost]$ cd /usr/src/pgsql
[localhost]$ gunzip -c /local/download/postgresql-6.5.x.tar.gz
| tar xvf -

[localhost]$ cd /usr/src/pgsql/postgresql-6.5.x/src
[localhost]$ ./configure --prefix=/usr/local/pgsql
--with-tcl --with-perl

[localhost]$ gmake all > make.log 2>&1 &
[localhost]$ tail -f make.log

[localhost]$ gmake install > make.install.log 2>&1 &
[localhost]$ tail -f make.install.log
[localhost]$ exit
```

Ahora tenemos que decirle al sistema donde poder encontrar las librerías necesarias, para ello actualizamos el fichero /etc/ld.so.conf:

```
[localhost]$ su
[localhost]$ echo /usr/local/pgsql/lib >> /etc/ld.so.conf
[localhost]$ /sbin/ldconfig.
[localhost]$ exit
```

También tendremos que actualizar el fichero ~/.bash_profile de la cuenta administradora de la base de datos, en este caso **Postgres** (si utilizais otro **shell** que no sea **bash**, tendreis que

cambiar el archivo correspondiente, en vez de `.bash_profile`):

```
[localhost]$ su postgres

*****
Editar el archivo ~/.bash_profile y
añadirle lo siguiente
*****

PATH=$PATH:/usr/local/pgsql/bin
MANPATH=$MANPATH:/usr/local/pgsql/man
PGLIB=/usr/local/pgsql/lib
PGDATA=/usr/local/pgsql/data
export PATH MANPATH PGLIB PGDATA

*****
Salir para que los cambios surtan efecto
*****

[localhost]$ exit
```

Una vez que hemos terminado de instalar la base de datos y configurar nuestro sistema, tenemos que inicializarla y arrancarla:

```
[localhost]$ su postgres
[localhost]$ initdb
[localhost]$ cd
[localhost]$ nohup postmaster -i > pgserver.log 2>&1 &
[localhost]$ exit
```

Ya tenemos nuestra base de datos **PostgreSQL**, instalada y funcionando. Ahora solo tenemos que administrarla, para ello nada mejor que leerse los [manuales de documentación](#) de la misma y aprender SQL. Solamente nos queda hacer un par de ajustes en la configuración para que podamos acceder a **postgresql** via PHP/web. Lo primero es incluir en el archivo `/usr/local/pgsql/data/pg_hba.conf` la siguiente línea:

```
host all tu_maquina_IP tu_maquina_NETMASK trust
```

Y la segunda es dar privilegios de acceso en tu base de datos/tablas al usuario para que pueda coger los datos de la misma (**Nobody** es el usuario que ejecuta el servidor **Apache** por defecto). Para ello puedes hacer lo siguiente:

```
*****
Suponemos que tenemos una base de datos
llamada prueba, con una tabla llamada
direcciones
*****

[localhost]$ su postgres
[localhost]$ psql prueba
prueba=> GRANT SELECT ON direcciones
prueba=> TO nobody;

prueba=> \z
prueba=> \q
[localhost]$ exit
```

2. Instalación de Apache y PHP como módulo mismo

Pasamos a la segunda parte de este artículo, para ello procederemos como sigue:

```
[localhost]$ su
[localhost]$ cd /usr/src
[localhost]$ gunzip -c /local/download/apache_1.3.x.tar.gz
| tar xvf -
[localhost]$ gunzip -c /local/download/php-3.0.x.tar.gz
| tar xvf -
[localhost]$ cd apache_1.3.x
[localhost]$ ./configure --prefix=/usr/local/apache
[localhost]$ cd ../php-3.0.x
[localhost]$ ./configure --with-pgsql=/usr/local/pgsql
--with-apache=../apache_1.3.x --enable-track-vars
--enable-sysvsem --enable-sysvshm
--enable-url-includes
[localhost]$ make
[localhost]$ make install
[localhost]$ cd ../apache_1.3.x
[localhost]$ ./configure --prefix=/usr/local/apache
--activate-module=src/modules/php3/libphp3.a
[localhost]$ make
[localhost]$ make install
[localhost]$ cd ../php-3.0.x
[localhost]$ cp php3.ini-dist /usr/local/lib/php3.ini
[localhost]$ exit
```

Ya tenemos apache instalado y **PHP** como módulo del mismo. Ahora tenemos que hacer unos cuantos ajustes en la configuración para que todo funcione. Tenemos que editar el fichero `/usr/local/apache/conf/httpd.conf` y añadirle lo siguiente:

```
AddType application/x-httpd-php3 .php
DirectoryIndex index.html index.php
```

Estas dos líneas son las únicas necesarias para que **Apache** sepa que hacer con un fichero que contenga código **PHP**. Existen otras opciones que deberías actualizar en vuestro fichero `/usr/local/apache/conf/httpd.conf` para terminar de configurar **Apache**, por ejemplo: **ServerAdmin**, **ServerName**, **DocumentRoot**, directivas "**Directory**", etc. Los comentarios incluidos en este fichero son autoexplicativos y no deberíais tener ningún problema para ajustar la configuración a vuestro sistema. Ahora sólo nos queda arrancar el servidor :

```
[localhost]$ su
[localhost]$ /usr/local/apache/bin/httpd
-f /usr/local/apache/conf/httpd.conf
[localhost]$ exit
```

NOTA: Para obtener toda la información/documentación completa pasaos por [Apache documentación](http://httpd.apache.org/) y [PHP documentación](http://php.net/).

3. Ejemplos

3. Ejemplos

Lo primero que tenemos que hacer es comprobar que **PHP** funciona bien. Para ello podemos crear un fichero `index.php` en nuestro catalogo web `/home/httpd/html/` con las siguientes lineas:

```
<HTML>
<HEAD>
  <TITLE>Pagina index de prueba</TITLE>
</HEAD>
<BODY>

<?php

/* Codigo php de esta pagina */

echo "Esto es una prueba<BR>
     Dia/hora: ".date("d/m/Y - H:i:s")."<BR>";
?>

</BODY>
</HTML>
```

Este fichero debería de daros como resultado dos líneas en pantalla, una de ellas con el día y la hora de vuestro servidor. Una vez comprobado que **PHP** funciona, vamos a crear una página web, que acceda mediante **PHP** a **PostgreSQL** y que nos devuelva como resultado el contenido de una de las tablas de la base de datos. Suponemos que ya tenemos una base de datos llamada `prueba`, con una tabla `direcciones` que contiene tres campos `calle`, `ciudad`, `pais`. La máquina que estamos utilizando es `servidor.domain.es` y **PostgreSQL** utiliza el puerto **5432**(puerto por defecto).

```
<HTML>
<HEAD>
  <TITLE>Pagina index de prueba</TITLE>
</HEAD>
<BODY>

<?php

/* ***** */
/* Conexion a PostgreSQL */
/* ***** */

/* Conexion a la base de datos */
$conexion = pg_pconnect("host=servidor.domain.es
                        port=5432 dbname=prueba");

if (!$conexion) {
    echo "<CENTER>
         Problemas de conexion con la base de datos.
        </CENTER>";
    exit;
}

$sql="SELECT * FROM direcciones ORDER BY pais;";

/* Ejecuta y almacena el resultado de la orden
   SQL en $resultado_set */
$resultado_set = pg_Exec ($conexion, $sql);
$filas = pg_NumRows($resultado_set);

/* Presenta la informacion almacenada en $resultado_set */
for ($j=0; $j < $filas; $j++) {

echo "Direccion: ".pg_result($resultado_set, $j, 0)." <BR>
     Ciudad: ".pg_result($resultado_set, $j, 1)." <BR>
     Pais: ".pg_result($resultado_set, $j, 2)." <P>";

}
}
```

```
/* Cierra la conexion con la base de datos */  
pg_close($conexion);  
  
?>  
  
</BODY>  
</HTML>
```

Esta página web nos debería de presentar la información contenida en la tabla `direcciones` de la base de datos `prueba`. Y a partir de aquí solamente teneis que leer la documentación y usar vuestra imaginación para crear páginas web dinámicas, actualizadas y que presenten la información contenida en vuestras bases de datos.

Acceso a base de datos de Access 2000 usando el ADO Data Control

URL: <http://www.programacion.net/articulos/ado2000.php>

Autor: Martín Suárez Viacava

Email: webmaster@visualb6.com

Si has intentado usar una base de datos creada o convertida con el Access 2000, te habrás llevado una pequeña sorpresa al ver que usando el Data Control que se incluye en el Visual Basic no te permite hacerlo. Esto es porque el nuevo formato de Access usa el motor Jet versión 4.0 y los datacontrol DAO sólo "entienden" hasta la versión 3.6...

Pero si tienes la versión 6.0 de Visual Basic, o bien has conseguido las DLLs de ADO (ActiveX Data Object), puedes usarlas sin problemas... incluso con el ADO Datacontrol.

En el siguiente código veremos cómo abrir y movernos en una base de datos, en este caso, lo mismo dará que la base de datos haya sido creada con Access 97 o con el Access 2000.

1. Para empezar

Crea un nuevo proyecto, en Proyecto/Componentes... selecciona Microsoft ADO Data Control (OLEDB), pulsa Aceptar y verás que se añade un nuevo control a la barra de herramientas: ¡Ese será el control que usaremos!

Haz dobleclick en ese control para que se añada al formulario, cámbiale la altura a un valor pequeño: 315 está bien, sitúalo donde más coraje te de y añade un TextBox, cambia el tamaño, yo le he dado estos valores:

- Alto 315, ancho 2955.

Crea un array del Text1 recién añadido:

- Selecciona el Text1, pulsa el botón derecho y copialo; pulsa en cualquier lado del Form y pulsa el ratón derecho del ratón, selecciona Pegar... te preguntará si quieres crear un array del control Text1, responde que SI.
- Vuelve a pegar de nuevo y se creará un tercer Text1.

- Ahora tendrás tres controles Text1 con los índices desde 0 a 2

Si te parece demasiado "básico" todos estos pasos... te aguantas... y espera un poco que ya mismo termino.

2. Ahora vamos a configurar el ADO Data Control

Selecciona el datacontrol, en la ventana de propiedades pulsa en Custom...

Te mostrará un cuadro de diálogo, (puede que si tienes la versión en castellano del VB, no te muestre lo que yo te digo, ya que la versión que tengo está en inglés, pero espero que no te lies...)

- En la ficha General, estará seleccionada la opción "Use Connection String", pulsa en el botón "Build..."
- Te mostrará otro cuadro de diálogo, en la ficha "Provider", selecciona Microsoft Jet 4.0 OLE DB Provider, pulsa en "Next>>" y selecciona la base de datos que quieres usar, (si tienes la que están incluidas en el ZIP, se llamará db2000.mdb); el resto de opciones déjalos como está; pulsa en "Aceptar" dos veces para que se cierren los cuadros de diálogo.

Ahora tenemos que **decirle que tabla usar** y otras cosillas.

- Selecciona la propiedad CursorType y de la lista desplegable selecciona 2-adOpenDynaset.
- Selecciona la propiedad RecordSource y pulsa en el botón, te mostrará un cuadro de diálogo.
- De la lista desplegable (Command type), selecciona: 2-adCmdTable, la lista "Table or Stored Procedure Name" se habrá habilitado, selecciona el nombre de la tabla que quieres usar, en este caso Table1 y pulsa en Aceptar.

Esto mismo se puede hacer mediante código, para ello asígnale estos valores al datacontrol: (por ejemplo en el evento Form_Load)

```
' Indicar la base de datos a usar
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=" & App.Path & "\db2000.mdb"
Adodc1.CursorType = adOpenDynamic

' Conectarlo a la tabla de prueba
Adodc1.RecordSource = "Table1"

' Refresh es necesario para que se cargue la tabla
Adodc1.Refresh
```

Nota: En el código de ejemplo asignaremos en tiempo de ejecución la base de datos y demás conexiones necesarias.

Ahora será necesario "ligar" los cuadros de texto con el Datacontrol y los campos correspondientes:

- Selecciona los tres Text1, en la ventana de propiedades selecciona DataSource y en la

lista desplegable selecciona Adodc1

3. Para ligar cada Text1 con un campo de la base de datos

Pulsa en cualquier parte del formulario para quitar la selección actual.

Selecciona el Text1 con índice 0.

En la ventana de propiedades, selecciona DataField y de la lista desplegable, selecciona "Nombre".

Haz lo mismo con los otros dos Text1, pero para el de índice 1, selecciona "e-mail" y para el otro: "Comentario".

Este último control sería conveniente que lo hicieras más grande y MultiLine, ya que se supone que aceptará textos más grandes porque el tipo de campo es "Memo".

Ya puedes pulsar en F5 para probar que todo esto funciona.

Realmente no es necesario añadir nada de código, pero si quieres hacerlo manualmente, añade lo siguiente en el evento Form_Load:

```
Private Sub Form_Load()  
    ' Indicar la base de datos a usar  
    Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
        "Data Source=" & App.Path & "\db2000.mdb"  
    Adodc1.CursorType = adOpenDynamic  
  
    ' Conectarlo a la tabla de prueba  
    Adodc1.RecordSource = "Table1"  
  
    ' Refresh es necesario para que se cargue la tabla  
    Adodc1.Refresh  
  
    ' Conectar manualmente los Text1 al recordset  
    Dim i As Long  
  
    ' Asignar el control data  
    For i = 0 To 2  
        Set Text1(i).DataSource = Adodc1  
    Next  
  
    ' Asignar los nombres de los campos  
    Text1(0).DataField = "Nombre"  
    Text1(1).DataField = "e-mail"  
    Text1(2).DataField = "Comentario"  
End Sub
```

Para que el Caption del data control nos muestre el número de registro, en este caso el campo ID de la tabla, añade este código:

```
Private Sub Adodc1_MoveComplete(ByVal adReason As ADODB.EventReasonEnum, _  
    ByVal pError As ADODB.Error, _  
    adStatus As ADODB.EventStatusEnum, _  
    ByVal pRecordset As ADODB.Recordset)  
    ' Mostrar el ID del registro actual  
    ' si se pasa del primero o del último, dará error  
    On Local Error Resume Next  
  
    ' Mostrar el ID del registro actual usando el recordset pasado como  
    ' parámetro  
    'Adodc1.Caption = "ID del Registro: " & pRecordset!ID
```

```
' También se puede usar:  
Adodc1.Caption = "ID del Registro: " & Adodc1.Recordset!ID  
  
Err = 0  
End Sub
```

Esta página web nos debería de presentar la información contenida en la tabla `direcciones` de la base de datos `prueba`. Y a partir de aquí solamente teneis que leer la documentación y usar vuestra imaginación para crear páginas web dinámicas, actualizadas y que presenten la información contenida en vuestras bases de datos.

4. Añadir y borrar registros de la tabla

Para tener estas dos opciones, añade dos botones al formulario, al primero lo llamas `cmdAdd` y en el Caption escribe `Añadir`, al otro botón le cambias el nombre a `cmdDel` y en el Caption escribe: `Eliminar`.

Añade el siguiente código:

```
Private Sub cmdAdd_Click()  
    Adodc1.Recordset.AddNew  
End Sub  
  
Private Sub cmdDel_Click()  
    Adodc1.Recordset.Delete  
End Sub
```

Email: elrincon@jet.es

1. Introducción

2. Solución

Podemos olvidarnos de utilizar la función `GetFullPathName`. A pesar de su nombre, no se obtiene un nombre de archivo en formato largo a partir de uno corto. Su cometido es el de generar un nombre completo de archivo a partir de la unidad y directorio actuales. Si le pasamos como primer parámetro la ruta completa de un archivo en formato corto, nos devolverá lo mismo en su tercer parámetro como se muestra en el ejemplo 1.

Generado por <http://www.programacion.net>

GetFullPathName generará una ruta a partir de la unidad y directorio en el que nos encontremos actualmente como se muestra en el ejemplo 2.

```
lstrcpy(lpszShortPath, "MSPAIN.T.EXE");

// cambiamos de directorio
SetCurrentDirectory("C:\\WINDOWS\\ESCRIT~1");
GetFullPathName(lpszShortPath, _MAX_PATH, lpszLongPath, &filepart);
// lpszLongPath contiene "C:\\WINDOWS\\ESCRIT~1\\MSPAIN.T.EXE"

// cambiamos de nuevo de directorio
SetCurrentDirectory("C:\\WINDOWS");
GetFullPathName(lpszShortPath, _MAX_PATH, lpszLongPath, &filepart);
// ahora lpszLongPath contiene "C:\\WINDOWS\\MSPAIN.T.EXE"
```

Hay sin embargo una manera de obtener el nombre de un archivo en formato largo y es utilizando la función del shell SHGetFileInfo. Para ello hay que especificar SHGFI_DISPLAYNAME en su quinto parámetro. Pero hay un pequeño problema, sólo convierte la última parte de la ruta completa, es decir, en el caso de tener por ejemplo la ruta **C:\\WINDOWS\\ESCRIT~1\\MIPRIM~1.EXE** únicamente podríamos obtener **MiPrimerPrograma.exe** correspondiente a **MIPRIM~1.EXE** como se muestra en el ejemplo 3.

```
SHFILEINFO sfi;

lstrcpy(lpszShortPath, "C:\\WINDOWS\\ESCRIT~1\\MIPRIM~1.EXE");
SHGetFileInfo(lpszShortPath, 0, &sfi, sizeof(sfi), SHGFI_DISPLAYNAME);
strcat(lpszLongPath, sfi.szDisplayName);
// lpszLongPath contiene "MiPrimerPrograma.exe"
```

Debido a esto es necesario llamar a la función SHGetFileInfo para cada subdirectorio especificado en la ruta del nombre completo empezando desde el primer nivel de subdirectorio. Seguidamente se muestra la función definitiva:

```
DWORD GetLongPathName2
(LPCSTR lpszShortPath, LPSTR lpszLongPath, DWORD cchBuffer)
{
    DWORD ret;
    char shortPath[_MAX_PATH];
    char longPath[_MAX_PATH];
    char tempPath[_MAX_PATH];
    char* backslash = "\\"; // carácter donde debemos detenernos cada vez
    char* token;
    SHFILEINFO sfi;

    ret = 0;

    // comprobamos que se trata de un nombre de archivo o ruta existente
    if (GetFileAttributes(lpszShortPath) != -1)
    {
        lstrcpy(shortPath, lpszShortPath);
        token = strtok(shortPath, backslash);
        ZeroMemory(tempPath, sizeof(tempPath));

        strcat(tempPath, token); // ej. "C:"
        strcat(tempPath, backslash); // ej. "C:\"

        // empezamos a generar el nombre en formato largo
        lstrcpy(longPath, token);

        // leemos el primer directorio o nombre de archivo
        token = strtok(NULL, backslash);

        while (token != NULL)
        {
            // añadimos el siguiente subdirectorio o nombre de archivo
```

```
    strcat(tempPath, token);

    // añadimos la barra "\" al final,
    // no importa si se añade al final de un nombre de archivo
    strcat(tempPath, backslash);

    // obtenemos el nombre en formato largo en el
    // campo sfi.szDisplayName
    SHGetFileInfo(tempPath, 0, &sfi, sizeof(sfi), SHGFI_DISPLAYNAME);

    // vamos añadiendo lo último convertido
    // a la cadena en formato largo
    strcat(longPath, backslash);
    strcat(longPath, sfi.szDisplayName);

    // leemos el siguiente directorio o nombre de archivo,
    // si ya se han leído todos, devolverá un puntero nulo
    token = strtok(NULL, backslash);
}

// obtenemos el número de caracteres que forman el nombre completo
ret = strlen(longPath);

if (cchBuffer >= ret)
{
    // copiamos el nombre obtenido en el primer parámetro de la función
    lstrcpy(lpszLongPath, longPath);
}

return ret;
}
```

Iconos en los favoritos del Explorer 5

URL: http://html.programacion.net/taller/tw_favicon.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Cuando trabajas con el Explorer 5 (y el 5.5) y añades un favorito, en ocasiones puede que te haya sorprendido el hecho de que éste se incorpore a la lista con un icono propio, en lugar de utilizar el estándar. En este artículo veremos lo sencillo que resulta hacerlo.

1. El icono

El Explorer 5 dispone de una nueva característica que permite crear un pequeño icono (16x16) y colocarlo en tu web de modo que cuando alguien te añada a sus favoritos, dicho icono acompañe en la lista al nombre de tu página.

Realmente lo más complicado de todo este asunto es crear el icono. Porque al Explorer no le gustan los formatos estándar en Internet como JPG, GIF o PNG para hacer esto y prefiere que se utilice el formato ICO, el que utiliza Microsoft en Windows para representar los iconos de los distintos programas.

Como prácticamente ninguno de los programas profesionales de diseño gráfico permiten grabar en este formato, lo mejor puede ser que nos hagamos con un programa específicamente diseñado para ello. Hay muchos, algunos de ellos disponibles en [Tucows](#), de los que recomiendo el [Microangelo](#), por recomendar alguno. Este programa permite tanto crear el icono en él como importar un gráfico que esté en un formato más normal y pasarlo a formato ICO.

Para que el asunto funcione debéis recordar hacer el icono del tamaño ya indicado de 16x16 y de llamar al archivo `favicon.ico`.

2. Poniendo el icono en tu web

Cuando se añade un favorito, el Explorer 5.0 automáticamente mira en la raíz de nuestro dominio por el archivo denominado `favicon.ico`. Es decir, si tenemos un dominio llamado, digamos, <http://html.programacion.net>, cuando un usuario añade a sus favoritos cualquier

página dentro del mismo, mirará a ver si existe el fichero

`http://html.programacion.net/favicon.ico`.

Así pues, si disponemos de dominio propio, deberemos colocar ese fichero en el directorio raíz del mismo.

2.1. Si no tienes dominio propio

Lo malo es que lo más habitual resulta ser que nuestras direcciones sean algo así como `http://www.geocities.com/~robertoalcazarypedrin` y no podamos introducir el icono en el directorio raíz. En ese caso disponemos de una opción extra, que consiste en añadir el siguiente código en la cabecera de nuestras páginas:

```
<LINK REL="SHORTCUT ICON" HREF="/~robertoalcazarypedrin/favicon.ico">
```

De este modo, toda página que disponga de este código utilizará el icono indicado en la lista de favoritos. Recordad que sólo funciona en Explorer 5 y superiores.

Paso de variables Javascript entre páginas (I)

URL: http://html.programacion.net/taller/tw_pasovar1.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Muchos habéis preguntado por la manera de pasar variables Javascript entre distintas páginas. Hay más de un modo de hacerlo, y aquí explico el que me parece más sencillo. Aquí tenéis un ejemplo.

1. El método

Vamos a utilizar el mismo método que se usa para pasar variables a scripts CGI, PHP o ASP. Si os fijas en cualquier buscador, cuando realizáis una búsqueda, la barra de direcciones se os llena de un montón de caracteres un poco extraños, llenos de "?" y "+" por todos lados. Bueno, pues nosotros vamos a utilizar la misma técnica.

Codificaremos nuestras variables en la URL de la dirección destino. Es lo mismo que hace un formulario cuando utilizais el método GET. Podeis ver un [ejemplo](#) de cómo funciona.

Aunque, como veremos, es relativamente sencillo pasar estos valores de una página a otra, deberemos tener en cuenta que sólo podemos pasar variables que tengan valores, como puedans er números o cadenas, pero nunca referencias a objetos, como son los vectores.

2. La página origen

Para generar la dirección destino debemos saber cómo se codifican una lista de variables para introducirlas en una URL. Esto se hace poniendo un símbolo de interrogación (?) detrás del nombre del archivo al que se pasan las variables (que en nuestro caso será `destino.html`). Después de ese símbolo viene la lista de variables, separada por el símbolo ampersand (&). Cada elemento de dicha lista consiste en el nombre de la variable, el símbolo de igualdad (=) y el valor de la variable.

Sin embargo, y como en una URL podemos incluir muy pocos caracteres, algunos de los valores de las variables tendrán que estar codificados. Por ejemplo, no puede haber espacios en una dirección en formato URL. ¿Cómo incluimos entonces los espacios de nuestras

variables tipo cadena? Sustituyendolos por %20, que es el equivalente al espacio dentro de la codificación URL. Así por ejemplo, si tenemos dos variables llamadas `var1` y `var2` cuyos valores son, respectivamente, 3 y "Hola, que tal", la URL resultante será:

```
destino.html?var1=3&var2=Hola%20que%20tal
```

Gracias a Dios, Javascript ofrece una pareja de funciones que permiten codificar y decodificar una cadena a formato URL. Son `escape` y `unescape`. Por lo tanto, ya tenemos todo lo que nos hace falta para enviar variables. Para que sea más sencillo utilizaremos una función que, al ser llamada, realice el envío. Esta función se llamará `pasarVariables` y recibirá dos parámetros de tipo cadena. El primero contendrá el nombre de la página a la que deseamos ir y el segundo una lista separada por comas de las variables que deseamos enviar:

```
function pasarVariables(pagina, nombres) {
    pagina += "?";
    nomVec = nombres.split(",");
    for (i=0; i<nomVec.length; i++)
        pagina += nomVec[i] + "=" + escape(eval(nomVec[i]))+"&";
    pagina = pagina.substring(0,pagina.length-1);
    location.href=pagina;
}
```

El método funciona añadiendo a la variable `pagina` el texto necesario para pasar las variables y, finalmente, redirigirse a la dirección resultado de esos añadidos. Por eso, lo primero es añadir el símbolo de interrogación. Luego crea un vector con los nombres de las variables utilizando el método `split`. El bucle añade a cada iteración el nombre de la variable, el símbolo de igualdad, el valor de dicha variable codificado en URL y el ampersand. Por último, una vez terminado el bucle, se borra el último ampersand que ya nos sobra.

Por fin, para utilizar esta función, la llamaremos desde un enlace:

```
<a href="javascript:pasarVariables('destino.html', 'var1,var2')">Pulse
aquí</a>
```

3. La página destino

Una vez que sabemos enviar las variables, tendremos que escribir también un poco de código para recibirlas desde la página destino. Este es el código que colocaremos en la cabecera de la página, antes que el resto del código, para que podamos utilizar las variables cuanto antes:

```
cadVariables = location.search.substring(1,location.search.length);
arrVariables = cadVariables.split("&");
for (i=0; i<arrVariables.length; i++) {
    arrVariableActual = arrVariables[i].split("=");
    if (isNaN(parseFloat(arrVariableActual[1])))
        eval(arrVariableActual[0]+"="+unescape(arrVariableActual[1])+"");
    else
        eval(arrVariableActual[0]+"="+arrVariableActual[1]+"");
}
```

Lo que hacemos en la primera línea es eliminar la interrogación del texto que viene detrás del nombre de la página en la URL. Para lograrlo, escogemos una subcadena de `location.search`, que es la propiedad que contiene ese texto.

Luego, convertimos esa cadena en un vector que tiene una pareja "variable=valor" en cada uno de sus elementos. Entonces, recorremos cada elemento de dicho vector, creamos un vector temporal que contiene en su primer elemento el nombre de la variable y en el segundo el valor y, utilizando la función `eval`, creamos la variable asignandola a su valor.

Por último, indicar que dentro del bucle existe una condición que comprueba si la variable es o no numérica para, al hacer la asignación, decidir si encierra o no el valor entre comillas.

Un ratón con estela

URL: http://html.programacion.net/taller/tw_estela.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Ultimamente parece haberse puesto de moda en muchas páginas web el hecho de que el ratón vaya dejando una estela. Te enseñamos a hacerlo de modo que funcione en todos los navegadores de versión mayor a la 4.

1. ¿Qué necesitamos?

Lo primero que vamos a necesitar para poder realizar esta técnica, son los gráficos que compondrán la estela. En el código en el que basamos este ejemplo, vamos a tener sólo uno llamado `circulo.gif`.



`circulo.gif`

De todos modos, el código permite que tanto el número de gráficos que componen la estela, como los nombres de los mismos sean fácilmente sustituibles. Lo normal sería tener un gráfico por cada elemento que compone la estela, pero como nosotros lo vamos a hacer con círculos, solamente utilizaremos uno cuyo tamaño cambiaremos adecuadamente.

Por otro lado, en el código utilizaremos el objeto `DetectorNavegador` que ya utilizamos en [artículos anteriores](#).

2. Objeto Estela

Cada uno de los gráficos que componen la estela será un objeto Javascript distinto al que daremos el nombre de `Estela`. Para inicializarlo utilizaremos el siguiente constructor:

```
// Objeto Estela
function Estela(capaID, anchura, altura, nombreImg) {
  this.x = 0;
  this.y = 0;
  this.mover = MoverEstela;
  this.seguirRaton = EstelaSeguirRaton;
  document.write(soporta.NS4 ?
    "<layer id='" + capaID + "' left=" + this.x + " top=" + this.y +
    " width=" + anchura + " height=" + altura + ">" :
    "<div id='" + capaID + "' " + " style='position:absolute;left:" +
```

```

        this.x + "; top:" + this.y + "; width:" + anchura + "; height:" +
        altura + ">");
    document.write("<img src='" + nombreImg + "' width=" + anchura +
    " height=" + altura + " border=0>");
    document.write(!soporta.NS4 ? "</div>" : "</layer>");
    this.capaRef = (soporta.DOM) ? document.getElementById(capaID) :
    (soporta.IE4) ? document.all[capaID] : document.layers[capaID];
    this.estiloRef = (soporta.NS4) ? this.capaRef : this.capaRef.style;
}

```

El constructor toma cuatro parámetros: el nombre que tendrá la capa (puede ser cualquiera, siempre que no se repita), la altura y anchura de la imagen y el nombre del fichero que contiene la misma. Lo que hace, primero, es construir las referencias adecuadas para los métodos y propiedades del objeto. Dispondrá de dos propiedades (*x* e *y*), que contendrán la posición de la capa, y dos métodos (*mover* y *seguirRaton*) que permitirán que la capa se mueva, ya sea a unas coordenadas concretas o las que en ese momento posea el ratón.

Lo siguiente es escribir el código HTML correspondiente a la capa. Esta capa contendrá el gráfico (y sólo eso), tendrá su misma altura y anchura y el nombre que le hayamos pasado como parámetro. Estará colocada en las coordenadas *x=0* e *y=0*, que corresponden a la esquina superior izquierda. Hay que tener en cuenta que la estela estará en estas coordenadas sólo el tiempo que tardemos en mover el ratón, por lo que no resultan muy importantes.

Finalmente, crea un par de referencias: a la capa y al objeto que contiene los atributos de la misma. Esto es necesario porque cada navegador alberga dichas referencias con nombres distintos.

Vamos a ver ahora que hacen los métodos:

```

function MoverEstela(x,y){
    this.estiloRef.left = x;
    this.x = x;
    this.estiloRef.top = y;
    this.y = y;
}

```

Este método simplemente mueve la capa a las coordenadas indicadas, actualizando a la vez las propiedades *x* e *y*.

```

function EstelaSeguirRaton(e){
    this.mover(
        soporta.IE4 ?
            event.clientX + (soporta.DOM ? document.body.scrollLeft : 0) :
            e.pageX+2,
        soporta.IE4 ?
            event.clientY + (soporta.DOM ? document.body.scrollTop : 0) :
            e.pageY+2);
}

```

Este resulta más complicado. También mueve la capa, pero lo hace a las coordenadas a las que actualmente indique el puntero del ratón. Para averiguar éstas, necesitamos de un objeto de tipo *Event* al que llamamos *e*. Este objeto sólo está disponible en el momento en que sucede un evento, por lo que no podremos averiguar las coordenadas del ratón hasta que se produzca uno.

Tanto Explorer como Netscape disponen de propiedades que permiten acceder a la posición

del puntero del ratón, pero tienen nombres distintos. De hecho incluso pueden tener valores distintos dependiendo de si estamos utilizando Explorer 4 o 5. Pero bueno, en el código tenéis la "solución" universal.

2.1. Inicialización

Una vez escrito el código del objeto, necesitaremos inicializar diversas instancias del mismo; una por cada gráfico que forme parte de la estela:

```
var soporta = new DetectorNavegador();
var numImg = 6;
var ritmo = 50;
var estelas = new Array(numImg);
estelas[0] = new Estela("est1", 30, 30, "graficos/circulo.gif");
estelas[1] = new Estela("est2", 25, 25, "graficos/circulo.gif");
estelas[2] = new Estela("est3", 20, 20, "graficos/circulo.gif");
estelas[3] = new Estela("est4", 15, 15, "graficos/circulo.gif");
estelas[4] = new Estela("est5", 10, 10, "graficos/circulo.gif");
estelas[5] = new Estela("est6", 5, 5, "graficos/circulo.gif");
```

En la variable `numImg` indicaremos el número de gráficos que componen la estela. Mejor dicho, el número de capas que contienen gráficos y que forman parte de la estela. Conviene aclararlo porque, en nuestro caso, estrictamente hablando, sólo disponemos de un gráfico. La variable `ritmo` indica la velocidad a la que la estela seguirá al ratón. Modificándola se puede variar sustancialmente el efecto. Por último, el vector `estelas` contiene los objetos `Estela` necesarios. Podéis ver cómo el nombre del gráfico no cambia en cada uno de ellos, pero sí su tamaño.

3. Funciones auxiliares

Para que todo el tinglado funcione, necesitaremos un par de funciones auxiliares:

```
function seguirRaton(e){
    estelas[numImg-1].seguirRaton(e);
}
```

Esta función es un controlador de evento que será llamado cuando el ratón se mueva. En tal caso le comunicará al último de los objetos `Estela` que componen el vector `estelas` (en nuestro ejemplo, el círculo más pequeñito) que se coloque en las mismas coordenadas que el puntero del ratón. Pero eso sólo mueve uno, así que necesitamos otra función para mover a los demás:

```
function ciclo(){
    for (i=0;i<(numImg-1);i++)
        estelas[i].mover(estelas[i+1].x,estelas[i+1].y);
    setTimeout("ciclo()", ritmo);
}
```

Esta función va recorriendo el vector de estelas del primer al penúltimo elemento, indicándole a cada uno que debe tomar la posición que tenía en el anterior ciclo el círculo inmediatamente anterior. Esta función es la que se encarga, por tanto, del movimiento de todos los demás círculos.

También se encarga de llamarse a sí misma con una espera de "`ritmo`" milisegundos. Si no lo hicieramos así no se produciría esa especie de inercia que se puede observar cuando se ejecuta esta técnica, pues todos los círculos seguirían al ratón a tal velocidad que no nos enteraríamos de nada.

4. Inicialización

Por último, debemos realizar un par de tareas de inicialización si queremos que todo este tinglado empiece a funcionar:

```
if (soporta.DHTML) {  
    if(soporta.NS4)  
        document.captureEvents(Event.MOUSEMOVE);  
    document.onmousemove=seguirRaton;  
    setTimeout("ciclo()", ritmo);  
}
```

Básicamente, hacemos dos cosas: capturamos el evento de movimiento de ratón y le decimos que en caso de que ocurra llame a nuestra función `seguirRaton` y le decimos que ejecute `ciclo` por primera vez dentro de un ratito.

Y eso es todo, si quereis podeis ver el código fuente de esta página y copiar el código Javascript literalmente en las vuestras y vereis que funciona sin problemas.

Barra de menús desplegables (III): En marcos distintos

URL: http://html.programacion.net/taller/tw_menus3.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Parece que muchos de nuestros lectores desean tener, en un marco, las opciones principales y, en otro, los menús desplegados. Os mostramos cómo hacerlo, de una manera robusta.

1. ¿Qué cambia?

Al contrario que el artículo anterior, este artículo no es una mejora que todos los usuarios de los menús desplegables deban utilizar. Tan sólo es recomendable para aquellos que quieran disponer de este tipo de menús en páginas con varios marcos. Desde luego, es recomendable la lectura de los anteriores artículos de esta serie para comprender éste.

El problema es el siguiente. Tenemos las opciones principales (es decir, la barra de menús) en un marco, y los menús deben desplegarse en otro distinto... Además, resulta conveniente no tener que estar escribiendo el código HTML correspondiente a las opciones en cada uno de los documentos en que se desplegarán, sino hacerlo sólo una vez en el marco que aloja las barras de menús. De este modo es más sencillo de mantener y más difícil cometer errores.

2. Escribir el código HTML de los menús

Si disponéis de Explorer 4 o Netscape 4 y superiores, podéis ver como nuestro [ejemplo](#). El código no varía mucho, pero hay que hacer cambios. Entre otras cosas, ahora parte del código está en una página y otra parte en otra. Debemos tener cuidado de que ambos marcos estén ya totalmente cargados y cosas así.

Prácticamente todo el código estará en el marco que contiene la barra de menús desplegables, ya que será éste el que no cambie mientras navegamos por nuestra página. Sin embargo, necesitará un par de "ayuditas" del otro marco, que procuraremos sean lo más pequeñas posible.

Lo primero que debemos hacer es encontrar la manera de escribir el código HTML de los

menús desde el marco que contiene la barra pero no los menús desplegados. A este marco lo llamaremos `opciones` siendo el otro `principal`. Esto lo conseguiremos, primero, añadiendo la siguiente función:

```
function escribirMenus() {
    top.principal.document.write('<DIV id="menu0" CLASS="menu">' +
    '<A HREF="../../recursos/img.htm">Imágenes</A><BR>' +
    ...
    '<A HREF="../../131098.htm">No quedar encerrado en los marcos</A><BR>' +
    '</div>');
}
```

Sin embargo, esta función deberá ser llamada desde el otro marco. Esto es debido a la manera en que funciona `document.write`, tema que ya está tratado en el [curso de Javascript](#), por lo que no abundaremos en exceso en ello. Lo que pasa es que si queremos escribir código HTML desde Javascript sin borrar el código que ya tiene la página, deberemos hacerlo **antes** de que se termine de leer la página. Es decir, llamando a `document.write` antes de que aparezca en el código fuente de la página la etiqueta `</BODY>`. Por tanto, colocaremos este código justo antes de esa etiqueta:

```
...
<SCRIPT LANGUAGE="Javascript1.2">
<!--
if (top.opciones.escribirMenus)
    top.opciones.escribirMenus();
else
    history.go(0);
//-->
</SCRIPT>
</BODY>
</HTML>
```

Lo primero que hacemos es comprobar si la función que hemos creado un poco más arriba ya existe. Si es así, la ejecuta, escribiendo el código HTML de nuestros amores. Si no es así, vuelve a cargarse la página. Esto puede parecer un desperdicio y, de hecho, lo es, pero es necesario ya que no tenemos manera de "obligar" al navegador a que cargue el otro marco antes que éste. De todos modos, no tendrá que recargarse muy a menudo, teniendo en cuenta que la función `escribirMenus()` la colocaremos al principio del fichero que contenga el otro marco y esta comprobación está al final del nuestro.

Por último, debemos incluir en el marco `principal` la hoja de estilos correspondiente a nuestros menús. Ya que lo que deseamos es poner el menor código repetido posible en esa página, lo mejor es guardar en un archivo externo dicha hoja de estilo:

```
.menu {
    position: absolute;
    visibility: hidden;
    background-color: white;
    layer-background-color: white;
    color: black;
    border-style: solid;
    border-color: black;
    border-width: 1px;
    padding: 3px;
    font-size: 12px;
    font-family: "arial", "helvetica";
}

.menu A: hover {text-decoration: underline; color: blue;}
.menu A {text-decoration: none; color: black;}
```

Y, por supuesto, llamarla desde nuestra página:

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

3. Cómo adaptar el código Javascript

Lo que hacemos es copiar y pegar el código que hicimos en el artículo anterior y colocarlo en la página que contiene la barra de menús. Luego, procederemos a modificarlo para que "comprenda" que los menús están ahora en otro marco. Lo primero que vamos a hacer es crearnos una referencia al marco donde se despliegan los menús para ahorrarnos el escribirlo muchas veces y no tener que hacer muchos cambios si algún día le cambiamos el nombre:

```
var wref = parent.principal;
```

Lo siguiente será adaptar todas las referencias dentro del objeto menú para que sean referencias a objetos situados en el otro marco. Esto se logra cambiando una sola línea del código del constructor:

```
this.capaRefStr = (soporta.NS4) ?  
  'wref.document["'+capaID+'"]' :  
  ((soporta.IE4) ? 'wref.document.all["'+capaID+'"]' : 'this.domRef');
```

Por último, o casi, nos queda hacer una pequeña labor de inicialización del marco `principal`. Dado que los menús funcionan desapareciendo cuando pulsamos en cualquier lugar de nuestras páginas, debemos extender esa comprobación a todos los marcos que las componen. Quedaría un poco feo que pulsáramos en el marco donde se despliegan las opciones y no pasara nada. Por lo tanto, deberemos añadir el siguiente código en la cabecera de cada página situada en dicho marco:

```
<SCRIPT language="javascript">  
var wref = parent.opciones;  
  
// Inicializacion  
function inicializar() {  
  if (wref.soporta.DHTML) {  
    if (wref.soporta.NS4)  
      document.captureEvents(Event.MOUSEUP);  
    document.onmouseup = wref.ocultarMenuActivo;  
  }  
}  
  
window.onload = inicializar;  
</SCRIPT>
```

Como vemos, es parte del código que hemos utilizado hasta ahora para inicializar, pero adaptado al hecho de que todo nuestro código está en un marco distinto, por lo que tenemos que colocar la referencia `wref` delante de cada llamada a funciones situadas en el marco `opciones`.

3.1. Controlar el scroll

Entre nuestros diversos problemas de adaptación, ahora tenemos uno nuevo, ¿qué pasa si el usuario intenta abrir un menú y la página situada en el marco `principal` está desplazada por

medio de una barra de scroll? Pues que, como está colocada en un lugar fijo, seguramente se vea sólo parte. Queda muy feo. Jo. Así que tendremos que buscarnos la vida para colocarlo en el sitio adecuado. Para ello tendremos que utilizar tres soluciones distintas.

La más sencilla de las tres nos viene dada por un estándar: el CSS2. Desafortunadamente, la característica que vamos a utilizar sólo está disponible en el Netscape 6PR1, así que tendremos que utilizar otra cosa tanto para Explorer como para Netscape 4. Consiste en posicionar cada uno de los menús no en forma absoluta (*absolute*) sino de forma fija (*fixed*). Esto lo haremos en el constructor de los menús:

```
if (soporta.DOM) {  
    ...  
    if (!soporta.IE4)  
        this.domRef.style.position = "fixed";  
}
```

Tenemos que colocar esta segunda condición, porque el Explorer 5 soporta el DOM pero no esta característica del CSS2. En este navegador y el Netscape 4, tendremos que utilizar dos propiedades que nos indican cuantos pixels está desplazada la página de la esquina superior izquierda. En caso del Netscape esas propiedades son `pageXOffset` y `pageYOffset`. En el del Explorer `document.body.scrollLeft` y `document.body.scrollTop`. Así, de nuevo en el constructor, creamos cuatro nuevas propiedades en el menú.

```
this.topOffsetStr = (soporta.NS4) ? 'wref.pageYOffset' :  
    (soporta.IE4 ? 'wref.document.body.scrollTop' : '0');  
this.leftOffsetStr = (soporta.NS4) ? 'wref.pageXOffset' :  
    (soporta.IE4 ? 'wref.document.body.scrollLeft' : '0');  
this.top = top;  
this.left = left;
```

Las dos primeras cadenas contienen el código necesario para acceder a esas propiedades en Netscape y Explorer y cero si es el Netscape 6. Ahora modificamos la función que coloca el menú:

```
function cambiarPosicionMenu(top, left) {  
    eval(this.capaRefStr + this.estiloRefStr + this.topRefStr +  
        ' = this.top + ' + this.topOffsetStr);  
    eval(this.capaRefStr + this.estiloRefStr + this.leftRefStr +  
        ' = this.left + ' + this.leftOffsetStr);  
}
```

Y ahora, debemos llamar a esta función desde los sitios adecuados. El primer sitio donde lo haremos será la función `mostrarMenu()`, de modo que así nos aseguramos que, cada vez que mostremos un menú, lo hagamos en el sitio adecuado. Ahora, como mejora adicional, intentaremos que los menús se coloquen bien cada vez que se muevan las barras de scroll. Para ello creamos la siguiente función:

```
function moverMenuActivo(e) {  
    if (menuActivo)  
        menuActivo.cambiarPosicion();  
}
```

Y la llamamos desde la función de inicialización:

```
function inicializar() {  
    if (wref.soporta.DHTML) {
```

```

    if (wref.soporta.NS4)
        document.captureEvents(Event.MOUSEUP | Event.MOUSEMOVE);
    document.onmouseup = wref.ocultarMenuActivo;
    document.onmousemove = wref.moverMenuActivo;
}

```

3.2. Sincronización

Lo que hemos hecho funcionaría a la perfección si viviéramos en un Internet de velocidad brutal donde acceder a una página fuera lo mismo que leerla del disco duro. Pero ya que no es así tenemos que lidiar con un problema adicional: la sincronización. Esto significa, en la práctica, que el código que maneja los menús debe saber de antemano que los dos marcos se han terminado de cargar.

Para lograrlo, lo primero que debemos decidir es cual de los dos marcos se encargara de este trabajo. Sólo puede haber una respuesta. Uno de los marcos se queda fijo, el de la barra de menús, y el otro varía, que es el principal. Lo más cómodo será que éste último controle lo que pasa en el fijo, que normalmente estará ya cargado y no nos causará mayores problemas. Lo haremos de la siguiente manera:

```

function inicializar() {
    if (!wref.inicializar) {
        setTimeout("inicializar()",10);
        return;
    }
    if (wref.soporta.DHTML) {
        if (wref.soporta.NS4)
            document.captureEvents(Event.MOUSEUP | Event.MOUSEMOVE);
        document.onmouseup = wref.ocultarMenuActivo;
        document.onmousemove = wref.moverMenuActivo;
        wref.inicializar();
    }
}

window.onload = inicializar;

```

La función de inicialización del marco principal se ejecuta cuando éste termina de cargar. Si el código fuente del otro marco (colocaremos la función `inicializar()` la última) no está aún cargado, esperamos a que lo esté. Y cuando eso sucede inicializamos también el marco de opciones con la llamada a `wref.inicializar()`.

Pero existe un último problema. Y de verdad que éste sí que es el último. En los intervalos en los que se pulsa un enlace y el marco que contiene la barra de menús está cargado e inicializado pero en el otro todavía no se ha cargado ninguna página, tendremos un error de Javascript si intentamos desplegar un menú. Para evitarlo colocamos la siguiente comprobación:

```

function mostrarMenu() {
    if (!eval(this.capaRefStr)) return;
    ...
}

function ocultarMenu() {
    if (!eval(this.capaRefStr)) return;
    ...
}

```

Comprobamos si existe el menú y, si no es así, no hacemos nada. Y esto es todo, amigos.

Protección con contraseña (II)

URL: http://html.programacion.net/taller/tw_password2.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Os mostramos otra manera, algo más flexible que la anterior, de proteger contra intrusos vuestras páginas.

1. Elegir contraseña y página

La mayor crítica que tenía el modo anteriormente propuesto era la obligatoriedad de que la contraseña y el nombre de la página coincidieran. En este caso, lo que haremos será codificar tanto la contraseña como el nombre de la página de una manera un tanto extraña.

Teclead una contraseña y un nombre de página cualquiera teniendo en cuenta las siguientes restricciones:

- Sólo pueden contener letras, no números ni otros caracteres.
- La página acabará en .html
- Si el nombre de la página excede en longitud en más de un carácter a la contraseña, esos caracteres de más no estarán codificados.

En el código de la página tendremos dos variables que contendrán la codificación de contraseña y página. Es difícil decodificarlos a mano sin tener la contraseña pero, ojo, no es imposible. De hecho, es probable que un buen hacker lo considere sencillo. Esta protección nos salvaguardará de la mayoría de los usuarios, pero no de todos.

2. Página de introducción de la contraseña

Suponiendo que nuestra página tenga como contraseña `password` y como página `bienvenido`, éste sería el código a utilizar:

```
<HTML>
<HEAD>
  <TITLE>Introduce la contraseña</TITLE>
  <SCRIPT LANGUAGE="Javascript">
    <!-- Esconde el código a navegadores antiguos
    var alfabeto= "ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHI"
```

```

var cod1 = "42691";
var cod2 = "COHOWITQHO";
function decodificar(formulario) {
    passcod = codificarC(formulario.password.value,3);
    if (passcod == cod1) {
        aux = "" + codificarC(formulario.password.value,10);
        pagina = decodificarP(cod2, aux);
        location.href = pagina + ".html";
    }
    else
        alert("La contraseña es incorrecta");
}
function codificarP(pagina,num) {
    var codigo=""
    for (var i=0;i<pagina.length;i++) {
        letra=pagina.substring(i,i+1).toUpperCase();
        a=alfabeto.indexOf(letra,0);
        a+=(num.substring(i,i+1)*1);
        codigo += alfabeto.substring(a,a+1);
    }
    return codigo;
}
function decodificarP(pagina,num) {
    var result="";
    for (i=0;i<pagina.length;i++) {
        letra=pagina.substring(i,i+1).toUpperCase();
        a=alfabeto.indexOf(letra,0);
        a-=(num.substring(i,i+1)*1);
        if (a<0) a+=26;
        result += alfabeto.substring(a,a+1).toLowerCase();
    }
    return result;
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="decodificador" onSubmit="decodificar(this); return false;">
    Contraseña: <INPUT TYPE="password" NAME="password">
    <INPUT TYPE="submit" VALUE="Entrar">
</FORM>
</BODY>
</HTML>

```

Es complicado y no voy a dar más que algunas pistas sobre cómo funciona, ya que cuanto más se entienda este script, más probable es que alguien cree un "rompedor" de contraseñas para él. Básicamente, lo que hace es codificar la contraseña que le indiquemos y comprobar si coincide con la contraseña codificada que tiene en la variable `cod1`. En caso afirmativo, ya sabe que la contraseña es correcta y pasa a decodificar la página.

La codificación de la página (contenida en la variable `cod2`) es un conjunto de letras de igual longitud que el nombre real de la página. Cada letra se separa de la letra original por un número que depende de una codificación algo distinta de la contraseña. Es debido a la naturaleza de dicha codificación a lo que se debe la restricción de las longitudes a la que antes hacía referencia.

Podéis probar el ejemplo que acabamos de describir

Por supuesto, este ejemplo siempre se puede mejorar, con cosas como soporte a varios usuarios. Pero eso es otra historia, y debe ser contada en otra ocasión (si os interesa saberlo, claro está).

Barra de menús desplegables (II): Soporte W3C-DOM

URL: http://html.programacion.net/taller/tw_menus2.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Actualización del código de las barras de menús desplegables para poder funcionar en navegadores que soporten el W3C DOM (Netscape 6 y Explorer 5).

1. ¿Qué es el W3C DOM?

Vistas las numerosas incompatibilidades surgidas entre las visiones del HTML dinámico de Netscape y Explorer, el consorcio W3 decidió crear un modelo de objetos del documento (DOM) que permitiera que el DHTML fuera compatible entre todos los navegadores.

Parece que dicho objetivo esta en vías de verse cumplido. Con la reciente aparición de la primera versión preliminar del Netscape 6 se ve confirmado que los dos navegadores mayoritarios van a soportar DOM. Netscape, incluso, ha eliminado el soporte de su antiguo modelo de HTML dinámico, por lo que podemos estar seguros que la mayoría de las páginas DHTML se actualizarán a DOM a lo largo de este año. Así pues, ¿por qué no hacer nosotros lo mismo?

2. Prestos a modificar nuestros menús

Si disponéis de Explorer 5 o Netscape 6, podéis ver como nuestro [ejemplo](#) funciona en dichos navegadores. Si miráis el código puede que os sorprenda lo cortito que es. Vamos a estudiar a partir de ahora qué modificaciones han sido necesarias para adaptarlo. Para seguir este artículo es más que recomendable que hayáis leído y entendido la [entrega anterior](#) de nuestra saga de menús desplegables.

Como usuarios la única novedad es que deberemos indicar la anchura de los menús al inicializarlos. Será el cuarto parámetro del constructor:

```
function inicializar() {  
    ...  
    menú[0] = new Menu("menu0", 20, 5, 100);  
}
```



```
menu[1] = new Menu("menu1", 20, 93, 120);  
menu[2] = new Menu("menu2", 20, 250, 310);  
}
```

Sin embargo, esta necesidad se debe principalmente a un bug del Netscape 6 PR1, que no parece soportar la anchura automática. Debido a ello es posible que eliminemos esta restricción en el futuro.

3. Cómo funcionan los menús en DOM

Lo primero que debemos hacer es adaptar nuestro detector de navegadores para que soporte los nuevos navegadores:

```
function DetectorNavegador() {  
    this.NS4 = document.layers;  
    this.IE4 = document.all;  
    this.DOM = document.getElementById;  
    this.DHTML = this.NS4 || this.IE4 || this.DOM;  
}  
  
var soporta = new DetectorNavegador();
```

La única novedad es la inclusión de la propiedad `this.DOM`, que indica si el navegador soporta este estándar. Lo averiguamos preguntando por la existencia del método `document.getElementById`, propia del DOM.

El grueso de nuestras modificaciones estarán, por tanto, en el objeto `Menu`. Comenzaremos viendo el nuevo constructor:

```
function Menu(capaID, top, left) {  
    this.activar = activarMenu;  
    this.mostrar = mostrarMenu;  
    this.ocultar = ocultarMenu;  
    this.cambiarPosicion = cambiarPosicionMenu;  
    if (soporta.DOM) {  
        this.domRef = document.getElementById(capaID);  
        this.domRef.style.width = width;  
        this.domRef.style.display = "none";  
        this.cambiarPosicion(top, left);  
    }  
}
```

La primera novedad es que la obtención de una referencia, que llamamos `domRef`, al nodo correspondiente a nuestro menú. DOM modeliza toda nuestra página HTML como un gigantesco árbol, siendo cada etiqueta un nodo del mismo. Por medio del método `document.getElementById` obtenemos el nodo cuyo atributo `id` sea idéntico al dado como parámetro.

Una vez obtenida la referencia al nodo correspondiente a la etiqueta `DIV` que contiene al menú, modificamos su anchura por medio de la línea `this.domRef.style.width = width;`. ¡Y funciona en ambos navegadores! Siglos hacía que una cosa así se podía hacer igual tanto en Explorer como en Netscape. Dejad que disfrute del momento y ahora continuo... Ya.

Hay que indicar que por medio del atributo `style` se pueden modificar todas las propiedades de las que dispone una etiqueta por medio de las hojas de estilo. En la siguiente línea modificamos la propiedad `display`, que es nueva y fue definida en el estándar CSS2.

Debemos indicar que no queremos que el elemento se visualice de ninguna manera. Si no lo hacemos así seguiríamos sin ver el menú, puesto que dijimos en la hoja de estilos que estuviera oculto (`visibility: hidden`), pero podríamos pulsar sobre los enlaces, aún cuando éstos no se vieran.

Finalmente llamamos al método que cambia de posición el menú y que ahora tiene este aspecto:

```
function cambiarPosicionMenu(top, left) {  
    this.domRef.style.top = top;  
    this.domRef.style.left = left;  
}
```

Como véis, el sistema es muy parecido a lo realizado anteriormente. Los métodos para mostrar y ocultar un menú son, a su vez, muy similares:

```
function mostrarMenu() {  
    this.domRef.style.visibility = "visible";  
    this.domRef.style.display = "block";  
}  
  
function ocultarMenu() {  
    this.domRef.style.visibility = "hidden";  
    this.domRef.style.display = "none";  
}
```

La mayor diferencia con respecto a la anterior versión es que tendremos que modificar tanto `visibility` como `display`.

3.1. Compatibilidad hacia atrás

Por supuesto, dado que podemos hacer menús tanto para DOM como para los navegadores de versión 4, es posible realizar una versión que funcione en todos ellos. No vamos a entrar en cómo realizarla, ya que principalmente no es más que un montón de `if` por todos lados, pero os dejo [un nuevo ejemplo](#) de cómo se hace.

Forzar un salto de página en la impresión

URL: http://html.programacion.net/taller/tw_forzar_salto.php

Autor: Alex Morales

Email: alexmm@iname.com

En la línea de los artículos relacionados con la impresión, en este taller te presentamos el procedimiento a utilizar para forzar un salto de página al imprimir una página Web desde el navegador.

1. Introducción

Si has tratado nunca de imprimir una página Web a menudo habrás sido víctima de saltos de página en secciones inesperadas y poco prácticas. Esto puede evitarse aplicando un pequeño recurso de DHTML (y más concretamente hojas de estilo) para Internet Explorer y mediante un truquillo muy sencillo en el caso de Netscape.

En ambos casos deberás disponer de versiones 4.0 o superior. No se pretende juzgar que navegador implementa mejor esta característica (en otros sitios web encontrarás extensa documentación a favor y en contra de ambos), simplemente selecciona el código que necesitas (en el mejor de los casos aplica ambos trucos) o pruebalo tu mismo imprimiendo esta [página de ejemplo](#).

2. Para Internet Explorer (versión 4.0 o superior)

Debes crear una hoja de estilos y definir el tag H1 del como sigue: (recuerda que debes insertar el tag `style` dentro del tag `head` del documento).

```
<STYLE>
  H1.SaltoDePagina
  {
    PAGE-BREAK-AFTER: always
  }
</STYLE>
```

En el sitio en que quieras forzar el salto de página deberás poner el tag `h1` aplicando el estilo `SaltoDePagina` definido anteriormente.

```
<H1 class=SaltoDePagina> </H1>
```

3. Para Netscape Navigator (versión 4 o superior)

Netscape no procesa el estilo PAGE-BREAK-AFTER con lo cual deberemos aplicar un truco simple pero efectivo para conseguir nuestro propósito.

Inserta todos aquellos contenidos que desees se incluyan en una página dentro de una **tabla** ya que cuando una tabla no puede imprimirse por completo en una página Netscape fuerza un salto de página. Por ejemplo:

```
<table>
<tr><td>
  Introduzca aquí los contenidos correspondientes a la página 1.
</td></tr>
</table>
<table>
<tr><td>
  Introduzca aquí los contenidos correspondientes a la página 2.
</td></tr>
</table>
```

Impedir el uso del botón derecho del ratón

URL: http://html.programacion.net/taller/tw_boton_derecho.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

En Netscape 4 y Explorer 4 podemos impedir que aparezca el menú cuando el usuario pulsa el botón derecho sobre una imagen y, por tanto, impedir que puedan bajarse nuestras imágenes.

1. Descripción y limitaciones

Lo que haremos será interceptar el evento `onMouseDown`, que ocurre cuando el usuario pulsa un botón del ratón. Como lo que nos interesa es proteger las imágenes, procuraremos interceptarlo sólo cuando pulse sobre las mismas. Una vez interceptado, deberemos comprobar que el botón pulsado sea el derecho, e impedir en caso afirmativo que aparezca el menú. En esta página, sin ir más lejos, está funcionando este código. Intenta bajarte una de las imágenes de arriba y verás.

Desafortunadamente, toda protección realizada con Javascript es susceptible de ser eliminada simplemente configurando el navegador para que no interprete Javascript. Nada es perfecto...

2. Lo primero es interceptar `onMouseDown`

Dado que lo que queremos es evitar que el usuario pulse sobre una imagen, lo mejor será interceptar ese evento en las mismas. Y eso es suficiente en Explorer. Sin embargo, y dado que muchas imágenes sirven también como enlaces, en Netscape seguirá apareciendo el menú cuando se de ese caso, por lo que deberemos interceptar también ese evento en los enlaces. Podríamos hacerlo así:

```
<IMG SRC="..." onMouseDown="nuestroControladorDeEventos()">
```

Pero, claro, hacer eso con todas nuestras imágenes puede resultar, como decirlo, aburrido. Así que lo mejor será hacer un `script` que llame al controlador de evento cuando suceda `onMouseDown` en cualquiera de las imágenes y enlaces del documento:

```
for (var i=0; i<document.images.length; i++)  
    document.images[i].onmousedown=noBotonDerecho;
```

```
if (document.layers)
  for (var i=0; i<document.links.length; i++)
    document.links[i].onmousedown=noBotonDerecho;
```

Supongo que los más avisados se habrán dado cuenta de que nuestro controlador de eventos se va a llamar `noBotonDerecho`. Hemos de indicar que éste código sólo funciona en los navegadores de cuarta generación (y siguientes) ya que fueron los primeros en soportar el evento que estamos utilizando, `onMouseDown`. Por eso, para comprobar rápidamente que utilizamos Netscape, preguntamos por la existencia del vector `document.layers`, que sólo aparece en dicho navegador.

Por último, queda la cuestión de donde colocar esta rutina. Podemos hacer dos cosas. La primera es colocarla en una función que sea llamada desde el evento `onLoad`, pero eso tiene la desventaja de que el usuario podría bajarse imágenes mientras la página no haya terminado de cargar pero en parte sea ya visible. La otra opción, que recomiendo, es colocarlo al final de la página, después de todas las etiquetas `IMG` que pueda haber en la misma.

3. El controlador de eventos

Por último, una vez interceptado el evento, debemos comprobar que el botón pulsado sea el derecho y evitar, si es así, que aparezca el menú. Esto lo hará la siguiente función:

```
function noBotonDerecho(e) {
  if (document.layers && (e.which == 3 || e.which == 2))
    return false;
  else if (document.all && (event.button == 2 || event.button == 3))
    alert('No tiene usted permiso para intentar bajarse las imagenes');
  return true;
}
```

Esta función recibe como parámetro un objeto `Event`, que contiene información sobre el evento interceptado, entre ella cuál de los botones ha sido pulsado. Desafortunadamente, dicho objeto es distinto en Explorer y en Netscape, por lo que deberemos acceder a él de manera distinta según sea un explorador u otro.

En ambas versiones de objeto existe una propiedad que contiene el botón pulsado. Esa propiedad, en caso de haber sido pulsado el botón derecho, será 2 o 3 (dependiendo de si el ratón tiene 2 o 3 botones). Pero, en el caso de Netscape, dicha propiedad se llamará `which` y en el de Explorer, `button`.

Por último, una vez comprobado que el botón pulsado ha sido el derecho, debemos evitar que salga el menú. En Netscape basta con que el controlador de evento devuelva `false`, mientras que en Explorer lo que tenemos que hacer es "distracer" al navegador obligándole a hacer otra cosa, por ejemplo mostrar un mensaje de advertencia al usuario. Y con eso y un bizcocho... hasta el próximo artículo.

Mandar imprimir una página

URL: http://html.programacion.net/taller/tw_imprimir.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Gracias a Javascript, tenemos la posibilidad (en Netscape 4 y Explorer 5) de ordenar imprimir nuestra página web.

1. El método `window.print()`

Este método invoca el mismo cuadro de diálogo al que podemos acceder si pulsamos las opciones de menú Archivo - > Imprimir. De este modo se impide que creadores de páginas web pérfidos y malvados puedan mandar imprimir páginas enormes sin nuestro permiso.

Sin embargo, este método sólo está disponible en Netscape 4 y Explorer 5 (y se supone que en sus siguientes versiones, claro), de modo que para utilizarlo conviene asegurarse de que el navegador del usuario lo tiene.

2. Una función `imprimirPagina()`

Esta función comprobará previamente si el usuario tiene un navegador con la capacidad de mandar imprimir desde Javascript, haciendolo en caso afirmativo y sacando un mensaje de error en caso contrario.

```
function imprimirPagina() {  
    if (window.print)  
        window.print();  
    else  
        alert("Lo siento, pero a tu navegador no se le puede ordenar imprimir" +  
            " desde la web. Actualizate o hazlo desde los menús");  
}
```

Como se puede ver, la comprobación se hace con el método más seguro y corto: comprobando que existe el método. De esta manera nos aseguramos de que la función funcionará (valga la redundancia) en cualquier navegador pasado y futuro.

Probarlo es sencillo, no tenéis más que [mandar imprimir ésta página](#) y ver lo que sale.

Dado que el método `print()` es un método del objeto `window` es posible imprimir un marco (`frame`) desde otro. No tenéis más que acceder al objeto `window` que corresponde a dicho marco y ordenarle imprimir con `print()`.

Redirección con Javascript

URL: http://html.programacion.net/taller/tw_redireccion.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

En muchas ocasiones, siendo la más común una mudanza de URL, se necesita redireccionar a los usuarios de una dirección a otra. Te enseñamos como implementarlo en JavaScript.

1. Un ejemplo de redirección

Para ver como funciona el asunto puedes ver el siguiente [ejemplo](#). La página que acabas (fugazmente) de ver utiliza el siguiente código:

```
<HTML>
<HEAD>
  <TITLE>Redireccionado</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    function redireccionar() {
      setTimeout("location.href='101199.htm'", 5000);
    }
  </SCRIPT>
</HEAD>
<BODY onLoad="redireccionar()">
<P>Bla, bla, bla, ...
</BODY>
</HTML>
```

Al terminar el navegador de bajar la página de Internet, el evento `Load` se activa y se ejecuta la función `redireccionar`, la cual consta de una única instrucción. Esta instrucción es `setTimeout`, que recibe como primer parámetro el código que ejecutará cuando pasen los milisegundos que se le pasan como segundo parámetro.

Así, en nuestro ejemplo, cuando pasen 5000 milisegundos (5 segundos), la dirección de la página actual (`location.href`) será la página que le indiquemos.

2. Personalizando la redirección

Esta técnica, la verdad, sólo tiene dos o tres opciones de personalización. Las dos primeras son las obvias: cambiar el lapso de tiempo que transcurre entre la carga de la página y la redirección y el cambio de la dirección que finalmente verá el usuario. La tercera es algo más sutil.

Entra dentro de lo posible que queramos que el usuario sea redireccionado rápidamente, sin

que siquiera se entere de que lo que ha pasado. En tal caso es mejor no esperar a que se termina de cargar la página ni, por supuesto, poner ningún lapso de tiempo. De hecho lo mejor será lo siguiente:

```
<HTML>
<HEAD>
  <TITLE>Redireccionando rápidamente</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    location.href='101199.htm';
  </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

El problema es que al usuario le puede sentar mal que le manejen a su antojo los poderes ocultos e ignotos de Internet. Es elección del diseñador escoger que prefiere.

Trabajar con mapas

URL: http://html.programacion.net/taller/tw_mapas.php

Autor: José Aladro

Email: jaladro@retemail.es

En ocasiones puede resultar complicado lograr que los mapas se comporten como deseamos. En este artículo observaremos varios detalles que nos facilitarán trabajar con los mismos.

1. Tooltips

Supongamos una imagen que tiene asociado un único mapa a una zona rectangular en el centro de la misma. Se trata de una imagen que toda ella hace referencia a un tema mientras que la zona sensible y pulsable con el ratón se reduce a un rectángulo en el centro.

Tendríamos este código:

```
<MAP NAME="map1">
  <AREA SHAPE="rect" COORDS="0,12,104,44"
    HREF="http://www.dondesea.com/blablabla.html">
</MAP>

```

Tal como está escrito, al situar el ratón dentro de la imagen pero fuera del mapa el navegador nos mostraría el tooltip "Texto alternativo". Sin embargo, al movernos encima del mapa el tooltip desaparecería, a pesar de que el texto se asignó a la imagen entera. La solución es colocar el texto tanto al mapa como a la imagen:

```
<MAP NAME="map1">
  <AREA SHAPE="rect" COORDS="0,12,104,44"
    HREF="http://www.dondesea.com/blablabla.html"
    alt="Texto alternativo">
</MAP>

```

Otra opción, quizá con mejor comportamiento, es aprovechar el atributo `NOHREF` de los elementos que definen las áreas:

```
<MAP NAME="map1">
  <AREA SHAPE="rect" COORDS="0,12,104,44"
    HREF="http://www.dondesea.com/blablabla.html"
    alt="Texto alternativo">
  <AREA SHAPE=DEFAULT NOHREF ALT="Texto alternativo">
</MAP>

```

2. Cambio de imágenes

Es muy habitual ver páginas con menús que tienen botones que cambian de estado al pasar el ratón por encima y/o al pulsar sobre ellos. Estos efectos, como [ya sabemos](#), se controlan con los eventos `onMouseLoqueSea`. Lo curioso es que estos eventos se suelen utilizar muy a menudo con imágenes enteras. Un ejemplo típico sería éste:

```

```

Que para funcionar, lógicamente, tendrán que haberse definido las funciones `activar()` y `desactivar()` en alguna parte de la página.

Sin embargo, si queremos que la imagen cambie sólo al entrar el ratón dentro del mapa, habrá que utilizar los eventos con el mapa en lugar de con la imagen:

```
<MAP NAME="map1">
  <AREA SHAPE="rect" COORDS="0,12,104,44"
    HREF="http://www.dondesea.com/blablabla.html"
    alt="Texto alternativo"
    onMouseOver="activar('img1');"
    onMouseOut="desactivar('img1');">
  <AREA SHAPE=DEFAULT NOHREF ALT="Texto alternativo">
</MAP>

```

En la misma línea, se podrían asignar diferentes cambios de imagen dependiendo del área sobre la que entre el ratón utilizando los eventos con cada área:

```
<MAP NAME="map1">
  <AREA SHAPE="rect" COORDS="0,12,51,44"
    HREF="http://www.dondesea.com/blablabla.html"
    alt="Texto1"
    onMouseOver="activar_1('img1');"
    onMouseOut="desactivar('img1');">
  <AREA SHAPE="rect" COORDS="52,12,104,44"
    HREF="http://www.otrositio.com/guaugau.html"
    alt="Texto2"
    onMouseOver="activar_2('img1');"
    onMouseOut="desactivar('img1');">
  <AREA SHAPE=DEFAULT NOHREF ALT="Texto alternativo común">
</MAP>

```

Para ello deberemos tener definidas las funciones `activar_1()`, `activar_2()` y `desactivar()` en alguna parte de la página. Cada una de las 3 funciones asigna a `document[imgName].src` una de las 3 imágenes necesarias.

En cualquier caso, tiene que haber una concordancia entre el nombre con que se ha "bautizado" a la imagen (etiqueta `NAME`) y el parámetro que se pasa a las funciones.

Barra de menús desplegables (I)

URL: http://html.programacion.net/taller/tw_menus1.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Uno de los usos más extendidos del HTML dinámico son estas barras de menús, al estilo de las que puedes ver en las páginas de [Microsoft](#) o [Silicon Graphics](#).

1. En qué consiste

Si pulsáis sobre los enlaces indicados arriba podréis ver unos menús desplegables muy bien hechos. Los nuestros no van a ser tan espectaculares, principalmente para que sea más sencillo entender cómo se hacen. Podéis comprobar las características de nuestros menús en este [ejemplo](#). Si este truco resulta de vuestro interés, procuraremos mejorarlo en próximas entregas.

Ahora veremos cómo se hace. Es recomendable que sepáis algo de [HTML dinámico](#) para entenderlo mejor.

2. Cómo personalizar los menús

Lo primero que vamos a ver son las cosas que deberemos cambiar para poder reutilizar el código en vuestras páginas. Lo primero que necesitamos es una clase CSS a la que llamaremos `menu` y que definirá el aspecto que tienen las capas de menú:

```
.menu {
    position: absolute;
    visibility: hidden;
    background-color: white;           // Color de fondo para Explorer
    layer-background-color: white;    // Color de fondo para Netscape
    color: black;
    border-style: solid;
    border-color: black;
    border-width: 1px;
    padding: 3px;
    font-size: 12px;
    font-family: "arial", "helvetica";
}

.menu A:hover {text-decoration: underline; color: blue;}
.menu A {text-decoration: none; color: black;}
```

Lo que estamos haciendo es señalar al navegador que los elementos de clase `menu` serán capas que posicionaremos de manera absoluta, con un borde negro, un fondo blanco, y cuyos enlaces serán subrayados de la manera que vimos en el artículo anterior. Todas las propiedades que definen el aspecto de las capas las podéis modificar a vuestro gusto.

También debemos definir que habrá dentro de cada menú. Eso lo haremos por medio de etiquetas `DIV` dentro del cuerpo del documento:

```
<DIV id="menu0" CLASS="menu">
  <A HREF="../../recursos/img.htm">Imágenes</A><BR>
  ...
</DIV>
```

Serán capas de clase `menu` que, en nuestro caso, contendrán sólo una lista de enlaces. En el vuestro, si lo deseáis, podéis incluir imágenes y todo lo que se os ocurra.

Ahora tenemos que incluir los enlaces que provocarán el despliegue del menú en caso de que el ratón pase por encima de ellos.

```
<A HREF="pagina.htm" onMouseOver="if (menu[0]) menu[0].activar();">...</A>
```

Incluimos la condición porque puede suceder que el usuario intente activar el menú cuando éste todavía no existe, es decir, cuando todavía no ha acabado de leerse la página

Para asegurarnos de que los visitantes que no tengan la suerte de poseer un navegador de cuarta generación recorrer nuestras páginas, el enlace irá a una página desde la cual podamos acceder a todas las opciones del menú. También debemos incluir, como controlador del evento `mouseover`, una llamada al método `activar` del objeto `Menu` que queramos desplegar.

Claro, que para saber cómo se llama el objeto `Menu` que queremos desplegar en cada enlace, deberemos inicializar dentro del código todos los objetos de tipo `Menu` que se encargarán de desplegar los menús. Esto se hace dentro de la función `inicializar`, situada al final del código JavaScript de la página:

```
function inicializar() {
  ...
  menu[0] = new Menu("menu0", 20, 5);
  menu[1] = new Menu("menu1", 20, 93);
  menu[2] = new Menu("menu2", 20, 250);
}
```

En el ejemplo existen tres menús, que deberán ser los tres primeros elementos del vector `menu`. Cada uno de ellos es inicializado por medio de la línea

```
menu[i] = new Menu(idMenu, posY, posX);
```

Donde `idMenu` es el nombre que, por medio del parámetro `ID`, le hemos puesto a la capa que contiene el menú; siendo `posX` y `posY` el desplazamiento respecto a la esquina superior izquierda del documento donde queremos que aparezca el menú. Este valor hay que colocarlo a mano, por medio del afamado método de ensayo y error.

Si te asaltan las dudas en algún punto, es recomendable ver el código fuente del [ejemplo](#).

3. Cómo funcionan por dentro

En general, lo primero que se debe hacer siempre en una aplicación que utiliza HTML dinámico es ver con qué navegador se está viendo la página. Para eso utilizaremos un objeto específico:

```
function DetectorNavegador() {  
    this.NS4 = document.layers;  
    this.IE4 = document.all;  
    this.DHTML = this.NS4 || this.IE4;  
}  
  
var soporta = new DetectorNavegador();
```

En realidad con un par de variables hubiera bastado, pero así podéis ver cómo se crea y se aplica un objeto creado por nosotros. La función `DetectorNavegador` es un constructor que llamaremos con el operador `new`. El constructor comprueba la existencia de objetos específicos de cada uno de los navegadores y vincula esa existencia al valor lógico de las propiedades `NS4`, `IE4` y `DHTML`.

Ahora necesitaremos un par de variables globales:

```
var menu = new Array();  
var menuActivo = null;
```

El vector `menu` contendrá todos los objetos `Menu`, mientras que la variable `menuActivo` será una referencia al objeto `Menu` que contenga al menú que esté desplegado en estos momentos, siendo `null` si no hay ningún menú desplegado.

3.1. Objeto Menu

Este objeto es el encargado de manejar los menús. Para su correcto funcionamiento necesita que hayamos creado las dos variables señaladas anteriormente y el objeto `soporta` de tipo `DetectorNavegador`. Para entender cómo funciona vamos a estudiarlo por partes. Lo primero será examinar el constructor:

```
function Menu(capaID, top, left) {  
    this.activar = activarMenu;  
    this.mostrar = mostrarMenu;  
    this.ocultar = ocultarMenu;  
    this.cambiarPosicion = cambiarPosicionMenu;  
    this.capaRefStr = (soporta.NS4) ?  
        'document["'+capaID+'"]' :  
        'document.all["'+capaID+'"]';  
    this.estiloRefStr = (soporta.NS4) ? '' : '.style';  
    this.topRefStr = (soporta.NS4) ? '.top' : '.pixelTop';  
    this.leftRefStr = (soporta.NS4) ? '.left' : '.pixelLeft';  
    this.cambiarPosicion(top, left);  
}
```

Lo primero que hace el mismo es convertir en métodos de la función a cuatro funciones que posteriormente estudiaremos. Luego declara cuatro propiedades que serán cadenas que nos permitirán acceder a las propiedades de la capa que contiene al menú, dependiendo del navegador que utilicemos (ya que la jerarquía de objetos en uno y otro es distinta).

La propiedad `capaRefStr` contiene una cadena que contiene la referencia a la capa que contiene el menú. `estiloRefStr` contiene la referencia a las propiedades de una capa y, finalmente, `topRefStr` y `leftRefStr` indican qué propiedad específica hay que cambiar para modificar la posición de la capa.

Por último, el constructor llama a la función encargada de cambiar la posición del menú para colocarla donde haya indicado el usuario al crear el objeto.

Por último, veamos como funcionan los métodos del objeto:

```
function activarMenu() {
    if (soporta.DHTML && menuActivo != this) {
        if (menuActivo) menuActivo.ocultar();
        menuActivo = this;
        this.mostrar();
    }
}
```

Este primer método se encarga de desplegar el menú. Primero comprueba que nuestro navegador lo soporte y que el menú no esté ya desplegado. Luego, si hay otro menú desplegado, llama al método `ocultar` del mismo. Finalmente, asigna la variable `menuActivo` al menú y lo despliega llamando al método `mostrar`.

```
function mostrarMenu() {
    eval(this.capaRefStr + this.estiloRefStr + '.visibility = "visible"');
}

function ocultarMenu() {
    eval(this.capaRefStr + this.estiloRefStr + '.visibility = "hidden"');
}
```

Estos dos métodos se encargan de mostrar u ocultar el menú, respectivamente. Para ello utilizan dos de las cuatro propiedades creadas en el constructor. Lo que hacen es crear una cadena que contenga una sentencia que haga el trabajo. Luego llaman a la función predefinida `eval`, que se encargará de ejecutar dicha sentencia.

Dado que éstas funciones sólo se llaman desde `activarMenu`, una vez éste ha comprobado que el navegador soporta DHTML, ya no necesitan código que realice dicha comprobación.

```
function cambiarPosicionMenu(top, left) {
    if (soporta.DHTML) {
        eval(this.capaRefStr + this.estiloRefStr + this.topRefStr + ' = top');
        eval(this.capaRefStr + this.estiloRefStr + this.leftRefStr + ' = left');
    }
}
```

Por último, y al igual que las dos funciones anteriores, este método cambia la posición del menú utilizando las propiedades creadas en el constructor.

3.2. Eventos

Como hemos visto, hemos incluido la llamada al método `activar` dentro del controlador de evento `onMouseOver` de los enlaces. Pero no hemos incluido ninguna manera de comprobar que el usuario ha pulsado el ratón para ocultarlos. Eso lo lograremos interceptando los eventos.

Desafortunadamente, la manera de hacerlo también es distinta en cada navegador, pero haremos lo que podamos.

Dentro de la función de inicialización que vimos antes tenemos el código encargado de realizar la intercepción:

```
function inicializar() {  
    if (soporta.DHTML) {  
        if (soporta.NS4)  
            document.captureEvents(Event.MOUSEUP);  
        document.onmouseup = ocultarMenuActivo;  
    }  
    ...  
}
```

No vamos a entrar en los distintos modelos de eventos. Por ahora basta decir que la última línea indica la función que se ejecutará si pulsamos el ratón en alguna parte del documento actual. Las dos anteriores indican al Netscape Communicator que deseamos interceptar ese tipo de eventos. El Explorer no necesita que se lo digamos.

La función que se llamará es, por tanto, ésta:

```
function ocultarMenuActivo(e) {  
    if (menuActivo) {  
        menuActivo.ocultar();  
        menuActivo = null;  
    }  
}
```

Comprueba si existe algún menú desplegado, para ocultarlo y poner la variable `menuActivo` a `null`, para indicar que ya no hay menús desplegados.

Subrayado de enlaces (II)

URL: http://html.programacion.net/taller/tw_subrayado2.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

El soporte de parte del estándar CSS2 tanto en Explorer 4 como en las nuevas versiones del futuro Netscape 5 nos permiten realizar este truco de una manera mucho más sencilla.

1. Adoremos al estándar CSS2 (alabado sea)

En general, todas las propiedades de hojas de estilo implementadas en los navegadores actuales responden al estándar CSS1. Por el momento (con la posible exclusión del Mozilla de Netscape) ningún navegador soporta este estándar por completo pero, sin embargo, algunos interpretan algunas propiedades del CSS2.

Este nuevo estándar incluye muchas cosas nuevas, incluyendo algunas pseudoclases que dan un comportamiento dinámico a los enlaces. Entre ellas está la pseudoclase `hover`.

Los enlaces siempre habían tenido pseudoclases dependiendo del estado en que estuvieran: normal, ya visitado o con el usuario pulsando sobre ellos. De este modo se podían modificar sus atributos por medio de hojas de estilo al igual que se hacía antes desde la etiqueta `BODY`. Esta nueva pseudoclase permite modificar los atributos de un enlace cuando el usuario pasa el ratón por encima del mismo.

2. Modo de empleo

El método es muy similar al que utilizabamos en la anterior entrega para dejar sin subrayado a nuestros enlaces. En esta ocasión, además, vamos a pintarlos de negro. Cuando el usuario pase el ratón por encima se subrayarán y colorearán de azul. Estos son los estilos que debemos incluir:

```
<STYLE TYPE="text/css">
  A.link { text-decoration: none; color: black; }
  A:hover.link { text-decoration: underline; color: blue; }
</STYLE>
```

Añadiendo a todas nuestras etiquetas `A` el parámetro `CLASS="link"` lograremos el efecto que buscamos:

Conversor pesetas / euros

URL: http://html.programacion.net/taller/tw_conversor.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Si quieres ofrecer servicios de valor añadido a tus visitantes, este sencillo conversor de euros a pesetas es una buena manera de empezar.

1. En qué consiste

Para usar esta calculadora, simplemente introduce una cantidad en la casilla correspondiente (la izquierda si dicha cantidad está en euros o la derecha si es en pesetas). Luego pulsa el botón situado al lado contrario y la cantidad convertida aparecerá.

2. Cómo realizarlo

Como suele ocurrir en estos casos, esta pequeña aplicación se puede dividir en dos partes, la incluida en las etiquetas HTML y las funciones Javascript. Vamos a empezar por lo primero:

```
<form name="conversor">
  <table border=0>
    <tr>
      <td><input type="text" name="euros"> euros</td>
      <td><input type="text" name="pesetas"> pesetas</td>
    </tr>
    <tr>
      <td><input type="button" value="<< Convertir a euros"
        onClick="convertirAEuros()"></td>
      <td><input type="button" value="Convertir a pesetas >>"
        onClick="convertirAPesetas()"></td>
    </tr>
  </table>
</form>
```

Lo primero que hay que resaltar es que tanto la etiqueta `FORM` como los dos campos de texto tiene definido el parámetro `NAME`. Esto nos permitirá, en el código Javascript, hacer referencia a las mismas cómodamente, llamándolos por su nombre. Vemos también que se realizan llamadas a funciones Javascript en ambos botones, una función para convertir a euros y otra para convertir a pesetas. Estas son:

```
var euro = 166.386;

function convertirAEuros() {
  document.conversor.euros.value =
    Math.round(parseInt(document.conversor.pesetas.value) * 100 / euro)
```

```

    / 100;
}

function convertirAPesetas() {
    document.conversor.pesetas.value =
        Math.round(parseInt(document.conversor.euros.value) * euro);
}

```

Es ahora cuando se ve la necesidad de bautizar a nuestro formulario y nuestras cajas de texto. Vamos a analizar en detalle la primera función, ya que la segunda es casi equivalente. Consta sólo de una asignación de un valor a `document.conversor.euros.value`. ¿Y eso qué es?

Mirémoslo detenidamente, leyéndolo de izquierda a derecha. Lo primero que encontramos es el objeto `document`, que hace referencia a la página HTML actual. Dentro de dicha página (seguimos leyendo) accedemos al formulario llamado `conversor` y, dentro del mismo, al elemento del formulario llamado `euros` (en nuestro caso, una caja de texto). Por último, el `value` indica que el valor que deseamos alterar es el atributo `value`, que alberga el contenido actual de la caja de texto.

El código situado en la parte derecha de la igualdad forma las operaciones necesarias para convertir el contenido de la caja de texto que contiene la cantidad en `document.conversor.pesetas.value` a euros. Para ello convertimos a número (con `parseInt`) dicho contenido. Lo dividimos por la constante `euro` (que contiene el número de pesetas que tiene un euro). Lo multiplicamos por cien, lo redondeamos (con `Math.round`) y lo dividimos por cien.

Este jaleo con el número 100 es para poder redondear el número resultante en céntimos de euro.

El código para realizar la operación inversa es equivalente, aunque algo más sencillo al no tener que manejar céntimos. Ahora sólo queda que compliquéis vosotros el asunto. Podéis hacer un conversor entre múltiples divisas o llegar al extremo de intentar que el Javascript os resuelva la declaración de la renta (bueno, quizá esto sea excesivo).

Convertir XML en HTML utilizando XSLT

URL: http://html.programacion.net/taller/tw_xml_y_xslt.php

Autor: Joaquin Bravo Montero

Email: jbravo@retemail.es

!!!Artículo adaptado a la recomendación XSLT del 16 Noviembre de 1999!!!

Este es el primer capítulo de una serie de artículos en los que iremos **estudiando qué posibilidades y herramientas existen hoy en día para convertir un documento XML en HTML y cómo visualizarlo en un navegador.**

1. ¿Qué es XSLT?

Uno de los problemas del HTML es que no identifica lo que está representando, se preocupa principalmente de que eso tiene que ir en un color o con un tipo de letra determinada, pero no dice que lo que está mostrando es el título de un libro o el precio de un artículo. **El XML hace precisamente esto: describe el contenido de lo que etiqueta**, sin tener en cuenta en un principio la forma de presentarlo.

El W3C está trabajando actualmente en el desarrollo de **un lenguaje de hojas de estilo denominado XSL** (Extensible Style Language) **que nos permita dar al XML un formato de salida comprensible para los humanos.**

Mediante el XSL podremos:

- Transformar un documento XML en otro XML.
- O convertirlo en otro formato de salida como puede ser [RTF](#), [PDF](#)

El W3C ha dividido recientemente esta especificación en dos partes: [XSL](#) y [XSLT](#), siendo la XSLT la parte de la especificación que **describe el lenguaje que nos permite convertir un documento XML en otro XML.**

2. Motores XSLT

Hay que tener en cuenta que la especificación XSLT sigue siendo todavía un borrador (se espera que sea una propuesta definitiva para el verano del 99) y que, por lo tanto:

- Al no ser definitiva puede todavía sufrir algunos cambios.
- Y que los motores que existen, no la implementen en su totalidad o proporcionen algunos añadidos que la especificación no contemple.

Existen muchos [motores](#) e incluso, como veremos en otros artículos, el propio Explorer 5 implementa un motor XSL que nos permite visualizar directamente XML.

En este artículo vamos a manejar el procesador XSLT denominado [XT](#), esencialmente por dos razones:

- Ha sido desarrollado por [James Clark](#), que es el artífice de la parte de la especificación que trata el tema de las transformaciones.
- Y porque su utilización resulta muy sencilla. Existe una [versión ejecutable](#) para Windows, aunque para poderla utilizar necesitamos la [Maquina Virtual Java](#) que viene con el Explorer 5.

En el caso de que estemos en condiciones de trabajar con la versión ejecutable, para utilizarla no tenemos más que escribir desde la línea de comandos:

```
xt fichero_xml fichero_xsl fichero_html
```

Si no es posible, al tratarse de una aplicación en Java no tenemos más que tener instalado el JDK (actualmente trabajo con la versión 1.1.5 y funciona correctamente) y seguir los siguientes pasos:

- Bajarnos la [distribución](#) del XT. Donde, entre otros, se encuentran los ficheros xt.jar y sax.jar.
- Bajarnos el [XP](#), parser de XML en Java, desarrollado por James Clark. Donde, entre otros, se encuentra el fichero xp.jar.
- Crearnos un fichero [xslt.bat](#), que debe tener el siguiente aspecto:

```
set oldClassPath=%CLASSPATH%
set CLASSPATH=%CLASSPATH%;d:\xt\xt.jar;d:\xt\sax.jar;d:\xt\xp.jar;
java com.jclark.xsl.sax.Driver %1 %2 %3
set CLASSPATH=%oldClassPath%
```

Indicándole en la definición del `set CLASSPATH` donde se encuentran en nuestra máquina los tres ficheros .jar antes mencionados.

- Y, finalmente, para ejecutarlo no tendremos más que escribir:

```
xslt fichero_xml fichero_xsl fichero_html
```

Para cualquier duda sobre la instalación y utilización os recomiendo que visitéis la sección sobre XT de la Web de James Clark.

3. El ejemplo

En nuestro ejemplo tendremos un único fichero XML en el que he reunido mis direcciones de gastronomía preferidas y del que obtendremos dos páginas HTML diferentes con dos XSLT distintas.

Este fichero, [gastrono.xml](#) tiene la siguiente estructura:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<direcciones>
  <tema>Mis direcciones de gastronomía</tema>
  <intro>Recopilación de mis direcciones de gastronomía favoritas.</intro>

  <direccion>
    <titulo>El Universal</titulo>
    <url>http://www.el-universal.com/scannone/</url>
    <descripcion>La cocina y la mesa de Armando Scannone con
      una amplia y buena selección de artículos recetas
      y consejos</descripcion>
  </direccion>

  .....
  .....
</direcciones>
```

Y dos XSLT:

- [gastrono1.xsl](#), con la que obtenemos la página [gastrono1.htm](#).
- [gastrono2.xsl](#), con la que obtenemos la página [gastrono2.htm](#).

Y la forma de generar el HTML es escribiendo desde la línea de comandos las siguientes líneas:

```
xt gastrono.xml gastrono1.xsl gastrono1.htm
xt gastrono.xml gastrono2.xsl gastrono2.htm
```

O

```
xslt gastrono.xml gastrono1.xsl gastrono1.htm
xslt gastrono.xml gastrono2.xsl gastrono2.htm
```

una para cada XSLT.

Aunque el objetivo de este artículo no es profundizar en la sintaxis de las XSLT, veamos el código de [gastrono1.xsl](#) para destacar algunos conceptos:

```
<?xml version="1.0"?>
<xsl:stylesheet
version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="direcciones">
  <html>
  <head>
    <title><xsl:value-of select="tema"/></title>
    <link rel="stylesheet" type="text/css" href="gastrono.css"/>
  </head>
  <body>
```

```

        <center>
        
        </center>
        <xsl:apply-templates/>
    </body>
</html>

</xsl:template>

<xsl:template match="tema">
    <p class="titulo">
        <xsl:apply-templates/>
    </p>
</xsl:template>

<xsl:template match="direccion">
    <div class="direccion">
        <p>
            <xsl:apply-templates select="titulo"/><br/>
            <xsl:apply-templates select="url"/><br/>
        </p>
        <xsl:apply-templates select="descripcion"/>
    </div>
</xsl:template>

<xsl:template match="titulo">
    <span class="dirtitulo">
        <xsl:apply-templates/>
    </span>
</xsl:template>

<xsl:template match="url">
    <xsl:element name="A">
        <xsl:attribute name="HREF"><xsl:value-of select="."/></xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

<xsl:template match="descripcion">
    <p class="descripcion">
        <xsl:apply-templates/>
    </p>
</xsl:template>

</xsl:stylesheet>

```

Como podemos observar:

- La sintaxis es XML. Por tanto, una vez que aprendamos cómo funciona el XML, no debe resultar muy complicado realizar nuestras primeras XSLT.
- Como ya hemos dicho, mediante una XSLT convertimos XML en XML; en nuestro ejemplo esto no es del todo cierto, ya que estamos convirtiendo XML en HTML que es una aplicación SGML pero no XML. En definitiva, hemos tenido que indicar al procesador que la salida va a ser HTML 4.0. Esto se realiza mediante la sentencia:

```
<xsl:output method="html"/>
```

Aunque como podeis observar no la hemos utilizado, ya que el procesador XT genera de forma automatica HTML al ver que la raiz de la salida es el elemento `html`.

```

<xsl:template match="direcciones">
    <html>
        ....
</html>

```

Actualmente el W3C está trabajando en una reformulación del HTML como una aplicación XML. Esta nueva versión del HTML se llama XHTML.

Para más información sobre XML y XSL os recomiendo que visitéis el [tutorial sobre XML](#), que estoy escribiendo en este mismo web.

Y si os animais a escribir diferentes XSLT para representar gastrono.xml estaría encantado de recibirlas y publicarlas.

4. Direcciones de interés

Y a continuación una serie de direcciones que os pueden resultar útiles para complementar el artículo:

El procesador de XSLT desarrollado por James Clark, la misma persona que desarrolla la especificación

La especificación de XSLT, en la cual se describe el lenguaje que nos permite transformar un documento XML en otro XML. Todavía es un borrador de trabajo.

La sección dedicada al XSL en la Web sobre XML y SGML de Robin Cover.

Estupendo tutorial de Miloslav Nic con multiples ejemplos a lo largo de los cuales se estudian diferentes aspectos sobre la utilización de una XSLT.

Lista de correo sobre XSL que podemos encontrar en las páginas de Mulberry Technologies. Indispensable para estar al día.

¿Cuánto queda para el año 2000?

URL: http://html.programacion.net/taller/tw_contador2000.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

En muchas páginas, atrapadas por la moda del supuesto fin de milenio, colocan contadores que nos indican lo que queda para que llegue el año 2000. Te enseñamos cómo lo hacen.

1. En qué consiste

Si cambias la fecha de tu ordenador y vuelves a cargar esta página, verás que la frase de arriba cambia. Esto es síntoma inequívoco de que dicha frase está escrita por medio de un programa que se ejecuta en tu ordenador. Por un `script`.

2. Cómo realizarlo

Necesitaremos dos cosas para realizar este truco. La primera es colocar en el lugar de nuestras páginas donde deseemos que se escriba la cuenta atrás una llamada a la función que se encargará de escribirlo:

```
<P><SCRIPT LANGUAGE="Javascript">
<!--
    escribirCuentaAtras();
// -->
</SCRIPT></P>
```

La segunda, como os podéis imaginar, es escribir dicha función. El código es el siguiente:

```
function escribirCuentaAtras() {
    var fechaActual = new Date();
    var fecha2000 = new Date("January 1, 2000");
    var tiempoRestante = fecha2000.getTime() - fechaActual.getTime();
    var dias = Math.floor(tiempoRestante / (1000 * 60 * 60 * 24));
    if (dias > 1)
        document.write("Quedan " + dias + " días para el año 2000")
    else if (dias == 1)
        document.write("Sólo queda un día para el año 2000")
    else if (dias == 0)
        document.write("Esta noche llega el apocalipsis")
    else
        document.write("Pero... ¿todavía funcionan los ordenadores?");
}
```

Lo primero que hace esta función es crear **instancias** del objeto `Date`. La fecha actual se crea sin aportar ningún parámetro al constructor, es decir, se crea igualándolo a lo bruto con `new Date()`. Esto consigue que la fecha creada contenga el día y la hora actuales.

La fecha del 1 de enero del año 2000 se crea llamando al constructor con un parámetro que permita identificar la fecha que deseamos crear. Javascript permite más de una manera de hacerlo; en este caso lo que hacemos es pasarle un único argumento de tipo cadena que contiene la fecha expresada en inglés. También podríamos escribir `new Date(año, mes, día)`, por ejemplo, con los tres parámetros expresados en números.

Lo siguiente que hacemos es restar ambas fechas. Para eso utilizamos el método `getTime()` que devuelve el número de milisegundos transcurridos desde las 0:00 horas del 1 de enero de 1970. La razón de tan extraña manera de convertir una fecha a un número (que se pueda sumar y restar cómodamente) es que el sistema operativo **Unix** también lo hace así. Las cosas raras se mantienen. Una vez que tenemos ambas fechas convertidas a números, las restamos y convertimos el resultado, de modo que éste esté expresado en días, y no en milisegundos.

Por último, nos queda escribir el resultado. Para ello utilizaremos el famoso método `document.write()`. Lo que vayamos a escribir depende del número de días que queden. No resultaría muy elegante decir que quedan -123 días para el año 2000, por poner un ejemplo.

Navegación con menús desplegables

URL: http://html.programacion.net/taller/tw_menus_desp.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Una buena manera de ahorrar espacio en nuestras páginas es usando menús desplegables en lugar de interminables listas de enlaces.

1. En qué consiste

Lo que queremos es incorporar a nuestras páginas menús de este tipo:

De este modo podemos incluir innumerables enlaces en un muy pequeño espacio, ahorrándonos esas listas de enlaces que parecen la lista de la compra de una familia numerosa. En contra de estos menús se puede alegar que no resultan muy vistosos que digamos.

2. Cómo realizarlo

Para realizar este truco necesitaremos crear la lista desplegable, colocando como parámetro `VALUE` la dirección de la página web e incluyendo un evento `onChange` (que se ejecuta cuando el usuario cambia la opción escogida en la lista) que llamará a una función que definiremos más adelante:

```
<FORM name="formulario">
  <SELECT NAME="menu" SIZE=1 onChange = "irA(this)">
    <OPTION VALUE="">Visitar
    <OPTION VALUE="http://www.ole.es">¡Olé!
    <OPTION VALUE="http://www.ozu.es">Ozú
    ...
  </SELECT>
</FORM>
```

La función `irA` recibe como parámetro `this`, que es una referencia al objeto actual, en este caso la lista. El código de dicha función será el siguiente:

```
function irA(menu){
  window.location.href = menu.options[menu.selectedIndex].value;
}
```

Esta función cambia la dirección de la página actual por el valor de la opción

`menu.options[...].value`) que el usuario ha escogido (`menu.selectedIndex`).

Subrayado de enlaces (I)

URL: http://html.programacion.net/taller/tw_subrayado1.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Leyendo esta página aprenderás a hacer enlaces que se subrayen automáticamente cuando pasas el ratón por encima. Aún funcionando sólo con Explorer 4 y la versión pre-alpha del Netscape 5, este truco es lo suficientemente interesante para ponerlo en tus páginas y suficientemente inocuo para que a aquellos que no puedan verlo no les afecte demasiado.

1. Lo primero: que los enlaces no estén subrayados

Parece lógico que nuestra primera necesidad sea conseguir que nuestros enlaces no estén subrayados para poder subrayarlos después. Como tantas otras cosas, esto lo conseguiremos por medio de hojas de estilo. Crearemos una clase de la etiqueta `A` de este modo:

```
<STYLE TYPE="text/css">
  A.link { text-decoration: none; }
</STYLE>
```

Añadiendo a todas nuestras etiquetas `A` el parámetro `CLASS="link"` lograremos que no estén subrayadas.

2. Cómo subrayar en Explorer 4

Este es el código necesario:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
  <!-- Se esconde el código a navegadores antiguos
  function encender(e) {
    src = event.srcElement;
    if (src.tagName == "A") {
      src.style.textDecoration = "underline";
    }
  }

  function apagar(e) {
    src = event.srcElement;
    if (src.tagName == "A") {
      src.style.textDecoration = "none";
    }
  }

  document.onmouseover=encender;
  document.onmouseout=apagar;
  // -->
</SCRIPT>
```

Es un poco complicado y requiere algunas explicaciones. Las dos últimas líneas provocarán que, cuando pasemos el ratón por algún elemento se llame a la función `encender`, haciendo lo mismo en el caso de que nos vayamos del elemento, momento en el que se llamará a `apagar`.

Cada una de las funciones reconoce que elemento les ha llamado por medio de la línea `src = event.srcElement`. `event` contiene toda la información del evento que ha ocurrido en el documento y, entre ella, una referencia al elemento que la ha provocado. Si el elemento en cuestión a resultado ser un enlace (`if (src.tagName == "A")`), procedemos a cambiar el estilo del mismo: `src.style.textDecoration = "loquecorresponda"`. De este modo nuestros enlaces quedarán tal que así:

3. Cómo subrayar en Netscape Gecko

Gecko es una versión pre-alpha del próximo navegador de Netscape. Tiene muchas mejoras en todos los aspectos, y entre ellas podemos encontrar que existe un modo de subrayar los enlaces. Es el siguiente:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
  <!-- Se esconde el código a navegadores antiguos
    function encender(e) {
      this.style.textDecoration = "underline";
    }

    function apagar(e) {
      this.style.textDecoration = "none";
    }

    function inicializarEnlaces() {
      var links = document.getElementsByTagName("A");
      for (var i=0; i<links.length; i++) {
        links[i].onmouseover=encender;
        links[i].onmouseout=apagar;
      }
    }

    window.onload = inicializarEnlaces;
  // -->
</SCRIPT>
```

Como veis algunas cosas son más sencillas y otras más complicadas. En el fondo hace lo mismo que el Explorer, pero en sitios distintos. Mientras en el Explorer se llamaba a `encender` cuando el ratón pasaba por encima de cualquier elemento, aquí discriminamos antes el elemento (con algo de trabajo) y así nos ahorramos el tener que mirarlo en el código de las funciones.

Lo último que queda es integrar ambos `scripts` de modo en uno que funcione en ambos navegadores. Os lo dejo, como suele decirse, como ejercicio al lector. La solución la veréis examinando el código fuente de esta página.

Protección con contraseña

(I)

URL: http://html.programacion.net/taller/tw_password1.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Si quieres proteger una página de una manera simple para impedir que entre en ella todo el mundo, en esta página te enseñaremos un `script` que te permite hacerlo de manera sencilla.

1. Cómo usarlo

Pongamos, por ejemplo, que quisiera poner una contraseña para una página llamada `bienvenido.html`. En tal caso colocaremos el siguiente `script`:

```
<HTML>
<HEAD>
  <TITLE>Ejemplo de imagenes</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    function entrar() {
      var password = prompt("Introduce la contraseña:", "");
      if (password) location.href = password + ".html";
    }
  </SCRIPT>
</HEAD>
<BODY>
  <A HREF="javascript:entrar()">Pulsa aquí</A> para entrar en las
  páginas protegidas.
</BODY>
</HTML>
```

Los usuarios que quieran acceder a la página deberán pulsar [aquí](#) y conocer la clave que será el nombre de la página protegida (`bienvenido`, en este caso).

2. Cómo funciona

Este `script` no es que sea la cosa más compleja del universo. Simplemente, desde el enlace se llama a la función `entrar`. En ella se le pregunta al usuario por la contraseña por medio de la función `prompt`, que devuelve lo que haya tecleado el usuario. Guardamos ese resultado en la variable `password` y nos vamos a la página que resulta de añadir a esa variable la extensión `html`.

Es un poco complicado y requiere algunas explicaciones. Las dos últimas líneas provocarán que, cuando pasemos el ratón por algún elemento se llame a la función `encender`, haciendo lo

mismo en el caso de que nos vayamos del elemento, momento en el que se llamará a `apagar`.

Si el usuario introduce una contraseña incorrecta pueden pasar, por tanto, dos cosas. Que la página exista, en cuyo caso irá a ella. O que no exista, en cuyo caso el servidor dará un error. Esto último, de todos modos, es mejor comprobarlo porque algunos servidores muestran el contenido del directorio en el caso de que el usuario pusiera un punto como clave. Conviene, por tanto, probar este `script` antes de proteger nuestras páginas con él.

Cambio de imágenes

URL: http://html.programacion.net/taller/tw_rollover.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Si quieres hacer un índice cuyos elementos se iluminen si pasas el ratón por encima suyo (como en mi [Curso de JavaScript](#)), en esta página te enseñaremos cómo. Funciona con todos los navegadores capaces de reproducir este efecto (Netscape 3, Explorer 4 y superiores).

1. Una sólo imagen

Al pasar el ratón por encima de la imagen que tienes aquí debajo, podrás observar que el gráfico cambia.

Para hacer esto deberemos crear dos gráficos distintos: el que se verá normalmente y el que únicamente podrá verse cuando el ratón pase por encima. Si llamamos al primero, por ejemplo, `apagado.gif` y al segundo `encendido.gif` el código necesario para que el truco funcione es:

```
<HTML>
<HEAD>
  <TITLE>Ejemplo de imagenes</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    if (document.images) {
      var activado=new Image();
      activado.src="encendido.gif";
      var desactivado= new Image();
      desactivado.src="apagado.gif";
    }
    function activar(nombreImagen) {
      if (document.images) {
        document[nombreImagen].src=activado.src; }
    }
    function desactivar(nombreImagen) {
      if (document.images) {
        document[nombreImagen].src=desactivado.src; }
    }
  </SCRIPT>
</HEAD>
<BODY>
  <A HREF="mipagina.html"
    onMouseOver="activar('prueba');"
    onMouseOut="desactivar('prueba');">
    <IMG NAME="prueba" SRC="apagado.gif" BORDER=0></A>
  </BODY>
</HTML>
```

Lo primero que hay que indicar es que para que funcione el invento la imagen debe ser un enlace. ¿Por qué? Porque los eventos que controlan si el ratón pasa o no por encima no son

admitidos por la etiqueta ``. También deberemos "bautizar" nuestra imagen usando el atributo `NAME="como-se-llame"` para permitir al código su identificación posterior.

El ejemplo funciona de la siguiente manera: en principio la imagen que vemos es la desactivada, que es la que indica la etiqueta ``. Al pasar el ratón por encima de ella el evento `onMouseOver` llamará a la función `activar` llevando como parámetro el nombre de la imagen. Esta función sustituirá en el objeto `document` el nombre del fichero donde se guarda la imagen por `encendido.gif`, que es el gráfico activado. Cuando apartemos el ratón de la imagen, será el evento `onMouseOut` el que se active, llamando a `desactivar`. Esta función sustituirá el gráfico de nuevo, esta vez por `apagado.gif`.

Leyendo esta explicación parece que una parte del código sobra. ¿Para qué sirve declarar dos objetos `Image` para albergar los gráficos? ¿No bastaría con cambiar directamente el nombre de la imagen escribiendo `document[nombreImagen].src = 'encendido.gif';`? Pues no del todo. Efectivamente funcionaría, pero cada vez que cambiásemos el nombre el navegador se traería la imagen del remoto lugar donde se encontrara. Es decir, cada vez que pasásemos el ratón por encima de la imagen o nos alejáramos de ella tendríamos que cargar (aunque ya lo hubiésemos hecho antes) el gráfico correspondiente. Es cierto que con la caché del navegador este efecto quedaría algo mitigado, pero siempre es mejor precargar las imágenes usando el objeto `Image`, ya que así el navegador comenzará a leer el gráfico en cuanto ejecute el código en vez de esperar a que el usuario pase por encima de la imagen con el ratón. El objeto `Image` tiene como atributos los distintos parámetros que puede tener la etiqueta ``.

Por último, hay que estudiar que significa eso de `if (document.images)`. En los navegadores que no poseen JavaScript 1.1 (léase Netscape 2 y Explorer 3 e inferiores) el objeto `Image` no existe y dará un mensaje si se lo encuentra por ahí. La solución a este problema consiste en detectar la capacidad del navegador para manipular gráficos preguntándole por la existencia del array `document.images`. Así podremos no crear las variables que guardan los gráficos ni ejecutar el código de las funciones para activar y desactivar en el caso de que el navegador no soporte este array, lo cual es lo mismo que decir que soporta la versión 1.1 de JavaScript.

2. Varias imágenes

La cosa se complica cuando utilizamos más de una imagen. Para simplificar el código conviene renombrar todas los gráficos que vamos a utilizar del siguiente modo:

Todos comienzan con la misma palabra, luego su número, y finalmente la letra `a` si corresponde a un gráfico que se muestra sólo al pasar el ratón por encima.

Lo primero que vamos a modificar del código original es la precarga de imágenes. Lo que haremos será crear dos arrays, uno para las imágenes originales (`desactivado`) y otro para las que se iluminan (`activado`). El índice de cada array será el nombre que luego asignaremos a la imagen en el parámetro ``.

```
if (document.images) {
```

```
var activado = new Array();
var desactivado = new Array();
for (i=0; i<=1; i++) {
    activado["menu"+i] = new Image();
    desactivado["menu"+i] = new Image();
    activado["menu"+i].src = "menu"+i+".a.gif";
    desactivado["menu"+i].src = "menu"+i+".gif";
}
```

Las funciones que modifican las imágenes no cambian mucho. Tan sólo el lado derecho de la igualdad. Ahora el nuevo gráfico a cargar será el correspondiente dentro de cada array.

```
function act(nombreImagen) {
    if (document.images)
        document[nombreImagen].src=activado[nombreImagen].src;
}
function desact(nombreImagen) {
    if (document.images)
        document[nombreImagen].src=desactivado[nombreImagen].src;
}
```

Por último deberemos tener cuidado con la manera de nombrar las imágenes en el código HTML. Debemos llamarlas por el nombre (sin el .gif) de su gráfico desactivado:

```
<A HREF="../recursos.htm"
    onMouseOver="act('menu0'); "
    onMouseOut="desact('menu0'); ">
<IMG NAME="menu0" SRC="menu0.gif" BORDER=0></A>
```

De este modo lograremos un bello resultado. Este truco es, por supuesto, extensible al número de imágenes que se necesiten, siempre y cuando se las nombre como se ha indicado y se aumente el número que aparece como límite en el .

Cómo no quedar encerrado en los marcos

URL: http://html.programacion.net/taller/tw_marcos.php

Autor: Daniel Rodríguez Herrera

Email: multivac@idecnet.com

Resulta desgraciadamente normal el encontrarse con páginas con frames, cuya sección de enlaces "encierra" páginas externas dentro de uno de ellos. Este artículo te ayudará a no cometer ese error, e incluso a evitar que tu página quede encerrada en los frames de otros.

1. No encierres los enlaces al exterior

Cuando tienes una página con `frames`, puedes elegir el frame donde se mostrarán tus enlaces por medio del parámetro `TARGET` de la etiqueta `<A>`. En este parámetro especificamos el nombre del frame donde queremos que se abra el enlace.

Pues bien, existen algunos nombres especiales que podemos indicar en ese parámetro. Son estas:

`_top`

Esta es la madre del cordero. Poniendo `TARGET="_top"` en nuestros enlaces estos se abrirán en la ventana completa impidiendo que los encerremos.

`_blank`

Este puede ser un método alternativo si no queremos que el navegante deje nuestras páginas. Abrirá el enlace en una ventana nueva.

`_self`

Abre el enlace en el mismo frame donde está alojado. No es de mucha utilidad práctica.

`_parent`

No se utiliza mucho. En la mayoría de los casos es equivalente a `_top` y se diferencia en el caso de que tengamos frames anidados. No es muy usado tampoco.

Así pues lo único que tendremos que hacer para no encerrar nuestros enlaces al exterior es lo siguiente:

```
<A HREF="http://www.programacion.net" TARGET="_top">
```

Pero, para que engañarnos, puede resultar algo peñazo poner esto en todos los enlaces, si resulta que estamos en una página donde todos los enlaces son al exterior, por ejemplo. En tal caso podremos alterar el valor por defecto de `TARGET` por medio de la etiqueta `<BASE>`:

```
<BASE TARGET="_top">
```

Con esto en la cabecera de nuestro documento HTML solucionaremos el problema. Si hay dudas, consulten el [curso de HTML](#) que albergamos en estas páginas.

2. Cómo evitar que te encierren

A pesar de nuestros esfuerzos por ser cuidadosos, puede resultar que otros no lo sean y encierren nuestras páginas sin compasión dentro de las suyas. Pero existe un método completamente automático que se cargará todos los frames que encierren tu página. Tan solo incluye el siguiente `script`:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- Se esconde el código a navegadores sin JS
  if (window != window.top)
    top.location.href = location.href;
// -->
</SCRIPT>
```

Colocando este código en la cabecera de tu página HTML evitarás que tu página quede encerrada en los frames de otros.

URL: http://asp.programacion.net/taller/paginacion_asp.php

Autor: Alejandro Zárate

Email: alejandrojz2@hotmail.com

Paginación entre registros

El siguiente artículo describe como realizar la paginación de un conjunto de registros. Se muestran las técnicas para :

- Realizar la paginación en conjuntos de x registros
- Construir la tabla con hipervínculos para navegar entre las páginas de resultados
- Pasar los parámetros necesarios entre las distintas páginas

Es recomendable tener conocimientos básicos sobre acceso a base de datos desde asp y paso de parámetros entre páginas.

1. Estableciendo la conexión y el conjunto de registros

El primer paso, si Ud. ya está familiarizado con el acceso a datos, no presenta nada nuevo, creamos una conexión y definimos la cadena de consulta de la manera habitual. En este ejemplo utilizaremos una tabla llamada "Libros" de una Base de Datos "Editorial" a la que accedemos con el origen de datos "dsnedit".

```
<%  
'----- conectar  
Set conn = Server.CreateObject("ADODB.Connection")  
conn.open "dsnedit", "ODBC;DATABASE=Editorial;UID=;PWD=;DSN=dsnedit"  
  
'----- definir cadena sql  
sql = "SELECT * FROM libros"  
%>
```

2. Definiendo la paginación

Ahora si vamos a dar los primeros pasos en la paginación. El proceso consiste básicamente en establecer las propiedades que determinarán la cantidad de registros a mostrar y la página actual.

Definimos una variable con la cantidad de registros por página

```
<%  
cantidadregistros=5  
%>
```

A continuación vamos a determinar que página hay que mostrar y asignamos ese valor a una variable. Aquí aparece el primer parámetro que debemos pasar entre páginas, "paginaactual". Más adelante veremos como se pasa este parámetro.

```
<%  
if request.querystring("paginaactual")<>" " then  
    mostrarpagina=request.querystring("paginaactual")  
else  
    mostrarpagina=1  
end if  
%>
```

... y es el momento de abrir el recordset

```
<%  
set rs=Server.CreateObject("ADODB.Recordset")  
rs.cachesize=60  
rs.open sql, conn, 3, 3  
%>
```

Ya estamos en condiciones de realizar la paginación,

```
<%  
rs.pagesize=cantidadregistros  
maxpagina=cint(rs.pagecount)  
maxregistros=cint(rs.pagesize)  
rs.absolutePage=mostrarpagina  
contreg=0  
%>
```

Veamos en detalle que hemos hecho. Hemos establecido la propiedad pagesize del objeto recordset en 5 ("cantidadregistro"). Guardamos la cantidad de paginas y su tamaño en dos variables. Con absolutePage indicamos que página debe mostrarse. Por último inicializamos un contador que vamos a usar al mostrar los registros.

3. Mostrando las páginas

Llegó el momento de mostrar el resultado. En primer lugar vamos a determinar los valores "desde" y "hasta".

```
<%  
if int(mostrarpagina)<>int(maxpagina) then      '-- Si no es la última página.  
    hasta=mostrarpagina*5  
    desde=hasta-4  
else  
    hasta=rs.recordcount  
    desde=(maxpagina*5)-4  
end if  
%>
```

Si en lugar de cinco registros quiere mostrar diez, sólo cambie los "5" por "10" y los "4" por "9".

Mostramos los valores recién calculados...

```
<%
response.write "Resultados : "
response.write desde & " a " & hasta & " de " & rs.recordcount & " encontrados."
%>
```

Y construimos el ciclo que muestra los registros,

```
<%
do while not rs.eof and contreg < maxregistros
    response.write rs.fields("campo") & "<br>"
    rs.movenext
    contreg=contreg+1
loop
%>
```

Observe que además de controlar la propiedad eof, como se hace habitualmente, debemos verificar la cantidad de registros mostrados.

4. Construir los hipervínculos para moverse entre las páginas

Para terminar, sólo resta construir la lista de páginas y opcionalmente los hipervínculos "Anterior" y "Siguiete". Al terminar debemos tener algo como,

[<<Anterior] 01 02 03 04 [Siguiete>>]

Los hipervínculos de esta lista llaman, obviamente, a la misma página en la que estamos trabajando, por lo que podemos referirnos a la misma con una variable (lo que facilita la reutilización del código)

La variable "cero" se la agregaremos a las páginas 1 a 9. Si hay más páginas asignamos a "cero" una cadena vacía.

A continuación, armamos la lista de hipervínculos. En primer lugar, si no estamos en la primera página guardamos en una variable el link "Anterior". En segundo lugar, guardamos en la variable la lista de páginas con un ciclo For...Next. Finalmente, si no estamos en la última página agregamos a la variable el link "Siguiete".

```
<%
cero=""
nombrescript=request.servervariables("script_name")
if int(mostrarpagina)<>1 then      '--si no es la primer página mostrar botón atrás
    ref="<a href='" & nombrescript & "?buscar=" & _
        Server.Urlencode(request.querystring("buscar"))
        ref=ref & "&paginaactual=" & mostrarpagina-1
        ref=ref & "'>[<< Anterior]</a>&nbsp;";
    end if
for contador=1 to maxpagina
```



```
If contador>9 then
    cero=""
end if

ref=ref & "&nbsp;<a href='" & nombrescript
ref=ref & "?buscar=" & Server.Urlencode(request.querystring("buscar"))
ref=ref & "&paginaactual=" & contador

if int(contador)=int(mostrarpagina) then      '-- pone en negrita pagina actual
    ref=ref & "'><strong>" & cero & contador & "</strong></a>"
else
    ref=ref & "'>" & cero & contador & "</a>"
end if

next

if int(mostrarpagina)<>int(maxpagina) then      '-- si no es la última página
                                                '    mostrar botón siguiente

    ref=ref & "&nbsp;<a href='" & nombrescript & "?buscar="
    ref=ref & Server.Urlencode(request.querystring("buscar"))
    ref=ref & "&paginaactual=" & mostrarpagina+1
    ref=ref & "'>[Siguiente >>]</a>"

end if

response.write ref
%>
```

Si observa detenidamente el código, se dará cuenta de que he incluido un parámetro "buscar". Al principio, cuando construí la consulta sql, simplemente accedí a una tabla entera, por lo que en este ejemplo no haría falta ese parámetro. No obstante, es muy probable que Ud. quiera paginar los resultados de una búsqueda. En ese caso, es importante que recuerde pasar entre las distintas páginas de resultados todos los parámetros necesarios.

5. Conclusión

En este artículo hemos visto un ejemplo muy simple de como paginar los resultados de una búsqueda. Para reducir al mínimo el código no he incluido verificaciones sobre si la tabla está vacía y otros detalles que Ud. deberá tener en cuenta en sus páginas. La paginación es casi siempre recomendable a mi criterio, tanto desde lo funcional como desde lo estético, y no requiere como habrá podido comprobar, ningún esfuerzo excepcional de programación.

Conceptos básicos de acceso a bases de datos

URL: http://asp.programacion.net/taller/basededatos_esp.php

Autor: Alex Morales Moliner

Email: alexmm@iname.com

El acceso a bases de datos es uno de los recursos más utilizados en las páginas ASP, la facilidad con la que puede crear un sitio web dinámico queda patente en este taller en el que aprenderá los conceptos básicos sobre conexión a bases de datos.

1. Introducción

Las páginas ASP con acceso a datos permiten interactuar con la información de una base de datos ya sea para obtener información y mostrarla al usuario o bien para actualizar su contenido.

Son muchas las aplicaciones de este concepto en los sistemas de información actuales por ejemplo, una empresa que vende sus artículos por Internet debe disponer de páginas en las que se visualicen los datos de sus productos, disponibilidad, precio, etc. y almacenar los pedidos de sus clientes. Al tratarse de información en continua actualización la presencia de una base de datos y su consulta dinámica se hacen indispensables.

Para conectarse a una base de datos, las páginas ASP utilizan la tecnología **ADO (ActiveX Data Objects)** y pueden accederse a sistemas de gestión de bases de datos compatibles con ODBC (entre otras SQL Server, Access, Informix o Oracle.)

2. Conexión

Pueden utilizarse dos sistemas de conexión a Base de Datos:

- Mediante DSN
- Sin DSN

DSN

Este sistema consiste en definir un identificador de la conexión mediante el driver ODBC accesible desde el Panel de Control. Posteriormente, desde las páginas ASP, se practica el

acceso mediante un string de conexión que incluye el identificador antes mencionado.

Para crear un DSN en Windows, haz clic en el botón Inicio selecciona la opción Panel de Control del menú Configuración. En la ventana del Panel de Control selecciona **Fuentes de Datos ODBC** y accede a la pestaña DSN de Sistema. Selecciona la base de datos que quieres añadir y define un nombre a la conexión y la localización física de la base de datos

Sin DSN

Este sistema requiere almacenar directamente el archivo de la BD (habitualmente de Access) en un directorio del servidor. De este modo, en la conexión se utilizará un String un poco más complejo ya que deben identificarse tanto el driver como el directorio físico completo de la base de datos.

Estos son los 3 pasos para realizar la conexión

- Crear el objeto para conectarse a la Base de datos mediante la instrucción **Server.CreateObject("ADODB.Connection")**
- Definir la conexión (con/sin DNS), mediante la instrucción **objConn.ConnectionString**
- Abrir la conexión mediante la instrucción: **objConn.Open**

```
<%  
    Dim objConn  
  
    'Creación del objeto que realiza la conexión a la base de datos  
    Set objConn = Server.CreateObject("ADODB.Connection")  
  
    'Proporcionar al objeto la información correspondiente a la conexión ODBC a utilizar  
    'Mediante DNS  
    objConn.ConnectionString = "DSN=bdProgramacion"  
    'Sin DNS  
    objConn.ConnectionString = "DBQ=C:\mis documentos\program.mdb;DRIVER={MS Access  
    (*.mdb)}"  
  
    'Abrir la conexión  
    objConn.Open  
    %>
```

3. El lenguaje SQL

Mediante el lenguaje SQL (Structured Query Language) puedes interactuar con los motores de base de datos relacionales para obtener y modificar la información almacenada en la base de datos.

La información en una base de datos se almacena en tablas que a su vez se componen de columnas o campos. Para realizar una consulta a una tabla deberá utilizar la instrucción **SELECT**. En una consulta básica en la cláusula SELECT se indican las columnas a visualizar, en la cláusula FROM, la tabla de la cual obtener los datos por último en la cláusula WHERE se indican las restricciones a aplicar.

```
SELECT * FROM clientes WHERE poblacion="Sevilla"  
SELECT nombre FROM Productos WHERE precio > 2000
```

Las operaciones de insertar, actualizar y borrar información de la base de datos también se realizan mediante instrucciones SQL concretamente utilizaremos **INSERT**, **UPDATE** y **DELETE** respectivamente.

Por ejemplo, la sentencia UPDATE se compone de la clausula SET para indicar el nuevo valor de los campos y WHERE para indicar los registros sobre los que se desea practicar la actualización.

```
INSERT INTO Productos (nombre, precio, distribuidor)  
VALUES ("Auriculares", 1730, "batech")
```

```
UPDATE Productos  
SET precio = precio + (0,02 * precio)  
WHERE distribuidor="ACME"
```

```
DELETE FROM Productos  
WHERE distribuidor="ACME"
```

4. Acceso a los datos

A continuación, practicaré las instrucciones para que desde la página ASP pueda acceder a la información de la base de datos y mostrarla en la ventana del navegador del usuario

Suponga que tiene una tabla de Productos con las siguientes columnas: nombre, precio y distribuidor.

En la página ASP a crear visualizaré aquellos productos cuyo precio sea inferior a los 30 Euros y para realizar esta consulta utilizaré el objeto **ADODB.Recordset**

```
<%  
Dim objConn  
Dim objRS  
Dim strSQL  
  
Set objConn = Server.CreateObject("ADODB.Connection")  
objConn.ConnectionString = "DSN=gestion"  
  
'Iniciación del objeto Recordset  
Set objRS = Server.CreateObject("ADODB.Recordset")  
  
' Construcción de la consulta mediante SQL  
strSQL = "SELECT nombre, precio FROM productos WHERE precio < 30"  
  
' ejecución de la consulta  
objRS.Open strSQL, objConn  
>%>
```

Como resulta de la anterior ejecución, el servidor obtiene un conjunto de registros procedentes de la base de datos y posiciona el cursor en el primero de ellos. El objeto recordset proporciona mecanismos para acceder a los diferentes campos y mover el cursor entre los diferentes registros. Puede conocer cómo mediante el siguiente código

```
<table>
<tr><td>NOMBRE</td><td>PRECIO</td></tr>
<%
'Bucle hasta encontrar EOF (final del cursor)
Do Until objRS.EOF = True
    Response.Write "<tr>"
    'objRS("nombre") y objRS("precio") contienen los valores
    'de los campos nombre y precio de la Base de datos
    Response.Write "<td>Nombre= " & objRS("nombre") & "</td>"
    Response.Write "<td>Precio = " & objRS("precio") & "</td>"
    Response.Write "</td></tr>"

    'Mover el cursor al siguiente registro.
    'Cuando ya esté en el último registro EOF valdrá cierto.
    objRS.MoveNext
Loop
%>
</table>
```

5. Cerrar conexiones

Es una práctica recomendable cerrar y borrar los objetos recordset y de conexión creados en la página ASP. Para ello se utiliza el código indicado a continuación:

```
<%
'Cerraar el objeto Recordset
objRS.Close

'Eliminar el objeto Recordset
Set objRS = Nothing

'Cerrar el objeto de la conexión
objConn.Close

'Eliminar el objeto de la conexión
Set objConn = Nothing
%>
```

6. ADOVBS.INC

ADOVBS.INC es un archivo que se incluye con el servidor Internet Information Server y en el que se definen todas aquellas constantes utilizadas con objetos del ADO.

La inclusión de archivos es muy aconsejable para realizar código legible y reutilizable dos características imprescindibles en páginas de acceso a datos. Para ello se utiliza la instrucción include virtual que pertenece al juego de instrucciones del protocolo HTTP y cuya sintaxis es la siguiente:

```
<%  
  <!-- #include virtual="/adovbs.inc" -->  
%>
```

En este caso deberá tener almacenado el archivo ADOVBS.INC en la raíz del directorio de su aplicación.

Envío de correo utilizando CDONTS

URL: <http://asp.programacion.net/taller/cdons.php>

Autor: Alex Morales Moliner

Email: alexmm@iname.com

En este taller practicará el envío de mensajes de correo desde una página ASP. Para ello utilizará el objeto **CDONTS** que se incluye con el Sistema operativo **NT versión 4.0**.

1. Descripción del objeto CDONTS

CDONTS (Collaboration Data Objects for NT) es un componente que permite enviar mensajes de correo electrónico desde las páginas ASP.

Como su nombre indica está disponible en servidores NT y en particular a partir de la versión 4.0, ya que para versiones anteriores se utilizaba el componente SendMail.

Entre algunas de sus características más importantes está la posibilidad de enviar mensajes en formato de texto o bien HTML.

Para utilizar CDO se requiere que esté instalado y operativo un servicio **SMTP** (Simple Mail Transfer Protocol) ya sea en el propio servidor o en otro servidor de la red local. SMTP es instalado por defecto con IIS 4.0 y puede acceder a sus propiedades desde la consola de administración de IIS.

2. Instrucciones necesarias en la creación y envío de mensajes

Creación del objeto

```
Set mailobj = Server.CreateObject("CDONTS.NewMail")
```

Formato del mensaje (texto o HTML)

```
mailobj.mailFormat = 0  
mailobj.bodyFormat = 0
```

Propiedades del correo

```
mailobj.from = "amorales@iname.com"
mailobj.to = "jbravo@retemail.es"
mailobj.subject = "saludos"
mailobj.body = "Hola, este es un mensaje enviado desde una página ASP"
```

Acción de enviar el mensaje (deben haberse inicializado las anteriores variables)

```
mailobj.send
```

Acción de enviar el mensaje

```
myCDO.Send "amorales@iname.com", "jbravo@retemail.es", "Saludos", _
"Hola, este es un mensaje enviado desde una página ASP"
```

Otras propiedades

Entre otras propiedades interesantes que pueden tener los mensajes enviados con CDONTS destaca el parámetro **importancia, ficheros adjuntos, o codificación MIME del mensaje**.

En el caso de adjuntar ficheros (attachment), éstos deben estar localizados en unidades accesibles por el servidor y no por el cliente o navegador).

3. Ejemplo

Página de construcción del mensaje

```
<HTML>
<BODY>
<H1>Creación del mensaje</H1>
<FORM ACTION="enviar.asp" METHOD="POST">
<TABLE>
<TR>
<TD ALIGN="RIGHT">Para:</TD>
<TD><INPUT TYPE="TEXT" SIZE="30" NAME="destinatario"></TD>
</TR>
<TR>
<TD ALIGN="RIGHT">De:</TD>
<TD><INPUT TYPE="TEXT" SIZE="30" NAME="remitente"></TD>
</TR>
<TR>
<TD ALIGN="RIGHT" NOWRAP>Asunto:</TD>
<TD ALIGN="LEFT">
<INPUT TYPE="TEXT" SIZE="67" NAME="asunto">
</TD>
</TR>
</TABLE>
<TABLE>
<TR>
<TD VALIGN="TOP" ALIGN="RIGHT" NOWRAP>Texto del mensaje:</TD>
<TD ALIGN="LEFT" NOWRAP>
<TEXTAREA WRAP="OFF" ROWS="7" COLS="50" NAME="cuerpo"></TEXTAREA>
</TD>
</TR>
</TABLE>
<INPUT TYPE="RESET" NAME="Cancelar" VALUE="Borrar"> &nbsp;
<INPUT TYPE="SUBMIT" NAME="Enviar" VALUE="Enviar">
</FORM>
</BODY>
</HTML>
```




```
<%
    Set mailobj = Server.CreateObject("CDONTS.NewMail")
    mailobj.mailFormat = 0
    mailobj.bodyFormat = 0
    mailobj.from = request.form("remitente")
    mailobj.to = request.form("destinatario")
    mailobj.subject = request.form("asunto")
    mailobj.body = request.form("cuerpo")
    mailobj.send
%>
```

4. Otras utilidades del objeto CDONTS

Si bien el uso más extendido del objeto CDONTS es el envío de correo, también puede utilizarse para visualizar en páginas ASP los **mensajes recibidos** por el servidor de correo SMTP de nuestro servidor NT.

El siguiente código ilustra como leer los mensajes de la carpeta "Bandeja de entrada" de la cuenta del usuario "Administrador"

```
<%
'Identificación
Set Correo = server.createobject("CDONTS.Session")
Correo.logonSMTP "administrador" , _
administrador@nombredominio.com"

'Acceder a la bandeja de entrada
Set Bandeja = Correo.inbox

'Crear un objeto que almacene los mensajes de la bandeja anterior
Set Mensajes = Bandeja.messages

'Recuperación de los mensajes
For k = 0 to Mensajes.count
```

```
Set elMensaje = Mensajes.item(k)
'Obtener el nombre del remitente
Response.write "Desde: " & elMensaje.sender & "<br>"
'Obtener el asunto del mensaje
Response.write "Asunto: " & elMensaje.subject & "<br>"
'Obtener el texto del mensaje
Response.write elMensaje.text & "<br>"
Next

'logoff
Correo.logoff
Set Correo = nothing
%>
```

Uso del objeto FileSystemObject

URL: <http://asp.programacion.net/taller/filesystemobject.php>

Autor: Alex Morales Moliner

Email: alexmm@iname.com

El componente FileSystemObject, uno de los más utilizados en las páginas ASP, facilita el acceso al sistema de archivos del servidor.

1. Introducción

Crear y leer archivos y directorios, son algunas de las operaciones más habituales que se realizan con el componente FileSystemObject, también conocido por las siglas FSO.

Es importante tener en cuenta que permite acceder al sistema de archivos del servidor pero **no al sistema de archivos del cliente**.

El siguiente código muestra el uso del componente FSO, en el que destacan las operaciones:

- Instanciación del objeto de acceso a archivos
- Abrir el archivo deseado en el modo deseado (lectura, escritura)
- Manipular/visualizar el contenido del archivo
- Cerrar los objetos utilizados

```
<%  
Const fsoLectura = 1  
Dim objFSO  
'Instanciación del objeto FSO  
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")  
  
'Abrir el archivo de texto  
Dim objTextStream  
Set objTextStream = objFSO.OpenTextFile("C:\ejemplo.txt", fsoLectura)  
  
'Visualiza en el navegador el contenido del archivo de texto  
Response.Write objTextStream.ReadAll  
  
'Cerrar e inicializar los objetos  
objTextStream.Close  
Set objTextStream = Nothing  
Set objFSO = Nothing  
%>
```

2. Algunos consejos en las operaciones de Lectura y Escritura

Como se vió en el anterior código, en el uso de FSO y en concreto para leer y escribir información en un archivo, se utiliza el objeto **TextStream** cuyo funcionamiento se profundiza en las siguientes líneas.

Antes de abrir el archivo es conveniente asegurarse de que existe, para ello puede utilizar la función **FileExists** disponible en el juego de funciones del objeto FSO.

```
<%  
  If objFSO.FileExists("c:\ejemplo.txt") then  
    'El archivo existe --> instrucciones de lectura  
  else  
    'El archivo no existe --> Mensaje de error  
  end if  
>%
```

En la operación de abrir el archivo pueden especificarse diferentes modos según se desee trabajar en el archivo para leerlo o escribir en él. En este ultimo caso, se puede indicar al objeto que cree el archivo si éste no existe. Es conveniente definir las correspondientes constantes.

- Lectura (1)
- Escritura (2)
- Añadir (8)

```
<%  
  Const fsoLectura = 1  
  Const fsoEscritura = 2  
  Dim objTextStream  
  
  'Ejemplo de acceso a un archivo para leer  
  Set objTextStream = objFSO.OpenTextFile("c:\ejemplo.txt", fsoLectura)  
  
  'Ejemplo de acceso a un archivo para escribir.  
  'El booleano "True" indica que en caso de no existir será creado  
  Set objTextStream = objFSO.OpenTextFile("C:\ejemplo.txt", fsoEscritura, True)  
>%
```

Para leer el contenido del archivo de texto deberá utilizar uno de los siguientes métodos del objeto TextStream.

- Read(x): lee el número de caracteres especificado por "x"
- ReadLine: Lee una línea entera
- ReadAll: Lee todo el archivo

Para escribir en un archivo de texto deberá utilizar uno de los siguientes métodos del objeto TextStream.

- Write(cadena): escribe la cadena en el archivo de texto
- WriteLine(cadena): como Write pero añade un retorno o caracter "fin de línea"

- WriteBlankLines(x): escribe x líneas en blanco

```
<%  
'Ejemplo de lectura  
Response.Write "El contenido del archivo es:" & objTextStream.ReadAll  
'Ejemplo de escritura  
objTextStream.WriteLine "Linea añadida por la ejecución de una página ASP"  
%>
```

Una vez finalizadas las operaciones con los objetos del FSO es importante cerrar los objetos utilizados mediante la siguiente instrucción

```
<%  
objTextStream.Close  
Set objTextStream = Nothing  
Set objFSO = Nothing  
%>
```

```
Dim objFSO  
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")  
  
Dim objTextStream  
  
const strNombreArchivo = "C:\pruebas\direcciones.txt"  
const fsoLectura = 1  
  
If objFSO.FileExists(strNombreArchivo) then  
    Set objTextStream = objFSO.OpenTextFile(strFileName, fsoLectura)  
    Response.Write "El contenido del archivo es:<br>"  
    Do while not objTextStream.EOF  
        Response.Write objTextStream.ReadLine  
    loop  
    objTextStream.Close  
    Set objTextStream = Nothing  
Else  
    Response.Write strFileName & " No se ha encontrado"  
End If  
  
Set objFSO = Nothing
```

3. FSO y los permisos

Para poder operar con la manipulación de archivos, es necesario tener en cuenta la cuestión de los permisos ya que se requiere que el usuario **IUSER_nombreservidor** creado automáticamente por el sistema al instalar IIS, pueda operar sobre el directorio al que se está accediendo.

En el caso de NT, se distinguen permisos de Lectura, Escritura o Acceso total y deberá seleccionarse el correspondiente según el tipo de operación que se desee llevar a cabo desde las páginas ASP ya que en caso contrario se produciría un error. Por ejemplo, IUSR_nombreservidor, debe tener permisos de Control total sobre un directorio en particular para borrar o mover un archivo de ese directorio.

Upload de archivos con el componente W3 Upload

URL: <http://asp.programacion.net/taller/w3upload.php>

Autor: Alejandro Zárte

Email: alejandrojz2@hotmail.com

Usando el componente de la empresa **Dimac** (puede descargar una versión de evaluación desde tech.dimac.net) construiremos una solución para permitir a sus visitantes realizar upload de sus archivos sin necesidad de utilizar FTP, ni escribir extensos códigos en ASP.

El componente **W3 Upload**, nos permite controlar que tipos de archivos pueden subirse, de que tamaño, en que carpeta guardarlos y opciones más avanzadas como guardar archivos en bases de datos.

Desarrollaremos un ejemplo con dos páginas, un formulario para seleccionar el archivo, **formulario.htm** y una página asp que realizará el proceso de upload, **upload.asp**

1. Formulario

Creemos un formulario con los siguientes campos:

- **Carpeta:** Es la carpeta de trabajo del usuario, donde se guardaran los archivos
- **Mensaje:** además de enviar el archivo, el usuario podrá enviar un mensaje que guardaremos en un archivo de texto.
- **Archivo:** desde aquí el usuario seleccionará el archivo a enviar.

```
<form method="POST" action="upload.asp" enctype="multipart/form-data">
Carpeta : <input type="text" name="carpeta" size="20" maxlength="20"><p>
Mensaje : <textarea rows="3" name="mensaje" cols="34"></textarea><p>
Archivo : <input type="file" name="archivo"><p>
<input type="submit" value="Enviar" name="Btnsubmit">
</form>
```

Quizás haya algunos elementos de este formulario que sean nuevos para Ud. Por un lado, debe indicar el tipo de codificación del formulario como **"multipart/form_data"**, lo que habilitará el envío de un archivo. En segundo lugar, aparece un campo de formulario de tipo "file", el cual mostrará un cuadro de texto y un botón "examinar".

2. Upload.asp

En el server deberemos tener una carpeta con los permisos correspondientes. Por ejemplo c:\uploads. Dentro de esa carpeta, los usuarios podrán crear sus subcarpetas de trabajo. El archivo adjunto y el mensaje enviado por el usuario se guardaran allí.

Comenzaremos creando una instancia del objeto:

```
<%  
Set upload = Server.CreateObject("W3.Upload")  
%>
```

De ahora en adelante, cuando hagamos referencia a los campos de formulario, lo haremos con el objeto upload y no con response. Esto es muy importante ya que de lo contrario perderemos el archivo enviado.

Verificaremos si existe la carpeta de trabajo, y de no ser así la crearemos. Para esto utilizaremos el objeto FileSystemObject.

```
<%  
Set fsys = CreateObject("Scripting.FileSystemObject")  
%>
```

"Limpiamos" la cadena ingresada por el usuario:

```
<%  
carpeta=upload.form("carpeta")  
carpeta=replace(carpeta, ".", " ")  
carpeta=replace(carpeta, ">", " ")  
carpeta=replace(carpeta, "<", " ")  
carpeta=replace(carpeta, "/", " ")  
carpeta=replace(carpeta, "\", " ")  
carpeta=replace(carpeta, ":", " ")  
carpeta=replace(carpeta, "?", " ")  
carpeta=replace(carpeta, "*", " ")  
carpeta=replace(carpeta, chr(34), " ")  
carpeta=replace(carpeta, "'", " ")  
carpeta=trim(carpeta)  
%>
```

Con **FolderExists** comprobamos si existe una carpeta; para crear una nueva usamos **CreateFolder**

```
<%  
if not fsys.FolderExists("c:\uploads\" & carpeta) then  
    fs.CreateFolder "c:\uploads\" & carpeta  
End if  
%>
```

Ya tenemos la carpeta de trabajo, ahora guardaremos en un archivo de texto el mensaje ingresado por el usuario. El nombre del archivo será del tipo:

mensaje_151100_031108pm.txt (esto es mensaje_fecha_hora)

```
<%
```

```

strmensaje=trim(upload.form("mensaje"))

if strmensaje<>" " then

    fecha=replace(date,"/","")
    hora=replace(time,":","")
    hora=replace(hora,".", "")
    hora=replace(hora," ", "")

    archivotexto = "c:\uploads\" & carpeta & "\mensaje_" & fecha & "_"
    archivotexto = archivotexto & hora & ".txt"

    set a = fsys.CreateTextFile(archivotexto,true)

    a.writeline(strmensaje)
    a.close
    set a=nothing

end if
%>

```

Ya no necesitamos el objeto fsys:

```

<%
set fsys = nothing
%>

```

Bueno, al fin empezamos a trabajar con nuestro objeto upload. En primer lugar, vamos a filtrar archivos que no queremos que los usuarios envíen. En este ejemplo, no se pueden enviar archivos de mas de 4 mb. ni aquellos con extensión "exe" o "vbs".

```

<%
Set elarchivo = upload.Form("archivo")

if elarchivo.size>4194000 then 'tamaño en bytes
    response.write "El archivo es demasiado grande para ser enviado.<p>"
    response.write "<a href='javascript:history.back()'>Regresar</a>"
    exit sub
end if

if instr(elarchivo.filename, ".")<>0 then
    extension=mid(elarchivo.filename, instr(elarchivo.filename, "."))
    if trim(extension)=".exe" or trim(extension)=".vbs" then
        response.write "Imposible enviar ese tipo de archivo.<p>"
        response.write "<a href='javascript:history.back()'>Regresar</a>"
        exit sub
    end if
end if
%>

```

Como podrá observar, utilizo "exit sub" para forzar la salida de la sub que se está ejecutando. Asumo por lo tanto que Ud. trabajará con Subs al construir la página asp (y si no es así, es una buena oportunidad para empezar a hacerlo)

Solo nos resta guardar el archivo en la carpeta de trabajo con el método **SaveToFile**

```

<%
if elarchivo.IsFile then
    strsave="c:\produccion\" & carpeta & "\" & elarchivo.filename
    elarchivo.SaveToFile(strsave)
end if

response.write "Se envio el archivo : " & elarchivo.filename

```



```
response.write "<br>Tamaño : " & elarchivo.size  
%>
```

3. Mejorando el proceso

Hay varias cosas que Ud. puede hacer para mejorar la solución que construimos:

- Combinar el componente upload con el W3 JMail para construir una completa solución de correo con attachments.
- Habilitar el envío de múltiples archivos (recorriendo la colección form)
- Puede evitar que los archivos se sobrescriban, cuando el cliente envía un archivo con un nombre ya existente. Solo debe cambiar el método **SaveToFile** por **SaveAsUniqueFile**. Así, por ejemplo, se guardará un archivo como file(1).txt si ya existe file.txt.

El upload, como podrá apreciar, es un proceso relativamente sencillo, y en general no será muy distinto si Ud. decide usar otros componentes. Las propiedades y los métodos descriptos, existen para la mayoría de los componentes disponibles y puede ahorrar con ellos unas cuantas horas de programación.

Desarrollo de un libro de visitas usando XML y ASP

URL: <http://asp.programacion.net/taller/wmlservlet.php>

Autor: Alfredo Reino Romero

Email: alf@ibium.com

Artículo cedido por [La página de Alfredo Reino](#).

1. El fichero de datos

El objeto de este mini-tutorial es desarrollar una aplicación práctica sencilla utilizando XML (eXtended Mark-up Language) y ASP (Active Server Pages). Para saber un poco más de XML, visita la "[Introducción a XML en castellano](#)" o cualquiera de las miles de páginas que existen en la red sobre el tema.

Para almacenar las firmas de los visitantes a nuestro website vamos a utilizar un documento XML de la siguiente forma:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<guestbook>
  <item>
    <author>Andres</author>
    <email>andres@correo.com</email>
    <datetime order="200005161603">16-5-2000 (16:03)</datetime>
    <ip>197.50.123.14</ip>
    <text>Hola, me gusta tu tutorial</texto>
  </item>
  <item>
    <author>Juan</author>
    <email>juan@email.es</email>
    <datetime order="200005171413">17-5-2000 (14:13)</datetime>
    <ip>195.2.170.174</ip>
    <text>Saludos desde Albacete</text>
  </item>
</guestbook>
```

Este documento XML contiene un elemento **guestbook**, que a su vez contiene elementos **item**. Un DTD para este documento sería el siguiente;

```
<!ELEMENT guestbook (item*)>
<!ELEMENT item (author, email, datetime, ip, text)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT ip (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT datetime (#PCDATA)>
  <!ATTLIST datetime order CDATA #REQUIRED>
```

La razón de almacenar la fecha y hora de dos maneras diferentes en el elemento **datetime** es tener un formato para visualización (dd-mm-aaaa hh:mm) y otro para ordenación (yyyymmddhhmm) sin tener que hacer cambios de formato de fecha.

2. La transformación a HTML

La transformación de XML a HTML la realizamos en el servidor, para asegurarnos una compatibilidad 100% con cualquier navegador web, utilizando el parser **MSXML** de Microsoft.

El documento XSLT que va a realizar la transformación es el siguiente (utilizando la implementación inicial de Microsoft para XSL)

```
<?xml version='1.0' encoding='UTF-7'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">

<xsl:for-each select="guestbook/item" order-by="-datetime/@order">

  <P><B><xsl:value-of select="author"/></B><BR/>

  <A>
<xsl:attribute name="href">mailto:<xsl:value-of select="email"/>
</xsl:attribute>
<xsl:value-of select="email"/>
</A>
<BR/>

  <xsl:value-of select="text"/>
  <I><xsl:value-of select="datetime"/></I>
</xsl:for-each>

</xsl:template>
</xsl:stylesheet>
```

Es fácil ver que lo que hacemos es iterar a través de todos los elementos **item** contenidos en **guestbook**, y presentarlos como un documento HTML.

3. El código ASP

Por último, necesitamos dos rutinas escritas en VBScript (o JScript) para realizar la transformación XML a HTML, y para añadir elementos **item** al documento XML a partir de un formulario en HTML.

El código para realizar la transformación es muy simple usando el parser de Microsoft.

```
xmlDoc = "data.xml"
xslDoc = "gbook.xsl"

Set xmlObj = CreateObject("Microsoft.XMLDOM")
xmlObj.Async = False
xmlObj.Load(Server.MapPath(xmlDoc))

Set xslObj = CreateObject("Microsoft.XMLDOM")
xslObj.Async = False
xslObj.Load(Server.MapPath(xslDoc))

Response.Write(xmlObj.transformNode(xslObj))
Set xmlObj = nothing
```

```
Set xslObj = nothing
```

El código para añadir un elemento **item** al documento XML, se basa en crear el nodo **item** en una cadena de texto, cargar la cadena de texto en una instancia del parser, y utilizar el método **appendChild** del DOM para añadirlo al documento XML que almacena nuestros datos.

```
stringNuevo = vbCrLf & "<item>" & vbCrLf & _
    "<author>" & nombre & "</author>" & vbCrLf & _
    "<email>" & email & "</email>" & vbCrLf & _
    "<datetime order="" & orden & "">" & fecha & "</datetime>" & vbCrLf & _
    "<ip>" & ip & "</ip>" & vbCrLf & _
    "<text>" & text & "</text>" & vbCrLf & _
    "</item>" & vbCrLf & vbCrLf

Set xmlObj = CreateObject("Microsoft.XMLDOM")
xmlObj.Async = False
xmlObj.Load(Server.MapPath("data.xml"))

Set xmlObj2 = CreateObject("Microsoft.XMLDOM")
xmlObj2.Async = False
xmlObj2.LoadXML(stringNuevo)

Set root = xmlObj.documentElement
Set root2 = xmlObj2.documentElement

root.appendChild(root2)
xmlObj.Save(Server.MapPath("data.xml"))

Set xmlObj = nothing
Set xmlObj2 = nothing
```

Por fin, integrando este código en un solo fichero .ASP, y haciendo que el formulario se llame a si mismo:

```
<% If Len(Trim(nombre))=0 Then
    okay = False
    mensaje = mensaje & "El nombre es obligatorio. "
End If

If Len(Trim(texto))=0 Then
    okay = False
    mensaje = mensaje & "El texto es obligatorio. "
End If

ip = Request.ServerVariables("REMOTE_HOST")

fecha = Day(Now) & "-" & Month(Now) & "-" & Year(Now)
fecha = fecha & " (" & Hour(Now) & ":"
If Minute(Now)<10 Then fecha=fecha & "0"
fecha = fecha & Minute(Now) & ")"

orden = Year(Now)

If Month(Now)<10 Then orden=orden & "0"
orden = orden & Month(Now)

If Day(Now)<10 Then orden=orden & "0"
orden = orden & Day(Now)

If Hour(Now)<10 Then orden=orden & "0"
orden = orden & Hour(Now)

If Minute(Now)<10 Then orden=orden & "0"
orden = orden & Minute(Now)

stringNuevo = vbCrLf & "<item>" & vbCrLf & _
    "<author>" & nombre & "</author>" & vbCrLf & _
    "<email>" & email & "</email>" & vbCrLf & _
    "<datetime order="" & orden & "">" & fecha & "</datetime>" & vbCrLf & _
    "<ip>" & ip & "</ip>" & vbCrLf & _
```

```

" <texto>" & texto & "</texto>" & vbCrLf & _
"</item>" & vbCrLf & vbCrLf

If okay Then
    Set xmlObj = CreateObject("Microsoft.XMLDOM")
    Set xmlObj2 = CreateObject("Microsoft.XMLDOM")

    xmlObj.Async = False
    xmlObj.Load(Server.MapPath("data.xml"))

    xmlObj2.Async = False
    xmlObj2.LoadXML(stringNuevo)

    Set root = xmlObj.documentElement
    Set root2 = xmlObj2.documentElement

    root.appendChild(root2)
    xmlObj.Save(Server.MapPath("data.xml"))
    Set xmlObj = nothing
    Set xmlObj2 = nothing
    nombre = ""
    email = ""
    texto = ""
End If
Else
    nombre = ""
    email = ""
    texto = ""
End If

xmlDoc = "data.xml"
xslDoc = "gbook.xsl"

Set xmlObj = CreateObject("Microsoft.XMLDOM")
xmlObj.Async = False
xmlObj.Load(Server.MapPath(xmlDoc))

Set xslObj = CreateObject("Microsoft.XMLDOM")
xslObj.Async = False
xslObj.Load(Server.MapPath(xslDoc))

Response.Write(xmlObj.transformNode(xslObj))
Set xmlObj = nothing
Set xslObj = nothing
%>

<br>
<form action="gbook.asp" method="POST">
<input type="Hidden" name="x" value="x">
  <table width="95%" border="0" cellspacing="0" cellpadding="2"
    align="center" class="col">
    <tr>
      <td width="100">Nombre</td>
      <td>
        <input type="text" name="nombre" value="<%= nombre %>">
      </td>
    </tr>
    <tr>
      <td width="100">Email</td>
      <td>
        <input type="text" name="email" value="<%= email %>">
      </td>
    </tr>
    <tr>
      <td width="100">Texto:</td>
      <td rowspan="2">
        <textarea rows="5" name="texto"><%= texto %></textarea>
      </td>
    </tr>
    <tr>
      <td width="100" align="left" valign="bottom">
        <input type="submit" name="Submit" value=" Enviar ">
      </td>
    </tr>
  </table>
</form>
<p align="center"><font color="red"><%= mensaje %></font></p>
</body>

```



El objeto diccionario

URL: <http://asp.programacion.net/taller/dictionary.php>

Autor: Alex Morales Moliner

Email: alexmm@iname.com

1. Uso del objeto diccionario

Uno de los objetos disponibles de la biblioteca de recursos de ASP es el **diccionario** que permite almacenar información mediante la técnica de **clave-valor**.

En el objeto diccionario puede definir claves como si se tratase de índices de un vector. El contenido de cada posición del supuesto vector será el valor que podremos consultar y actualizar cuando precisemos.

Para crear el objeto diccionario se utiliza la siguiente instrucción

```
set objDict = createObject("Scripting.Dictionary")
```

2. Métodos

A continuación se detallan los **Métodos** del diccionario

Añadir una clave / valor en el diccionario

```
objDict.Add strClave, strValor
```

recuperar las clave

```
strClavesArray = objDict.Keys
```

recuperar los valores

```
strValoresArray = objDict.Items
```

Comprobar si existe una clave

```
objDict.Exists(strClave)
```

Eliminar una clave del diccionario

```
objDict.Remove(strClave)
```

Eliminar todos los elementos del diccionario

```
objDict.RemoveAll()
```

3. Propiedades

A continuación se detallan las **Propiedades** del diccionario

Número de elementos del objeto diccionario

```
objDict.Count
```

Recuperar el valor de una clave

```
strValor = objDict.Item(strClave)
```

Cambiar una clave

```
objDict.Key(strClave) = strClaveNueva
```

4. Ejemplo

Pongamos en práctica algunas de estas propiedades mediante el siguiente ejemplo

```
<%  
    set objConn = Server.CreateObject("ADODB.Connection")  
    objConn.open "DSN=wapbolsa"  
    strSQL = "SELECT * FROM valor"  
    set objRS = objConn.Execute(strSQL)  
  
    ' crear el objeto diccionario y añadir los valores de la BD  
    set objDict = createObject("Scripting.Dictionary")  
    do while not objRS.eof  
        strclave = objRS("idValor")  
        strvalor = objRS("Importe")  
        objDict.Add strclave, strvalor  
        objRS.movenext  
    loop  
    objRS.Close  
    set objRS = nothing  
    objConn.Close  
    set objConn = nothing  
  
    ' operaciones sobre el objeto diccionario  
    strClavesArray = objDict.Keys  
    strValoresArray = objDict.Items  
    for i = 0 to objDict.Count -1  
        response.write(strClavesArray(i) & ": " & strValoresArray(i) & "<br>")  
    next  
  
    if objDict.Exists("TELE") then  
        response.write ("Existe Tele con el valor: " & objDict.Item("TELE"))  
    else  
        response.write ("No existe Tele")  
    end if  
end if  
>%
```


Codificación de URL's

URL: <http://asp.programacion.net/taller/urlencode.php>

Autor: Alex Morales Moliner

Email: alexmm@iname.com

1. Server.URLEncode

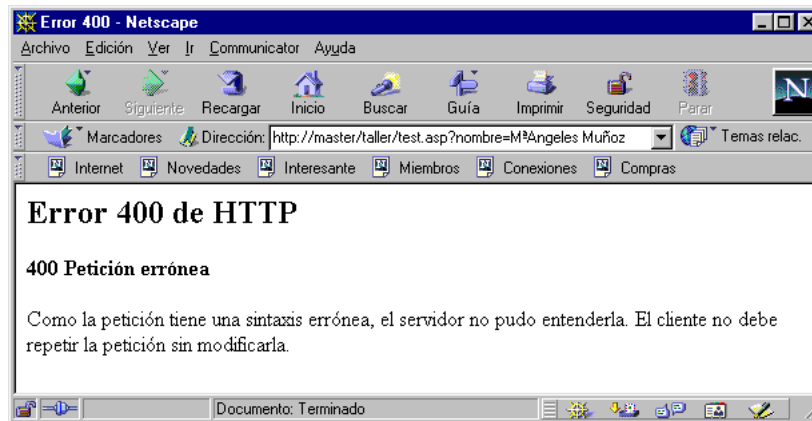
Cuando utilices páginas con paso de variables entre ellas (como parámetros en la URL) es importante que tengas en cuenta que el navegador **Netscape** no será capaz de interpretar caracteres especiales como espacios, guiones, o la letra Ñ. Esta característica puede evitarse utilizando el método **URLEncode** del objeto **server** mediante la cual se puede formatear cualquier cadena con caracteres de escape que serán correctamente procesados por los dos navegadores más conocidos del mercado.

Por citar un ejemplo el valor "MªAngeles Muñoz" se convertiría -utilizando URLEncode- en "M%AAngeles+Mu%F1oz"

```
<html>
<head>
  <title>Datos personales</title>
</head>
<body>
  Enlace a la <a href="test.asp?nombre=MªAngeles Muñoz">página de pruebas</a>
</body>
</html>
```

```
<html>
<head>
  <title>Datos personales</title>
</head>
<body>
  El nombre recibido es: <% response.write request.querystring("nombre") %>
</body>
</html>
```

Si ejecutas el enlace al que apunta esta página, verás que con Netscape se produce un error de tipo 400 Petición errónea



La sentencia correcta debería haber sido:

```
Enlace a la <a href="test.asp?nombre=<%=server.URLEncode( "MªAngeles Muñoz" )%>">página de pruebas</a>
```

Observa en la siguiente imagen la diferencia en la URL respecto a la imagen anterior (incorrecta)



2. Codificación de caracteres en formato HTML

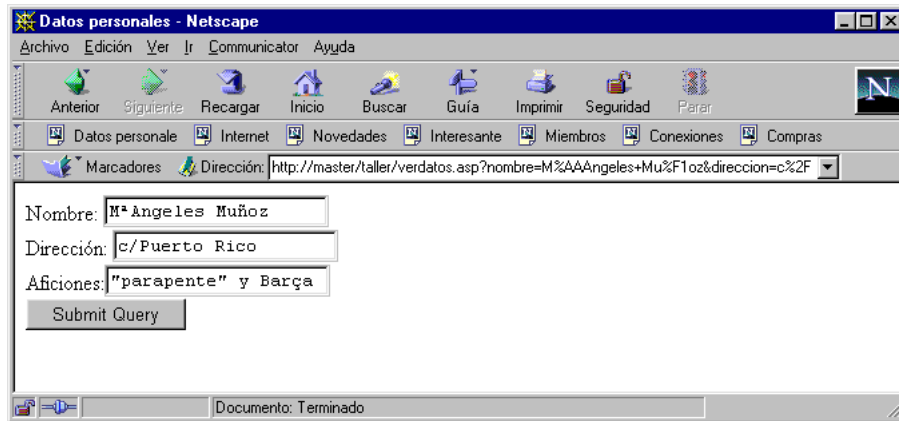
El concepto inverso al explicado anteriormente también puede ser necesario aplicarlo en los desarrollos Web. Para ello utilizaremos el método **HTMLEncode** del objeto **Server** mediante el cual podremos mostrar una cadena con los caracteres de escape que se utilizan en HTML.

Un ejemplo de la utilidad de **HTMLEncode**, lo tenemos cuando se desea mostrar un texto recibido por **querystring** como valor de un objeto de un formulario HTML.

Sigue el siguiente ejemplo para ponerlo en práctica:

```
<html>
<head>
    <title>Datos personales</title>
</head>
<body>
<form name="form1" action="verDatos.asp" method="get">
    Nombre: <input type="text" name="nombre" value=
        "<%=Server.URLEncode(request.querystring("nombre"))%>"> <br>
    Dirección: <input type="text" name="direccion" value=
        "<%=Server.URLEncode(request.querystring("direccion"))%>"> <br>
    Aficiones: <input type="text" name="aficiones" value=
        "<%=Server.URLEncode(request.querystring("aficiones"))%>"> <br>
```

```
<input type=submit>
</form>
</body>
</html>
```



En el ejemplo anterior, los parametros:

```
nombre=MªAngeles Muñoz
direccion=c/Puerto Rico
aficiones="parapente" y Barça
```

serían convertidos mediante HTMLEncode en

```
nombre=M&#170;Angeles Mu&#241;oz
direccion=c/Puerto Rico
aficiones=&quot;parapente&quot; y Bar&#231;a
```

Uso de Formularios

URL: http://asp.programacion.net/taller/forms_esp.php

Autor: Alex Morales Moliner

Email: alexmm@iname.com

En este taller practicará diversos sistemas para recuperar la información de un formulario con elementos texto, select o boton radio.

1. Declaración del formulario

En primer lugar debe disponer de una página en la que declarará los elementos del formulario. Cree un archivo de nombre 'Datos.htm' e introduzca el siguiente código:

```
<html>
<head>
  <title>Datos personales</title>
</head>
<body>
<form name="form1" action="ProcesarDatos.asp" method="post">
  Nombre: <input type="text" name="nombre"> <br>
  Sexo:   <input type="radio" name="sexo" value="H">Hombre
         <input type="radio" name="sexo" value="M">Mujer <br>
  Edad:  <select name="edad">
         <option selected value="">menos de 24 años</option>
         <option value="">entre 24 y 40 años</option>
         <option value="">entre 40 y 55 años</option>
         <option value="">más de 55 años</option>
       </select>
  <input type="submit">
</form>
</body>
</html>
```

Un formulario de HTML lo componen un elemento `<form>` y diversos elementos contenedores de datos como campos de texto, botones radio, listas seleccionables o campos ocultos.

Entre los atributos del tag form destacan action y method.

- **Action** permite especificar el receptor de los datos del formulario cuando el usuario pulsa el botón de enviar -típicamente un CGI, servlet o como en este caso, una página ASP-.
- Por su parte, el atributo **method** especifica el modo en que los datos son enviados ya sea mediante la URL (post) o en variables ocultas (get). Cómo verá más adelante,

según el método utilizado, se utilizarán también diferentes instrucciones ASP para recoger el valor de los campos del formulario.



Con el método **Post**, no se visualizarán los parámetros en la ventana de dirección del navegador. Sin embargo, con el método **Get**, como puede ver en la captura anterior el nombre y el valor de las distintos objetos aparece en la URL de la página destino del formulario unidos por el carácter "&" del modo siguiente:

Nombre_pagina.asp?variable1=valor1&variable2=valor2&variable3=valor3

2. Recepción de las variables

En la segunda parte de esta práctica, debe crear la página receptora del formulario (ProcesarDatos.asp). En ella leerá los contenidos de los objetos del formulario "nombre", "sexo" y "edad" que hayan sido introducidos por el usuario.

```
<html>
<head>
  <title>Valores introducidos</title>
</head>
<body>
  Ha introducido:<br>
  <%= request.form("nombre")%> en el campo nombre<br>
  <%= request.form("sexo")%> en el campo Sexo<br>
  <%= request.form("edad")%> en el campo Edad<br>
</body>
</html>
```

Post y Get

Si el formulario era transmitido mediante método POST, debe utilizar la sentencia **Request.form("nombre_variable")** para recuperar el valor de los distintos objetos del formulario (tal y como se refleja en el ejemplo anterior). Por el contrario, si utilizó el método GET, debe recurrir a la sentencia **Request.querystring("nombre_variable")** para obtener los resultados.

3. Tratamiento de listas

Cuando utilice formularios con objetos del tipo **<select multiple>**, en los que un mismo objeto almacena más de un valor, es interesante disponer de un método flexible para recuperar los valores seleccionados.

Por ejemplo suponga una página con el siguiente formulario, en el que puede seleccionar un

conjunto de países de la lista mediante la combinación del clic del mouse y las teclas Mayúsculas o Control.

```
<form name="form1" action="ProcesarDatos.asp" method="POST">
  <select multiple size="4" name="países">
    <option value="España">España</option>
    <option value="Argentina">Argentina</option>
    <option value="Uruguay">Uruguay</option>
    <option value="Chile">Chile</option>
    <option value="Paraguay">Paraguay</option>
    <option value="Colombia">Colombia</option>
    <option value="Cuba">Cuba</option>
    <option value="México">México</option>
    <option value="Panamá">Panamá</option>
  </select>
  <input type="submit">
</form>
```

A continuación, debe crear la página que leerá los datos del formulario. Un elemento del tipo Select como el del ejemplo devuelve en la variable **países** la lista de opciones seleccionadas separadas por el carácter ",". En el código siguiente se utiliza la función **"ListToArray"** para copiar el contenido del objeto select a un array mucho más manejable. Para finalizar se realiza un recorrido del array para mostrar los elementos que contiene.

```
<% function ListToArray(Byval Lista)
  redim Array(1)
  pos = instr(Lista, ",")
  if pos <> 0 then
    do while pos > 0
      actual = Ubound(Array)
      redim preserve Array(actual + 1)
      Array(actual) = mid(Lista, 1, pos - 1)
      Lista = trim(mid(Lista, pos+1, len(Lista) - (pos)))
      pos = instr(Lista, ",")
    loop
  end if
  actual = Ubound(Array)
  redim preserve Array(actual)
  Array(actual) = Lista
end function %>

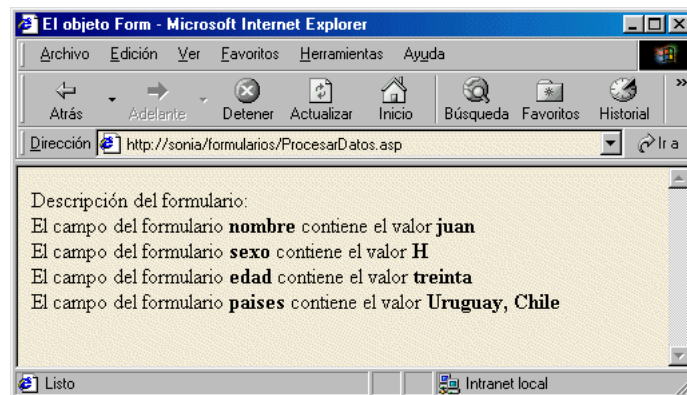
<html>
<head>
  <title>Países seleccionados</title>
</head>
<body>
  Ha seleccionado los países:<br>
  <%
    Dim Array()
    Lista = request.form("países")
    ListToArray(Lista)
    for i = 1 to Ubound(Array)
      response.write Array(i) & "<br>"
    next
  %>
</body>
</html>
```

4. El objeto Form

También puede referirse a los objetos del formulario sin conocer sus nombres mediante el objeto Request.Form. Para finalizar este artículo, pruebe el siguiente código que le permitirá

imprimir el nombre del campo del formulario y el valor que contiene.

```
<html>
<head>
    <title>El objeto Form</title>
</head>
<body>
    Descripción del formulario:<br>
    <%
        For Each x In Request.Form
            response.write "El campo del formulario <b>" & x & "</b> _
                           lcontiene el valor <b>" & Request.Form(x) & "</b><br>"
        next
    %>
</body>
</html>
```



Aspecto de la página que lee los datos de un formulario.

Evitar la cache del navegador

URL: http://asp.programacion.net/taller/cache_asp.php

Autor: Alex Morales Moliner

Email: alexmm@iname.com

En este taller conocerá como forzar la recarga de una página asp eludiendo la cache del navegador.

1. Instrucciones para evitar la Caché

El navegador (ya sea Internet Explorer o Netscape Navigator) con el que está visualizando esta página ha almacenado en su disco duro las imágenes y el texto que la componen. De esta manera cuando acceda la próxima vez su navegador accederá a su disco duro para mostrarle la página evitando el tiempo de descarga a través de Internet.

Este mecanismo, se repite mediante el proxy de una Intranet o del proveedor de acceso a Internet.

La ventaja de esta característica se convierte en desventaja en el caso de páginas dinámicas, (como ASP) que necesitan procesarse en el servidor antes de visualizarse en el navegador del cliente, para lo cual deberá desactivar esta característica.

Los siguientes métodos del objeto **Response** aseguran que una página ASP se genere siempre dinámicamente (previo proceso por el servidor) evitando la recarga de la caché del navegador (o del proxy) en caso de que se haya accedido con anterioridad.

Al indicar **CacheControl = Private**, se evita el almacenamiento en el proxy y la instrucción **Expires** permite indicar la fecha y hora en el que la página será eliminada de la caché del navegador cliente. Por ejemplo, basta con poner un cero, un número negativo o una fecha anterior a la actual.

```
<%  
Response.AddHeader "pragma", "no-cache"  
Response.CacheControl = "Private"  
  
' Selecciona una de las tres opciones siguientes  
Response.Expires = -1441  
Response.Expires = 0  
Response.ExpiresAbsolute = #1/5/2000 12:12:12#
```



```
%>
```

Creación de un contador

URL: http://asp.programacion.net/taller/contador_esp.php

Autor: Alex Morales Moliner

Email: alexmm@iname.com

Utilice un contador para mostrar el número de usuarios que han accedido a su página

1. Uso de un contador

Los contadores de visitas son ampliamente utilizados por muchos sitios de Internet ya que dan una muestra del número de visitas de una página.

Pasos previos para utilizar este contador

En primer lugar debe identificar la página ASP que contendrá el contador -habitualmente se trata de la home page de su sitio Web-

A continuación debe crear un archivo de texto que se llamará "contador.txt" en el directorio "contador" y debe escribir una línea con el número "1", para inicializar el contador.

```
<%  
    ' Creación del objeto de acceso a ficheros y del nombre del fichero  
    ' a acceder -contador.txt- del directorio contador  
    Set FileObj = Server.CreateObject("Scripting.FileSystemObject")  
    nombFichero = Server.MapPath ("/contador") & "\contador.txt"  
  
    ' Lectura del fichero contador.txt e incremento del número de visitas  
    Set Entrada= FileObj.OpenTextFile (nombFichero, 1, false )  
    Actual = Trim(Entrada.ReadLine)  
    Nuevo = Actual + 1  
  
    ' Creación del fichero contador.txt y escritura del número de visitas  
    Set Salida= FileObj.CreateTextFile (nombFichero, True)  
    Salida.WriteLine(Nuevo)  
%>  
<!-- En la siguiente línea se muestra el número de visitas actuales -->  
Número de visitas <%=Nuevo%>.
```

Uso de las variables de Servidor

URL: <http://asp.programacion.net/taller/servervariables.php>

Autor: Alex Morales Moliner

Email: alexmm@iname.com

1. Introducción

Las variables de Servidor (**ServerVariables**) almacenan información relativa al entorno de ejecución de una aplicación ASP.

La sintaxis para acceder a estas variables es mediante el objeto **Request** indicando entre paréntesis el nombre de la variable a obtener.

Una de las aplicaciones más utilizadas de estas variables es la obtención del identificador de usuario. Esta información se conoce como el **nombre de usuario** que se introduce al hacer login en un equipo de un entorno de red local.

Esta sería la sintaxis:

```
<%  
    Response.write Request.ServerVariables("LOGON USER")  
%>
```

2. Descripciones de algunas variables de servidor

A continuación se describe el significado de las diferentes variables de servidor:

AUTH_TYPE: Indica el método de autenticación que utiliza el servidor para validar a un usuario

CONTENT_TYPE: Tipos de dato del contenido

LOGON_USER: Cuenta de Windows NT con la que se ha loginado el usuario

QUERY_STRING: Cadena que sigue al signo interrogante (?) en la petición HTTP

REMOTE_ADDR: Dirección IP del equipo remoto que realiza la petición al servidor

REMOTE_HOST: Nombre del Host que realiza la petición

REQUEST_METHOD: Método utilizado en la petición (GET, HEAD, POST)

SCRIPT_MAP: Prefijo de la URL anterior a la pagina

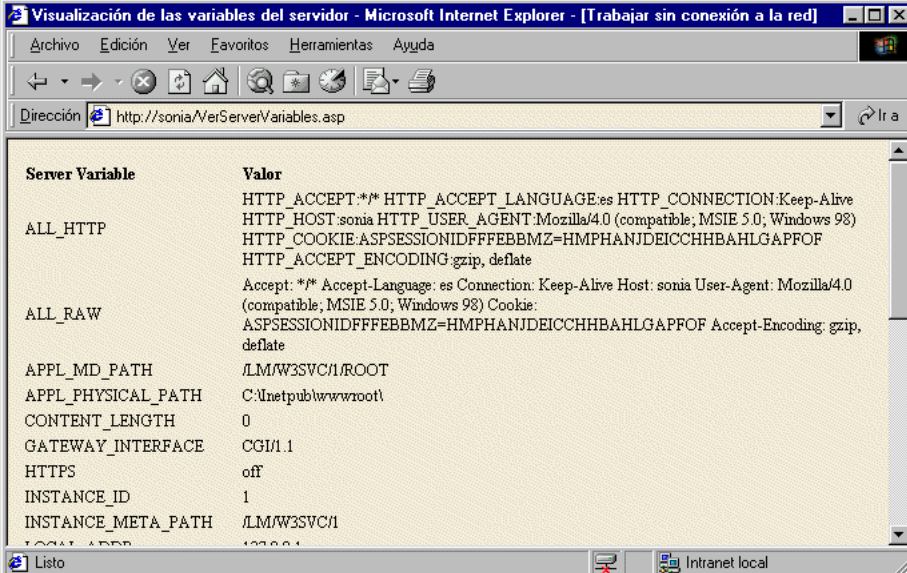
SERVER_NAME: Nombre o IP del servidor

3. Ejemplo

El siguiente ejemplo utiliza un bucle para recorrer el objeto ServerVariables visualizando el nombre y el contenido de todas ellas en una tabla de dos columnas

```
<TABLE>
<TR>
<TD><B>Server Variable</B></TD>
<TD><B>Value</B></TD>
</TR>
<% For Each name In Request.ServerVariables %>
<TR>
<TD> <%= name %> </TD>
<TD> <%= Request.ServerVariables(name) %> </TD>
</TR>
<% Next %>
</TABLE>
```

Este sería el resultado de ejecutar esta página:



Server Variable	Valor
ALL_HTTP	HTTP_ACCEPT:*/ HTTP_ACCEPT_LANGUAGE:es HTTP_CONNECTION:Keep-Alive HTTP_HOST:sonia HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.0; Windows 98) HTTP_COOKIE:ASPSESSIONIDFFFEBBMZ=HMPHANJDEICCHHBAHLGAPFOF HTTP_ACCEPT_ENCODING:gzip, deflate
ALL_RAW	Accept: */* Accept-Language: es Connection: Keep-Alive Host: sonia User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98) Cookie: ASPSESSIONIDFFFEBBMZ=HMPHANJDEICCHHBAHLGAPFOF Accept-Encoding: gzip, deflate
APPL_MD_PATH	/LM/W3SVC/1/ROOT
APPL_PHYSICAL_PATH	C:\inetpub\wwwroot\
CONTENT_LENGTH	0
GATEWAY_INTERFACE	CGI/1.1
HTTPS	off
INSTANCE_ID	1
INSTANCE_META_PATH	/LM/W3SVC/1
LOCAL_ADDR	127.0.0.1

El archivo Global.asa

URL: <http://asp.programacion.net/taller/globalasa.php>

Autor: Alex Morales Moliner

Email: alexmm@iname.com

El archivo Global.asa es procesado por cada sesión de navegador que inicia la aplicación. En este taller conocerá su funcionamiento, qué contiene y dónde debe almacenarse

1. ¿Dónde grabar el archivo Global.asa?

Uno de los errores más frecuentes en la utilización del archivo Global.asa es su ubicación física en el servidor web ya que debe estar en la raíz del Directorio Virtual IIS creado para la aplicación.

Dado un directorio del sistema de archivos del servidor en el que tiene almacenados los archivos, asp, html, imágenes y otros recursos que componen su aplicación, puede crear, mediante Internet Information Server un **Directorio Virtual** que le permitirá acceder a la su aplicación asp desde el navegador.

De este modo el directorio virtual actúa de directorio **inicial** a partir del cual cuelgan los recursos de su aplicación en forma de archivos y directorios. Por ejemplo:

Dado el directorio físico del servidor "**c:\inetpub\wwwroot\stopper**" y el Directorio Virtual de IIS "**AppStopper**"

Cuando en el navegador se escriba la dirección: "**http://nombreservidor/AppStopper**" accederá a los archivos, directorios que cuelgan de "**c:\inetpub\wwwroot\stopper**"

Según esto el archivo global.asa deberá almacenarse en el directorio físico de servidor al que hace referencia el directorio virtual. En el ejemplo anterior:

c:\inetpub\wwwroot\stopper\global.asa

2. Contenido del archivo Global.asa

El código asp que puede escribirse en el archivo global.asa debe enmarcarse en los eventos de Inicio o Fin de la Aplicación o de la Sesión.

Evento: Inicio de Aplicación

Este evento ocurre antes del inicio de una nueva sesión de un usuario

Evento: Fin de Aplicación

Este evento sucede cuando la aplicación finaliza lo que sucede cuando es servidor web es detenido.

Evento: Inicio de Sesión

Este evento ocurre antes de que el servidor Web cree el objeto Session para indicar que un nuevo usuario realiza una petición. Típicamente se especifica en esta subrutina el tiempo de inactividad antes de finalizar la sesión mediante la sentencia **Session.Timeout**

Evento: Fin de Sesión

Este evento sucede cuando la sesión finaliza o sea cuando se excede el tiempo de inactividad o cuando el usuario cierra el navegador. También puede provocarse este evento con la instrucción **Session.Abandon**

3. Ejemplo del archivo Global.asa

Los siguientes fragmentos de código corresponden a un archivo Global.asa ficticio y una página asp que imprime en pantalla el valor de las variables declaradas en el propio archivo Global.asa.

```
<script Language="VBScript" RUNAT=Server>
Sub Application_OnEnd()
End Sub

Sub Application_OnStart()
    Application("NumSession") = 0
    Application("NumVisitas") = 0
End Sub

Sub Session_OnEnd()
    Application("NumSession") = Application("NumSession") - 1
End Sub

Sub Session_OnStart()
    Application("NumSession") = Application("NumSession") + 1
    Application("NumVisitas") = Application("NumVisitas") + 1
End Sub
</script>
```

```
<html>
<head>
    <title>Página de Inicio</title>
</head>
<body>
    <h1>Bienvenido a la Aplicación Stopper</h1>
    <h2>
        <% Response.write "Eres el visitante número: & Application("NumSession") & " de " &
        Application("NumVisitas")    %>
    </h2>
</body>
```

```
</html>
```

4. Consideraciones

Suponga que en el directorio raíz de la aplicación tiene un archivo Global.asa. Como vio anteriormente cuando un usuario acceda a un archivo asp de cualquiera de los directorios virtuales (*) o físicos, el servidor, ejecutará los eventos OnStart correspondientes haciendo que las variables inicializadas en ellos sean accesibles por todos los archivos asp jerárquicamente pertenecientes a la raíz.

(*)Debe tener en cuenta que la posibilidad de tener directorios virtuales con sus respectivos archivos Global.asa anula la inicialización realizada en primer lugar perdiendo el valor de esas variables.

Es recomendable controlar el número de archivos Global.asa que utiliza una determinada Web y no olvidar las condiciones de validez de las variables creadas en él.

Redirección de una página ASP

URL: http://asp.programacion.net/taller/redireccionar_asp.php

Autor: Alex Morales Moliner

Email: alexmm@iname.com

En este taller practicará como puede interrumpir la ejecución de una página ASP para redireccionar a otra página.

1. Introducción

Para redireccionar la ejecución de una página ASP utilizará el método **Redirect** proporcionado por el objeto **Response**.

Esta técnica es útil en aquellas situaciones en las que debe mostrar en el navegador del usuario dos páginas distintas según una condición previa (tipo de usuario, hora del día, versión del navegador, etc.)

2. Ejemplo

En el siguiente ejemplo se utiliza la sentencia **Response.Redirect mi_pagina** para interrumpir la ejecución de la página y visualizar en el navegador la página identificada en la variable "mi_pagina"

```
<%  
    response.buffer = true  
    Dim mi_pagina  
  
    'Comparación ficticia para inicializar la variable mi_pagina  
    If tipo_usuario = "cliente" Then  
        mi_pagina = "clientes.htm"  
    Else  
        mi_pagina = "inicioNoClientes.htm"  
    End If  
    ' Llamada al método redirect  
    Response.Clear  
    Response.Redirect mi_pagina  
    Response.End  
%>
```


3. Observaciones

Debe tener en cuenta que el procedimiento utilizado por el servidor es enviar al cliente una cabecera HTTP indicando que el objeto ha sido movido. A continuación el navegador recibe en el location la dirección de la nueva página a visualizar (destino de la redirección).

La sentencia **response.buffer = true** sirve para indicar al servidor que la página debe almacenarse en un buffer (espacio de memoria temporal) y ser enviada al cliente cuando ha finalizado la ejecución de la misma. Este uso, evita que el navegador cargue una página que será actualizada en unos segundos lo que además resulta molesto para el usuario.

Generar WML desde un Servlet

URL: <http://java.programacion.net/taller/wmlservlet.htm>

Autor: Joaquin Bravo Montero

Email: jbravo@retemail.es

Técnicamente, el modelo de operación cliente/servidor de **WAP** es muy similar al utilizado en el WWW, y de hecho este **ha sido diseñado para que sea posible aprovechar la infraestructura tecnológica existente en la Web para el aporte de contenidos.**

Por tanto, esto significa que en el desarrollo de nuestras aplicaciones WAP podremos utilizar las aplicaciones y tecnologías que hasta el momento venimos utilizando en el desarrollo de nuestras aplicaciones Web: **CGIs, ASP, PHP, Perl, Servlets, JSP**, etc.

Sin duda alguna, **una de las más idóneas y utilizadas es la plataforma Java, y en concreto la utilización de servlets y JSP.**

En este artículo explicaremos como generar de forma dinámica nuestras páginas WML desde un servlet y veremos que sus diferencias con las generación de páginas HTML es mínima.

1. Página WML que generaremos

Este es el código de la página WML que generaremos:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
    "-//WAPFORUM//DTD WML1.1//EN"
    "http://www.wapforum.org/DTD/wml1.1.xml">

<wml>
<card id="cardpre" title="presentacion">
<p align="center"><b>Java en castellano</b></p>
<p align="center"><small>http://java.programacion.net</small></p>
<p align="center">Ejemplo generado desde un servlet</p>
</card>
</wml>
```

Que visto en el emulador WML de Nokía, presenta el siguiente aspecto:



Imagen del movil

No es el objetivo de este artículo, pero brevemente acerca del WML diremos que:

- Que es **una aplicación XML**, y como tal los documentos WML podemos editarlos, validarlos y manipularlos utilizando las herramientas que normalmente utilizamos en XML.
- Una página WML se denomina deck(baraja) y se subdivide en cards(cartas), de manera que podemos movernos por varias cards dentro de un mismo deck. En nuestro ejemplo solo tenemos un card.

2. Servlet que genera el código anterior

Este es el **código** que genera la página WML anterior

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class WMLServlet extends HttpServlet{

    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);
    }

    public void service (HttpServletRequest req,
                        HttpServletResponse res) throws
                        ServletException, IOException
    {

        PrintWriter f = res.getWriter();
        res.setContentType("text/vnd.wap.wml");
        f.println("<?xml version=\"1.0\"?>");
        f.println("<!DOCTYPE wml
                PUBLIC \"-//WAPFORUM//DTD WML1.1//EN\"
                \"http://www.wapforum.org/DTD/wml1.1.xml\">");
        f.println("<wml>");
        f.println("<card id=\"cardpre\" title=\"presentacion\">");
        f.println("<p align=\"center\"><b>Java en castellano</b></p>");
        f.println("<p align=\"center\">
                <small>http://java.programacion.net</small></p>");
        f.println("<p align=\"center\">Ejemplo generado desde un servlet</p>");
        f.println("</card>");
        f.println("</wml>");

        f.close();
    }
}
```

Como podemos observar, es **un servlet típico**. La única diferencia con uno que genere HTML es lo que tenemos que poner en el método **setContentType** del objeto **HttpServletResponse**. En este caso tenemos que indicar que lo que vamos a enviar va a ser una página WML y esto lo indicamos con la siguiente expresión:

```
res.setContentType("text/vnd.wap.wml");
```

3. Aplicaciones utilizadas en el ejemplo

Este ejemplo ha sido desarrollado y probado con las siguientes aplicaciones.

- El [emulador de Nokia](#) para visualizar la página WML generada.
- El [JSWDK-1.0.1](#) de Sun para poder ejecutar el servlet en mi máquina.
- El [parser XML](#) de IBM para verificar que el documento WML que ibamos a generar desde el servlet era correcto.

4. Direcciones de interes

Estas son algunas direcciones interesantes relacionadas con el tema:

4.1. WML y WAP

- Tutorial de WML en [WMLclub](#)
- Tutorial de WML en [WapFacil](#)
- [Lista](#) de sobre WAP en castellano

4.2. Java y Servlets

- [Java](#) en castellano
- Tutorial de [Servlets](#) en Java en castellano.

Descompresion de ficheros ZIP.

URL: <http://java.programacion.net/taller/20000630.htm>

Autor: José Antonio Pérez Martínez

Email: jperezmar@eresmas.com

1. Introducción

El tamaño de los ficheros para su almacenamiento y envío por la red es significativo. Son muchos los formatos usados para guardar de información de forma comprimida.

Se puede distinguir dos tipos de compresión de datos:

- Los que **tienen** perdidas usados en campos donde una reducción de la información no es significativa: gráficos, sonido, etc.
- Los que **no tienen perdidas** usados en campos donde la totalidad de la información debe ser recuperada.

El JDK proporciona un soporte excelente para el tratamiento de ficheros ZIP.

Para la implementación de las clases que dan soporte al formato ZIP, Java utiliza como base una librería en código nativo basada en la **ZLib**.

Las clases para el tratamiento de los ficheros en formato ZIP contemplan el uso de ficheros comprimidos con el método `deflate/inflate` y ficheros no comprimidos (simplemente almacenados).

El algoritmo `deflate` es el que permite la compresión de ficheros y está implementado en la clase `deflate` y la descompresión corre a cargo de la clase `inflate`.

La librería ZLIB fue desarrollada inicialmente como parte de los gráficos PNG. La descripción del formato `deflate` se puede encontrar en la [RFC 1951](#).

Las clases que intervienen en la compresión/descompresión de ficheros son varias aunque sólo tres de ellas nos interesan:

- `ZipFile`
- `ZipEntry`

- `ZipException`

La clase `ZipFile` se usa para abrir el fichero Zip y es por tanto el punto de partida para la extracción de ficheros.

Esta clase nos proporciona una enumeración de todas las entradas de los ficheros que están almacenados en el ZIP por medio del método `entries()`. Los objetos que devuelve la enumeración son objetos de la clase `ZipEntry`. También es posible obtener los datos concernientes a un fichero en concreto de forma directa si conocemos el nombre con el que está almacenado el fichero.. Dados estos métodos podemos buscar una entrada (`ZipEntry`) asociada a un nombre de fichero comprimido de forma directa, o recorrerlas todas las entradas de forma secuencial.

2. El descompresor

Para demostrar el uso de las clases crearemos un pequeño ejemplo consistente en una aplicación capaz de extraer los ficheros contenidos en un archivo ZIP.

El programa consta de 4 clases:

- `jZipException`: la excepción generada por la aplicación.
- `jUnzip`: clase que contiene los métodos para la extracción de ficheros.
- `toolsZip`: contienen los métodos de creación de directorios y nombres de los ficheros.
- `javaUnzip`: es la clase que contiene el main.

```
import java.io.*;
import java.util.*;

public class jZipException extends Exception {
    public jZipException(String strError) {
        super(strError);
    }
}
```

La clase `jZipException` únicamente implementa el constructor al que se le pasa la cadena con el mensaje de error.

```
import java.io.*;
import java.util.*;
import java.util.zip.*;

public class jUnzip {

    static public boolean decompressFile(String fileName, String dirBase)
        throws jZipException{
        ZipFile zf= null;
        byte buffer[] = new byte[4048];/* tamaño del buffer de descompresión*/
        int lenToWrite;
        boolean flagDirBase;
        String separator = "";
        if (dirBase == null) {
            flagDirBase = false;
            dirBase = "";
        }else {
```

```

        flagDirBase = true;
        if(!dirBase.endsWith(File.separator) && dirBase.length()>0) {
            separator = File.separator;
        }
    }
    try {
        zf = new ZipFile(fileName);
        Enumeration e = zf.entries();
        for (;e.hasMoreElements();) {
            ZipEntry z = (ZipEntry)e.nextElement();
            System.out.println(z.getName() + " -> " + dirBase + separator +
                toolsZip.getRealNameFile(z.getName(), flagDirBase));
            InputStream in = zf.getInputStream(z);
            if(flagDirBase)
                toolsZip.createDirFromNameFile(dirBase, z.getName());
            FileOutputStream fTarget = new FileOutputStream(dirBase +
                separator + toolsZip.getRealNameFile(z.getName(), flagDirBase));
            while ((lenToWrite=in.read(buffer)) != -1) {
                fTarget.write(buffer,0,lenToWrite);
            }
            fTarget.close();
        }
    }
    catch(IOException e) {
        System.out.println("[decompressFile] Exception " + e.getMessage());
        throw new jZipException("[decompressFile]" + e.getMessage());
    }
    finally{
        try {
            if ( zf != null) {
                zf.close();
            }
        }
        catch(IOException ee) {
            System.out.println("[decompressFile] Exception in close file:"
                + ee.getMessage());
            throw new jZipException("[decompressFile]" + ee.getMessage());
        }
    }
    return true;
}
}

```

La clase `jUnzip` es la que lleva el peso de la descompresión con el método `decompressFile`. Este método tiene dos parámetros:

- String `fileName`: es el nombre del fichero ZIP.
- String `dirBase`: es el path base en el que se descomprime los archivos.

La instancia de la clase `ZipFile` nos proporciona las entradas de los archivos que contiene y añade a los nombres de cada uno de los archivos comprimidos el directorio base: `dirBase`. Si no especificamos ningún directorio por defecto los descomprime en el que se ejecuta la aplicación.

En el caso de que haya algún problema se lanzará una excepción de tipo `jZipException`.

```

import java.io.*;
import java.util.*;

public class toolsZip {
    static public boolean mkdir(String namePath) throws jZipException{
        File file = new File(namePath);
        boolean ret = false;
        try {
            if (file.exists()) {
                if (file.isDirectory())
                    return true;
            }
        }
    }
}

```

```

        else
            return false;
        }
        else
            ret = file.mkdirs();
    }
    catch(SecurityException ee) {
        String str = "[mkdir] " + ee.getMessage();
        System.err.println(str);
        throw new jZipException(str);
    };
    return ret;
}
/*****
static public boolean createDirFromNameFile(String dirBase, String pathfile) {
    String path;
    String file;
    int lenpath;
    int lenfile;

    //path completo sustituyendo caracteres no adecuados

    path = getRealFileName(pathfile, true);
    file = getRealFileName(pathfile, false);
    lenpath = path.length();
    lenfile = file.length();
    if(lenpath == lenfile && dirBase.length()==0)
        return true;
    file = path.substring(0, lenpath-lenfile);
    if(file.endsWith(File.separator)) {
        path = file.substring(0, lenpath-lenfile-1);
    }
    else
        path = file;
    try {
        if(!mkdir(dirBase + path))
            return false;
    }
    catch(Exception e) {
        System.err.println("[mkdir]Exception " + e.getMessage());
        return false;
    }
    return true;
}
/*****/
static public String getRealFileName(String file, boolean full) {
    String ret = file;
    String aux = file.replace('/', File.separator.charAt(0));
    if(full)
        ret = aux;
    else {
        StringTokenizer st = new StringTokenizer(aux, File.separator.charAt(0));
        while(st.hasMoreTokens()) {
            ret = st.nextToken();
        }
    }
    return ret;
}
}

```

La clase `toolsZip` contiene todos los métodos que son de utilidad para la manipulación de ficheros y directorios:

- `mkdir` crea un directorio con el nombre especificado por el parámetro `namePath`.
- `getRealFileName` tiene una doble utilidad: por un lado coge el el nombre del fichero `file` y traduce las barra de los directorios a las propias de la plataforma de ejecución y si `full` es `false` sólo retorna el nombre del fichero, si es `true` retorna todo el `path`.
- `createDirFromNameFile` crea el directorio donde se debe grabar el fichero teniendo en cuenta el directorio base y el nombre del directorio de la aplicación.


```
import java.io.*;
import java.util.*;

public class javaUnzip {
    public static void main(String args[]) {
        String path = "";
        String file = "";
        if (args.length < 2 ){
            System.out.println("sintaxis: " +
                "javaUnzip <nombre fichero> <directorio_descompresión>");
            return;
        }
        file = args[0];
        path = args[1];
        try {
            jUnzip.decompressFile(file, path);
        }
        catch(Exception e ) {
            System.out.println("ERROR:" + e.getMessage());
        }
    }
}
```

javaUnzip contiene el código que obtiene los parámetros y llama a jUnzip.descompresFile

3. Bibliografía y referencias

- [1] Nelson, Mark, "The Data Compression Book", M&T Books, 1991
- [2] Zlib page <http://www.info-zip.org/pub/infozip/zlib/>
- [3] PkWare <http://www.pkware.com/>
- [4] Winzip <http://www.winzip.com/>
- [5] RFC 1951 <http://www.w3.org/Graphics/PNG/RFC-1951>

CREACION DE FICHEROS ZIP: Implementación un compresor de ficheros ZIP.

URL: <http://java.programacion.net/taller/javazipc.htm>

Autor: José Antonio Pérez Martínez

Email: jperezmar@eresmas.com

1. Introducción

En el artículo "DESCOMPRESION DE FICHEROS ZIP" se explicaba la manera de extraer archivos de un fichero comprimido ZIP.

En presente artículo pretende mostrar la manera de crear ficheros ZIP usando las clases java para compresión que proporciona el JDK, así como las consideraciones a tener en cuenta para un resultado correcto.

Las clases del JDK que intervienen en la compresión de ficheros son varias aunque sólo tres de ellas nos interesan:

- ZipEntry
- ZipOutputStream
- ZipException

ZipEntry es usada para representar una entrada de fichero comprimido. Esta clase nos permite incluso establecer el método de compresión del archivo a incluir.

ZipOutputStream es usado para crear un stream comprimido de salida.

ZipException representa la clase especifica en caso de error en el proceso de compresión/descompresion.

2. El compresor

Para demostrar el uso de las clases crearemos un pequeño ejemplo consistente en una aplicación capaz de crear un fichero zip en el que podremos incluir los ficheros que creamos oportuno.

El programa consta de 4 clases:

- `jZipException`: la excepción generada por la aplicación
- `zipFileFilter`: clase usada en `toolsZip` para filtrado de nombres de archivo
- `jZip`: clase que contiene los métodos para la creación de fichero ZIP
- `toolsZip`: contienen los métodos para análisis y descomposición de nombre de ficheros. Notese que aunque en el artículo de descompresión se incluye una clase con el mismo nombre, no contienen los mismos métodos.
- `javaZip`: es la clase que contiene el main

```
import java.io.*;
import java.util.*;

public class jZipException extends Exception {
    public jZipException(String strError) {
        super(strError);
    }
}
```

La clase `jZipException` únicamente implementa el constructor al que se le pasa la cadena con el mensaje de error.

```
import java.io.*;
import java.util.*;
import java.util.zip.*;

public class jZip {
    static public boolean _compressFiles(String fileNameZip, Vector filesSrc,
        Vector fileName) throws jZipException {
        FileInputStream fis = null;
        FileOutputStream fos = null;
        ZipOutputStream zos = null;
        if (filesSrc == null || filesSrc.size() == 0)
            return false;
        if (fileName == null || fileName.size() != filesSrc.size())
            //mismo vector de nombres
            fileName = filesSrc;
        try {
            byte buffer[] = new byte[4048];
            int lenToWrite;
            if (toolsZip.existFileDir(fileNameZip)) {
                toolsZip.deleteFile(fileNameZip);
            }
            fos = new FileOutputStream(fileNameZip);
            zos = new ZipOutputStream(fos);
            for (int inx = 0; inx < filesSrc.size(); inx++) {
                fis = new FileInputStream((String)filesSrc.elementAt(inx));
                ZipEntry ze = new ZipEntry((String)fileName.elementAt(inx));
                System.out.println(filesSrc.elementAt(inx) + "-->" + fileName.elementAt(inx));
                zos.putNextEntry(ze);
                while ((lenToWrite=fis.read(buffer)) != -1) {
                    zos.write(buffer,0,lenToWrite);
                }
                fis.close();
                fis = null;
            }
        }
        catch (Exception e) {
            System.err.println("Error -> "+e.getMessage());
            throw new jZipException("[compressFile]" + e.getMessage());
        }
        finally {
            try {
                if (fis != null)

```

```

        fis.close();
        if (zos != null ) {
            zos.closeEntry();
            zos.close();
        }
    } catch (Exception e) {
        throw new jZipException("[compressFile]" + e.getMessage());
    }
}
return true;
}
static public boolean compressFiles(String file, String args[])
throws jZipException {

    Vector filesSrc= new Vector();
    Vector fileName = new Vector();
    Vector v = null;
    int start= 1;
    for (int n = start; n<args.length;n++) {
        String name;
        String path;
        String ext;
        name= toolsZip.getName(args[n]);
        path= toolsZip.getPath(args[n]);
        ext = toolsZip.getExtension(args[n]);
        v = toolsZip.GetFilesList(path, name, ext);
        for (int m = 0;m<v.size();m++) {
            String aux;
            aux = ((File)v.elementAt(m)).getPath();
            filesSrc.addElement(((File)v.elementAt(m)).getPath() );
            fileName.addElement(((File)v.elementAt(m)).getPath().replace('\\','/')) ;
        }
    }
    return _compressFiles(file, filesSrc, fileName);
}
}

```

La Clase `jzip` es la que encargada de realizar la compresión con el método **compressFiles**. Este método tiene dos parámetros:

- String `fileName`: es el nombre del fichero ZIP a crear
- String `args[]`: es el array de strings con todos los ficheros a incluir. Acepta el comodín asterisco (*) y el path.

El método `compressFiles` llama a su vez a `_compressFiles` que tiene los siguientes parámetros:

- String `fileNameZip`: es el nombre del fichero ZIP a crear
- Vector `filesSrc`: es el vector que contiene el origen de los ficheros a comprimir
- Vector `fileName`: es el vector que contiene el nombre con el que se almacenan los ficheros a comprimir

Por cada fichero a comprimir se crea una entrada (instancia) de `ZipEntry` que se añade al stream de salida de tipo `ZipOutputStream` asociada al fichero Zip a crear y se copia los ficheros en el output stream proporcionado.

En el caso de que haya algún problema se lanzará una Excepción de tipo **jZipException**

```
import java.io.*;
```

```

import java.util.*;

public class zipFileFilter implements java.io FilenameFilter {
    private String name;
    private String extension;
    private boolean caseSensitive;
    public zipFileFilter (String namefile, String ext, boolean casesensitive) {
        caseSensitive = casesensitive;
        if (caseSensitive) { // for unix enviroments
            name = namefile;
            extension = ext;
        }
        else {
            name = namefile.toUpperCase();
            extension = ext.toUpperCase();
        }
    }

    public boolean accept(java.io.File dir, String fileName) {
        if (name.equals("") && extension.equals("")) //this ignore the case letter
            return true;
        if (name.equals("")) {
            if (caseSensitive) {
                if (! fileName.endsWith("." + extension)) return false;
            }
            else {
                if (! fileName.toUpperCase().endsWith("." + extension)) return false;
            }
            return true;
        }
        if (extension.equals("")) { //this ignore the case letter
            if (caseSensitive) {
                if (! fileName.startsWith(name)) return false;
            }
            else {
                if (! fileName.toUpperCase().startsWith(name)) return false;
            }
            return true;
        }
        if (caseSensitive) {
            if (! fileName.startsWith(name)) return false;
            if (! fileName.endsWith("." + extension)) return false;
        }
        else {
            if (! fileName.toUpperCase().startsWith(name)) return false;
            if (! fileName.toUpperCase().endsWith("." + extension)) return false;
        }
        return true;
    }
}

```

La Clase `zipFileFilter` implementa el interficie `FilenameFilter` para filtrado de nombres de ficheros, usado en la clase `toolsZip`.

```

import java.io.*;
import java.util.*;

public class toolsZip {
    static public String getPath(String name) {
        String tmp = new File(name).getName();
        String tmp0 = new File(name).getPath();
        int len = tmp0.length() > tmp.length() ? tmp0.length() - tmp.length():0;
        if (len > 0)
            return tmp0.substring(0, len);
        return "";
    }
    static public String getName(String name) {
        String tmp = new File(name).getName();
        String ret = "";
        StringTokenizer st = new StringTokenizer(tmp, ".");
        while(st.hasMoreTokens()) {
            ret = st.nextToken();
        }
        if (ret.length() != 0) {

```

```

        int len = tmp.length()-ret.length()-1;
        if (len >0)
            return tmp.substring(0, len);
        if (tmp.startsWith("."))
            return "";
    }
    return tmp;
}
static public String getExtension(String name) {
    String tmp = new File(name).getName();
    String ret = "";
    if (tmp.length()==0)
        return ret;
    StringTokenizer st = new StringTokenizer(tmp, ".");
    while(st.hasMoreTokens()) {
        ret = st.nextToken();
    }
    if (ret.length()!= tmp.length())
        return tmp.substring(tmp.length()-ret.length());
    return "";
}

static public Vector getFilesList(String sourcePath, String name, String ext){
    Vector v = new Vector();
    String fullPathFile = null;
    File filetmp = null;
    boolean existfile = false;
    String separador = File.separator;
    if (sourcePath == null)
        sourcePath = "";
    if (sourcePath.endsWith(File.separator)) {
        if (sourcePath.length()>1)
            sourcePath = sourcePath.substring(0, sourcePath.length()-1);
        else {
            separador = "";
        }
    }
    if (sourcePath.length()==0)
        sourcePath = System.getProperty("user.dir", File.separator);
    File dir = new File(sourcePath);
    if (name == null)
        name = "*";
    if (ext == null)
        ext = "*";
    try {
        if (dir.exists()) {
            String[] files = dir.list(new zipFileFilter(name, ext, false));
            if (files.length != 0) {
                for (int i = 0; i < files.length; i++) {
                    if (files[i].equals(".") || files[i].equals(".."))
                        continue;
                    fullPathFile = sourcePath + separador + files[i];
                    filetmp = new File(fullPathFile);
                    if (filetmp.isDirectory()) { // si es un directorio lo ignoramos
                        continue;
                    }
                    else {
                        v.addElement(filetmp);
                    }
                }
            }
        }
    } catch (SecurityException ee) {
        String str = "[getFilesList] " + ee.getMessage();
        System.err.println(str);
    };
    return v;
}

static public boolean deleteFile(String nameFile){
    File file = new File(nameFile);
    boolean ret = false;
    try {
        if (file.exists())
            if (file.isFile()) {
                ret = file.delete();
            }
        else {
            String str = "[deleteFile] " + nameFile + "no es un fichero.";

```

```

        System.err.println(str);
        return false;
    }
}
catch(SecurityException ee) {
    String str = "[deleteFile] " + ee.getMessage();
    System.err.println(str);
    return false;
};
return ret;
}
static public boolean existFileDir(String nameFile) {
    File file = new File(nameFile);
    boolean ret = false;
    try {
        if (file.exists())
            return true;
    }
    catch(SecurityException ee) {
        String str = ee.getMessage();
        System.err.println(str);
    };
    return ret;
}
}

```

La Clase `toolsZip` contiene métodos para la manipulación de ficheros y directorios:

- **getPath** obtiene el path de fichero de una cadena.
- **getName** obtiene el nombre de fichero de una cadena.
- **getExtension** obtiene la extensión de fichero de una cadena.
- **getFilesList** obtiene todos los archivos con el patrón marcado por los parámetros **sourcePath**, **name** y **ext**.
- **deleteFile** borra el fichero especificado por el parámetro **nameFile**.

```

import java.io.*;
import java.util.*;

public class javaZip {
    public static void main(String args[]) {
        String file = "";
        if (args.length < 2 ){
            System.out.println("sintaxis: " +
                "javaZip <nombre fichero ZIP> " +
                "<nombre fichero 1>[<nombre fichero 2>...<nombre fichero n>]");
            return;
        }
        file = args[0];
        try {
            jZip.compressFile(file, args);
        }
        catch(Exception e ) {
            System.out.println("ERROR:" + e.getMessage());
        }
    }
}

```

`javaZip` contiene el código que obtiene los parámetros y llama a **jZip.compressFile**

3. BIBLIOGRAFIA Y REFERENCIAS

- [1] Nelson, Mark, "The Data Compression Book", M&T Books, 1991
- [2] Zlib page <http://www.info-zip.org/pub/infozip/zlib/>
- [3] PkWare <http://www.pkware.com/>
- [4] Winzip <http://www.winzip.com/>
- [5] RFC 1951 <http://www.w3.org/Graphics/PNG/RFC-1951>