

MANUAL DE CONFIGURACIÓN DEL GENERADOR DE CATÁLOGOS OFF-LINE SOBRE EL ESTÁNDAR XML DEL PROYECTO BUSCATEX: CÁTALEX



CÁTALEX

CATÁLOGO OFF-LINE DE PRODUCTOS PARA XML COMPATIBLE CON EL ESTÁNDAR DE BUSCATEX

Introducción

El Catálogo offline de Catalex es una aplicación para Windows que carga un catálogo basado en la tecnología XML, que integra el estándar del proyecto BUSCATEX dirigido por AITEX - Instituto Tecnológico del Textil, y es interpretado mediante una plantilla WEB del tipo html. El resultado final es una aplicación local que contiene un catálogo organizado de productos, pertenecientes a una determinada empresa, sin necesidad de estar conectado a Internet (off-line). Por tanto, este catálogo es fácilmente transportable en CD-Card, CD-Rom o bajo cualquier soporte, incluso vía e-mail ó descarga web.

CATALEX es práctico porque la empresa cliente puede tener ya su propio catálogo de productos generado. Aplicar este catálogo XML el diseñador web sólo tiene que incluir una serie de variables y procedimientos en una web HTML que unirán el contenido del XML al catálogo off-line.

Ventajas

- Catalex es un catálogo de productos ***dinámico***. Actualizar productos es tan fácil como editar su código XML, de fácil interpretación.
- Catalex es ***versátil***. El aspecto y las funcionalidades pueden variar, adaptándose a las necesidades de cada cliente.
- Catalex es ***compatible***. Compatible con los formatos estándar: *Buscatex, XML, HTML, Java...* Con lo que se pueden usar las herramientas de uso común y extendido para desarrollar el catálogo. El diseñador no necesita cambiar su entorno de trabajo, utilizando los programas que estime adecuados para su edición.
- Catalex es ***Potente y Visual***. Tiene toda la potencia de ser compilado en C++, además de tener todas las posibilidades que ofrece un navegador web para hacer sus diseños. Catalex está preparado para interpretar otros XML de apoyo que posibilitan un diseño más dinámico todavía, como puedan serlo las variaciones visuales debidas al

multilinguaje, evitando así tener que hacer varios catálogos, uno por idioma, y facilitar cualquier cambio a posteriori sea lo más sencillo posible.

- Catalex es un producto de **Futuro**. Catalex no se detiene aquí y sigue mejorando día tras día, compatible con las versiones anteriores.

Diseño del catálogo.

Realizar el diseño es tan sumamente sencillo como hacer una página web, pues Catalex es al mismo tiempo, un explorador WEB. Por tanto, se puede usar HTML y java estándar. Soporta elementos flash y similares, aunque de ello dependerá que el ordenador donde se ejecute disponga de los plugins necesarios instalados.

Por tanto, el diseño:

- Puede realizarlo cualquier usuario con conocimientos de HTML o de cualquier programa de diseño web.
- Puede reutilizarse un diseño web existente del cliente.
- Puede darse en forma de plantillas tipo, de tal modo que el cliente, con sólo cambiar imágenes, logotipos y texto pueda personalizar su catálogo.

Hacer del diseño web un catálogo de productos.

Para pasar de la plantilla puramente HTML a un catálogo off-line funcional deben introducirse etiquetas especiales dentro del HTML para gestionar la relación entre el HTML y el XML.

En este documento se representan con ejemplos las tareas más comunes.

En cualquier caso, dado que se aportarán plantillas tipo, con cambiar el aspecto HTML será suficiente en la mayoría de los casos para que el catálogo sea funcional y personalizado.

Requisitos.

- Windows 98 o superior.
- Cualquier versión del Internet Explorer

Descarga via web de la aplicación principal.

La aplicación principal consta únicamente de dos ficheros: "catalex.exe" y "configurar.exe", los cuales no requieren instalación ni configuración manual. Esto resulta práctico, pues sólo con copiar la carpeta contenedora del catálogo, Catalex está listo para usarse.

La aplicación "catalex.exe" iniciará inmediatamente el catálogo, mientras que "configurar.exe" nos permitirá cambiar el catálogo activo en caso de tener varios instalados.

Se recomienda distribuir la aplicación junto a un catálogo por defecto.

Además, para distribuir Catalex, recomendamos el uso del programa WinRAR (bajo licencia), con el que se pueden generar de forma muy sencilla archivos de instalación. También se pueden distribuir los catálogos por separado de la aplicación de la misma forma.

Archivos básicos de Catalex

Catalex\catalex.exe	1,1 Mb
Catalex\configurar.exe	37 kb
Catalex\catalex.ini	1kb (generado por configurar.exe)

Dentro del directorio Catalex se pueden crear tantos catálogos como se requieran. Cada catálogo puede estar en una carpeta distinta sin ninguna restricción de nombre. Simplemente, en el archivo **catalex.ini** debe reflejarse cuál será el catálogo a iniciar, esto es, el directorio y la página de inicio, tal como muestra el ejemplo.

catalex.ini

```
[Form]
Top=75
Left=424
Width=830
Height=760
[Catalogo]
TPL_Home=catalogo1\index.htm
XML_Catalog= catalogo1\catalogo1.xml
Splash=catalogo1\splash.jpg
[SESSION]
Idioma=ES
```

MANUAL DE PROGRAMACIÓN PARA CATALEX

Variables. Estructura de Árbol. Rutas y Resolución.

Variables y su estructura de árbol.

Catalex procesa internamente el archivo XML y lo coloca en memoria para poder acceder a los datos con mayor eficacia. Así pues, Catalex almacena los datos en estructura de árbol, tal y como viene reflejado en el XML.

Por tanto, una etiqueta XML

```
<Catalogo>  
  <Productos>  
    <Producto>  
      <Codigo>Juego Sábana Bordada Esencia</Codigo>
```

Se correspondería a la siguiente variable (ruta) del sistema:

root.#document.Catalogo.Productos.Producto.Codigo

Nótese que cada punto determina la separación entre niveles.

root.#document es el raíz del sistema, y será común a todas las variables. Sin embargo el sistema permite acceder a la misma variable sólo poniendo un último tramo significativo, como por ejemplo:

Producto.Codigo

Esto sería correcto siempre y cuando no existiese otra variable cuya terminación fuese idéntica.

Nota: Se entiende como tramo mínimo una sucesión de, al menos 2 niveles. Por lo que colocar "***Codigo***" como variable, aunque fuese única, no sería bien resuelta por el sistema.

Dado que el sistema necesita diferenciar casos como el siguiente:

```

<Productos>
  <Producto>
    <Codigo>Juego Sábana Bordada Esencia</Codigo>
    ...
  </Producto>
  <Producto>
    <Codigo>Mantel Cuadros Escocia</Codigo>
    ...
  </Producto>
  ...
</Productos>

```

Aparentemente, ambos productos tendrían la misma ruta/variable, esta es:

root.#document.Catalogo.Productos.Producto.Codigo

Por ello, el sistema asignará al primer elemento, esta variable:

root.#document.Catalogo.Productos.Producto.Codigo

Al segundo valor encontrado:

root.#document.Catalogo.Productos.Producto[1].Codigo

Y así tendríamos el siguiente array de valores:

```

root.#document.Catalogo.Productos.Producto.Codigo
root.#document.Catalogo.Productos.Producto[1].Codigo
root.#document.Catalogo.Productos.Producto[2].Codigo
root.#document.Catalogo.Productos.Producto[3].Codigo
...
root.#document.Catalogo.Productos.Producto[n].Codigo

```

Rutas y variables del sistema.

Hasta ahora hemos estado hablando de rutas y variables del sistema como un mismo elemento. Esto es cierto, puesto que las variables están estructuradas como un camino, una estructura de árbol que ya hemos visto.

Sin embargo, es importante distinguir entre la variable y el valor de la variable, o, lo que es lo mismo, entre la ruta y lo que contiene esa ruta.

No podemos asignar el valor de ninguna variable del sistema directamente, sino que generalmente nos valdremos de ellas para leer su contenido, no para escribirlo. En realidad el valor de las variables del catálogo XML se modifican desde el propio XML. De cara al programador web, estas variables pueden considerarse CONSTANTES.

Podemos colocar directamente una ruta en nuestro HTML, tal que así:

`root.#document.Catalogo.Productos.Producto[7].Codigo`

pero el sistema no devolvería el valor, no la RESOLVERÍA, apareciendo este mismo texto tal cual en nuestro catálogo HTML. Si lo que queremos es que el sistema nos devuelva el contenido de la variable (nos lo resuelva), tendremos que poner la ruta (variable) entre claves { y } así:

`{root.#document.Catalogo.Productos.Producto[7].Codigo}`

O usando el método abreviado:

`{Producto[7].Codigo}`

Esto nos devolvería el Código del producto insertado en octavo lugar (7+1) en el XML y aparecería en nuestro catálogo web.

Por ejemplo:

```
<HTML>
  <center>
    <!-- BEGIN bloqueprocesador -->
      {Producto[7].Codigo}
    <!-- END bloqueprocesador -->
  </center>
</HTML>
```

Tendría el siguiente resultado:

Tapicería Pantera Negra

NOTA: en realidad, para que el se procesen las variables, éstas deben de estar dentro de un bloque. En este ejemplo hemos puesto un bloque llamado "bloqueprocesador". Los bloques los explicaremos más adelante, pero adelantar que comprenden el tramo de código que va desde BEGIN hasta END.

Variables especiales

Existen unas variables del sistema especiales.

- Las que se crean a partir de otras del XML. Esto es, la información del XML se desglosa a nivel interno para agilizar las búsquedas.
- Las variables de sesión.
 - Almacenan claves de búsqueda (*SearchKey*)
`root.#document.Catalogo.Colecciones.Coleccion[1].
 .SearchKey.Talla.Valor276 = 'Ancho 90 cm'`
 - TPL_SELF

`{TPL_SELF}` Representa el nombre de la propia plantilla HTML (por ejemplo: *micatalogo.htm*). Es útil para llamarse a sí mismo mediante `<a href>` y pasar una variable por el método GET como se haría en PHP, de la siguiente manera:

```
<a href="{TPL_SELF}?NombreVariable=valorvariable">
</a>
```

- valores que varían a cada clic (*variables tipo GET*)

Las variables pasadas por el método anterior, serían leídas en el próximo clic de la siguiente manera:

```
{_GET.Nombrevariable}
```

- Variables de nombres de bloque. Son las que le dan nombre a un bloque de código. Les podemos poner el nombre que queramos siempre y cuando no se repita en otro lugar del código. Las veremos más adelante.
- Variables resultantes del bucle FOREACH. Las veremos más adelante.

En cualquier caso, todas las variables deben tener un valor único e irrepetible a lo largo del código.

Procedimientos y bloques: Bucles, condiciones y demás.

BLOQUES de código

Ya adelantamos anteriormente que el procesado de variables requiere que éstas se encuentren contenidas dentro de un BLOQUE. Un bloque lo compone una porción de código comprendida entre claves BEGIN y END. El nombre del bloque es indistinto (variable) y sólo sirve para identificarlo como único.

```
<HTML>
  <!-- BEGIN bloque_1 -->
    <center>
      Bloque número 1 <br>
      Nombre del fichero activo en negrita:
      <b>{TPL_SELF}</b> <br> <br>
    </center>
  <!-- END bloque_1 -->

  <!-- BEGIN bloque_2 -->
    <center>
      Bloque número 2 <br>
      Nombre del fichero activo en cursiva:
      <i>{TPL_SELF}</i> <br> <br>
    </center>
  <!-- END bloque_2 -->
</HTML>
```

BUCLE FOREACH

Ya hemos visto como podemos acceder a una variable individual, a un producto determinado como podrían ser cualquiera de los siguientes:

```
root.#document.Catalogo.Productos.Producto.Codigo
root.#document.Catalogo.Productos.Producto[1].Codigo
root.#document.Catalogo.Productos.Producto[2].Codigo
root.#document.Catalogo.Productos.Producto[3].Codigo
...
root.#document.Catalogo.Productos.Producto[n].Codigo
```

No obstante, si lo que queremos directamente es mostrar todos los productos sin la necesidad de saber priori cuántos de ellos existen introducidos en el XML, si lo que queremos es que nuestro catálogo sea dinámico y fácilmente ampliable, la solución es usar unas etiquetas especiales en nuestro catálogo que permitan recorrer y duplicar el código automáticamente según lo que dicte

el XML. Es lo que llamamos BUCLE. En nuestro caso, las etiquetas de bucle las denominamos FOREACH.

Este código de ejemplo, mostraría todos los productos de cualquier catálogo XML compatible con Buscatex centrado en pantalla:

```
<HTML>
  <!-- BEGIN_FOREACH Bucle_ejemplo Catalogo.Productos $P -->
    <center>
      {$P.Codigo} <br>
    </center>
  <!-- END Bucle_ejemplo -->
</HTML>
```

Todo tramo de código comprendido entre etiquetas BEGIN y END se le llama BLOQUE. Por tanto, el tramo anterior, un BUCLE, se trata de un BLOQUE FOREACH.

- **Bucle_ejemplo** es el nombre del bloque (cualquier nombre)
- **Catalogo.Productos** es la raíz (puede ser cualquier raíz existente)
- **\$P** es la variable que almacena todos los subnodos que cuelgan de la raíz especificada. (puede ser cualquier nombre)

El bloque FOREACH permite filtrar al mismo tiempo mediante la cláusula WHERE. Así por ejemplo, podríamos filtrar todos los productos cuya clasificación contenga la palabra "Sábanas", tal que así:

```
<!-- BEGIN_FOREACH Bucle_ejemplo Catalogo.Productos $P WHERE
Clasificacion" LIKE "Sábanas" -->
```

Esto filtraría los productos que tuvieran una estructura y unos valores como el siguiente caso:

```
<Producto>
  ...
  <ClasificacionProducto>
    <ClasificacionValor>
      <Clasificacion>textil hogar</Clasificacion>
      <Valor>Sábanas Bordadas</Valor>
    </ClasificacionValor>
  </ClasificacionProducto>
  ...
</Producto>
```

La sintaxis completa de FOREACH sería:

```
<!-- BEGIN_FOREACH nombrebloque rutaorigenbusqueda $variable
WHERE "clavedebúsqueda" [comparador] "valor" ) -->
```

COMPARACIONES

Se ha usado en el ejemplo anterior la cláusula WHERE para filtrar y el comparador LIKE para determinar que Sábanas debe de estar incluida en el valor de la clave de búsqueda.

Los comparadores permitidos son:
 (Para **A** comparado con **B**)

"A" LIKE "B"	A incluido en B
"A" !LIKE "B"	A no incluido en B
"A" == "B"	A idéntico a B
"A" != "B"	A distinto de B
"A" > "B"	A mayor que B
"A" < "B"	A menor que B

BLOQUE SWITCH

Se usa este bloque para decidir si se ejecuta o no una parte de código. Si tenemos una portada donde hay dos links, uno para "**mostrar catálogo**" y otra para ver un "**quiénes somos**", una manera podría ser la siguiente:

```
<a href="{TPL_SELF}?link=catalogo">Mostrar Catálogo</a><br>
<a href="{TPL_SELF}?link=quienessomos">Quiénes somos</a><br>
```

En el siguiente click se podría procesar lo siguiente.

```
<!-- BEGIN_SWITCH vercatalogo IF "_GET.link"=="catalogo" -->
<center><b>Nuestro Catálogo de Productos</b></center>
<!-- BEGIN_FOREACH for_prods Catalogo.Productos $P -->
<center> { $P.Codigo } <br> </center>
<!-- END for_prods -->
<!-- END vercatalogo -->

<!-- BEGIN_SWITCH verquienessomos IF "_GET.link"=="quienessomos" -->
<center><b>Nuestro Catálogo de Productos</b></center>
<!-- BEGIN_FOREACH for_prods Catalogo.Productos $P -->
<center> { $P.Codigo } <br> </center>
<!-- END for_prods -->
<!-- END verquienessomos -->
```

En la misma parte de código se puede contemplar el tramo anterior y añadir que si no se pasó ningún link por GET, muestre la portada:

```
<!-- BEGIN_SWITCH verportada IF "_GET.link"==" " -->
  <a href="{TPL_SELF}?link=catalogo">Mostrar Catálogo</a><br>
  <a href="{TPL_SELF}?link=quienessomos">Quiénes somos</a><br>
<!-- END verportada -->
```

La sintaxis completa del bloque SWITCH sería:

```
<!-- BEGIN_SWITCH nombrebloque IF "variable" [Comparador] "valor" -->
```

OTRAS ETIQUETAS DE CÓDIGO VÁLIDAS. SINTAXIS

CELLS_BY_ROW:

Asigna la variable `cells_by_row` de un bloque `FOREACH` al valor deseado.

Uso: `<!-- CELLS_BY_ROW NOMBRE_BLOQUE_FOREACH VALOR -->`

Sirve para configurar la subdivisión de las filas en los listados `foreach`.

LOAD_XML:

Sirve para cargar un XML adicional. Normalmente es para cargar un XML que contenga los textos fijos del catálogo en distintos idiomas.

USO: `<!-- LOAD_XML nombre_fichero.xml -->`

NOTA: Los ficheros son cargados en `root.#variables._SESSION.XML`, más el nombre del fichero. `idiomas.xml` se ubicaría en: `root.#variables._SESSION.XML.idiomas`

Eventos: (Son los que empiezan con `ON_`)

`ON_BEGIN_ROW:`

`ON_END_ROW:`

Sirven para el cambio de línea en un `foreach`.

NOTA: Los eventos también deben terminar con su respectivo `END`.