

## DynaBase y xml: edición/gestión web de información corporativa

Por Antonio de la Rosa

EL MODELO de hojas de estilo de *DynaBase* es un medio para gestionar y editar en el web de forma dinámica grandes colecciones de información. En estas páginas describiremos someramente su entorno y componentes: *DynaBase server 3.12*, *DynaBase web developer*, *DynaBase web manager* y *DynaBase scripting language*.

Sobre todo se intenta destacar una opción de implementación práctica de una serie de buenas ideas como la generación dinámica de páginas web, la separación de contenido —identificado, cada vez más a menudo, con una colección de documentos xml y su/s correspondiente/s *DTD/s*—, presentación y comportamiento, la explotación de los navegadores para descargar al servidor de procesos, el uso de distintos lenguajes de *scripting* y la decidida orientación a objetos. Lo más interesante es observar cómo las soluciones avanzan hacia la aplicación de nuevas tecnologías multiplataforma, escalables y no propietarias —la arquitectura de las hojas de estilo para

presentación se basa en el estándar xml del *W3C*—.

*Inso* es una de las compañías punteras a la cabeza de la I+D comercial en aplicaciones xml y sgml. Sólo es necesario mencionar, por ejemplo, *DynaText* o *DynaTag*. La novedad es, tal vez, que facilita a sus clientes el desarrollo de soluciones, sobre todo para intranets, basadas en tecnologías abiertas. Esta apuesta puede parecer algo arriesgada —realmente no lo es tanto si se analiza la forma en que comercializan sus productos o sus cursos— y es sintomático de algo que ya es un hecho: xml es útil y su rentabilidad no deja de crecer.

El entorno reseñado en este texto es *DynaBase dynamic web publishing system*, un sistema integrado de publicación y gestión web basado en la orientación a objetos, la arquitectura cliente servidor, el protocolo http, la utilización de xml, dhtml, html y diversos lenguajes de *scripting* para organizar

directamente la información. En primer lugar, quizá sería útil presentar sus principales componentes de forma gráfica, tal y como pueden verse en la figura 1.

El servidor *DynaBase* contiene una base de datos orientada a objetos (servidor de

datos) y un *plugin* que interactúa con el servidor web que lo esté usando (*Netscape enterprise server 3.5.1* o *Microsoft internet information server 4.0*). Una de sus funciones es controlar los cambios en los objetos que mantiene. Como se analizará más

adelante, *DynaBase* mantiene múltiples versiones de, por ejemplo, los documentos que gestiona haciendo uso de memorias temporales y de la caché del navegador, lo que presenta muchas ventajas y algunos inconvenientes.

**«DynaBase mantiene múltiples versiones de los documentos que gestiona haciendo uso de diversas memorias temporales y de la caché del navegador»**

En el servidor también hay un intérprete para el lenguaje de *scripting DynaBase*, mediante el cual se puede controlar el comportamiento de los objetos (a nivel de documento, objeto *DynaBase* e, incluso, etiqueta). Su utilización no significa que se renuncie a las posibilidades de otros medios como *Java*, *JavaScript*, *Perl* o *Visual basic*. Simplemente unos scripts son interpretados y ejecutados en el servidor y otros en los clientes. Esta modularización de tareas resulta realmente efectiva.



Antonio de la Rosa

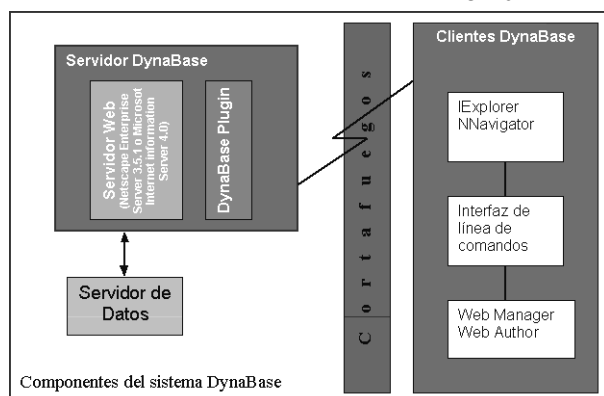


Figura 1

**Tabla 1. Inso general tag pack:** algunas etiquetas con funciones especiales en el sistema *DynaBase*.

El comportamiento de muchas de estas etiquetas está asociado a librerías de ficheros *.wbs* específicos desarrollados por *DynaBase*, como por ejemplo: *Inso general script library (general\_lib.wbs)*, *Inso search tag pack (search\_tag\_pack.wbs)*, *Inso Xpointer script library (Xpointers\_lib.wbs)*, *Inso CLI script library (cli\_lib.wbs)*, *Inso CLI command script library (cli\_command\_lib.wbs)*.

<Xpointer>

Se usa para recoger el contenido de un fichero y servirlo dentro de la hoja de estilo. La parte que debe recuperarse se define mediante los atributos de <Xpointer>. Por defecto, el documento sobre el que actúa es el especificado por la URL incluida en el mensaje de petición del navegador (se puede especificar otro mediante el atributo href de <Xpointer>, lo que lleva a la aplicación de conceptos como el de la transclusión).

<!--Escribe el primer P encontrado en el primer body del documento /content/antonio.html . -->

<Xpointer href="/content/antonio.html" Xpointer="child(1,body).child(1,P)" serve="False">

<list>  
<query />  
</list>

Crea una lista de artículos DynaBase definidos mediante etiquetas de consulta <query> y las muestradentro de la hoja de estilo.

<!--Escribe una lista en orden ascendente con los títulos de los documentos obtenidos. La búsqueda recupera los documentos cuya "docClass" sea fútbol, baloncesto, taeKwonDo o deportes y contenga la cadena de texto incluida en el parámetro search de la URL requerida -->

```
<list sortorder="ascending" sortproperty="sortKey" sortkey="child(1,title)" formatID="list1">
  <query type="property" query="fútbol" op="6" property="docClass">
  <query type="property" query="baloncesto" op="6" property="docClass" merge="0" >
  <query type="property" query="taeKwonDo" op="6" property="docClass" merge="0" >
  <query type="content" query="deportes" merge="0">
  <query type="content" query="search" dynamic="True" merge="1">
</list>
```

<include />

Se utiliza para servir un objeto (un documento, por ejemplo) *inline* con la hoja de

estilo. No se halla en el interior de ésta pero se procesa cuando el parser llega a la referencia localizada en la etiqueta <include>. El artículo se referencia mediante el atributo SRC.

```
<include src="/content/antonio.html" />
<switch>
<case />
<else />
</switch>
```

La etiqueta <switch> sirve un documento que responde de forma diferente a distintos valores de una propiedad especificada.

<!--Dependiendo de la propiedad "docClass" del documento requerido, esta etiqueta escribe dentro de la hoja de estilo una de las versiones de copyright.html (sports\_copyright o news\_copyright) si el valor de la propiedad no es news ni sports, entonces escribe el documento por defecto, copyright.html -->

```
<switch prop="docClass">
  <case href="/content /sports_copyright.html" value="sports" />
  <case href="/content /news_copyright.html" value="news" />
  <else href="/content/default_copyright.html" />
</switch>
```

<prop />

Recupera el valor de una propiedad de un objeto y lo sirve dentro de la hoja de estilo. Esta propiedad se define mediante el atributo prop. La fuente por defecto donde se busca es la URL incluida en el mensaje de petición.

```
<prop prop="docClass" />
<prop prop="whenVersionCreated" />
<prop src="/content/123098.html" prop="estimatedExportSize" />
```

<date />

Recupera del servidor el valor de la fecha y lo presenta dentro de la hoja de estilo. El formato por defecto es 'mmmm, dd, aaaa'.

```
<date />
<date format="dd/mm/yy" />
```

<frame>

Esta etiqueta es equivalente a <frame> en html. Se han hecho modificaciones para implementar frames en las hojas de estilo. Por ejemplo, el valor del atributo SRC puede

cambiarse a: CurrentSRC. De este modo dentro del frame aparecerá el contenido de la dirección URL incluida en el mensaje de petición.

```
<frameset border="false"
rows="100,* ,100" frameborder="0">
  <frame name="top"
src="/content/top_frame.html" noresize="noresize" scrolling="no" marginheight="0" marginwidth="0">
  <frame name="body" src="currentsrc"
noresize="noresize" marginheight="0" marginwidth="0" template="/demo_folder/sample_template.html">
  <frame name="bottom"
src="/content/bottom_frame.html" noresize="noresize" scrolling="no" marginheight="0" marginwidth="0">
</frameset>
<body><p>this browser does not support frames.</p></body>
</frameset>
</frameset>
```

```
<search>
<query />
</search>
```

Crea una lista de resultados de búsqueda correspondientes a las consultas <query> y las incluye dentro de la hoja de estilo.

```
<form action="/search.htm" method="get">
  <input name="search"
size="10">&nbsp;  <input type="submit" value="Search"><br>
</form><br><br>
```

<!-- Escribe una lista en orden ascendente con los títulos de los documentos recuperados. Con la búsqueda se obtienen aquellos cuya clase sea fútbol, baloncesto, taeKwonDo o deportes y que contengan la cadena de texto incluida en el parámetro search de la URL requerida -->

```
<search sortorder="ascending" sortproperty="SortKey" sortkey="hits" formatID="SearchResults" maxcount="10">
  <query type="property" query="fútbol" op="6" property="docClass">
  <query type="property" query="baloncesto" op="6" property="docClass" merge="0" >
  <query type="property" query="taeKwonDo" op="6" property="docClass" merge="0" >
  <query type="content" query="deportes" merge="0">
  <query type="content" query="search" dynamic="True" merge="1">
</ search >
```

El servidor puede funcionar sobre los sistemas operativos *Windows NT* o *Unix solaris* y su instalación es bastante difícil. Los requisitos para una instalación *NT* son: procesador mínimo *Pentium pro 200 Mhz.*, sistema operativo *NT 4.0* y *Service pack 3*, 128 Mb de ram, unos 60 Mb de espacio en disco y cualquiera de los servidores web anteriormente mencionados.

**«El programa facilita el mantenimiento del contenido, el diseño o rediseño de las sedes web»**

Los clientes permiten inspeccionar, editar, enlazar y gestionar el contenido de la/s sede/s web mantenida/s en el servidor. Además de los navegadores y cualquier herramienta que los editores utilicen normalmente —html o xml (el muchas veces menospreciado *Notepad* sobre todo), editores gráficos, etc.— *DynaBase* presenta, por orden de importancia, los siguientes programas:

1. *Web manager*: para localizar, importar/exportar, editar o mantener los artículos incluidos en el servidor.

2. *Web developer*: edita y controla la sintaxis de los scripts creados con el lenguaje de *scripting DynaBase*. Sirven para añadir cierto comportamiento a los documentos mediante el uso de los objetos definidos en el entorno.

3. *Command line interface (CLI)*: proporciona comandos que,

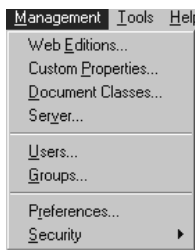


Figura 2

al ejecutarse sobre el servidor, permiten automatizar diferentes tareas: gestión de enlaces, asignación de clases, programación de algunas rutinas, etc.

4. *Web author*: edición de contenido.

La seguridad del sistema merecería un epígrafe aparte, pero por razones de espacio y enfoque sólo se señalan sus tres componentes principales:

— Autenticación con un servidor web. La identidad y el estatus de los usuarios se verifica mediante la base de datos del servidor.

— Autenticación gracias a un servidor proxy si es que se desea implementar un cortafuegos, reglas *DynaBase* de seguridad, etc.

— La gestión de contenidos y el control de acceso de flujos se llevan a cabo mediante reglas que se aplican globalmente sobre la sede e individualmente a los artículos que contiene.

**Separación de contenido, comportamiento y estilo**

Cuando el servidor *DynaBase* responde a una petición devolviendo una página web, lo puede hacer distinguiendo su contenido de la presentación y del comportamiento añadido a esa página mediante scripts. Para su separación se usan hojas de estilo: ficheros que contienen cierta información que es posible utilizar para formatear más de una página. Esto resulta enormemente útil a la hora de publicar grandes cantidades de datos puesto que se simplifican en gran medida todos los problemas tradicionales de la edición.

Las ventajas son las que se han citado ya tantas veces: se facilita el mantenimiento del contenido, el diseño o rediseño de las sedes web, que se

**Tabla 2. Etiqueta Xpointer.**

Formato:	<Xpointer />
No se requiere final de etiqueta.	
Atributos :	
Cache	Define el comportamiento de la memoria caché para la presentación de la página. Se le puede dar el valor <i>Disable</i> (con lo que la caché no sirve para nada) o cualquier otro permitido de fecha que funciona como indicador para el sistema en el momento de sustituir un documento en memoria por uno nuevo del servidor.
Href	Define el documento fuente del cual se va a recuperar parte del contenido, que debe ser <i>html</i> o <i>xml</i> . Si no se proporciona un valor se entiende que el documento fuente es el referido por la URL actual.
Serve	Es un atributo de dos valores <i>True/False</i> que indica si la parte de contenido a recuperar debe servirse permitiendo eventos que accionen procesos o no. El valor por defecto es <i>True</i> .
Xpointer	Define la parte del contenido que se debe recuperar dentro del documento referido por la URL. Usa el lenguaje <i>Xpointer</i> definido por el <i>W3C</i> .
Ejemplos:	<p>&lt;!—Sirve la primer parte del texto solicitado. —&gt;</p> <p>&lt;Xpointer Xpointer="child(1,body)"&gt;</p> <p>&lt;!—Escribe el primer título del documento solicitado. —&gt;</p> <p>&lt;Xpointer Xpointer="child(1,title)" serve="False"&gt;</p> <p>&lt;!—Escribe el primer párrafo encontrado en el primer cuerpo del documento solicitado. —&gt;</p> <p>&lt;Xpointer Xpointer="child(1,body).child(1,P)" serve="False"&gt;</p> <p>&lt;!—Sirve la primera parte del documento /content/copyright.html —&gt;</p> <p>&lt;Xpointer href="/content/copyright.html" Xpointer="child(1,body)"&gt;</p>

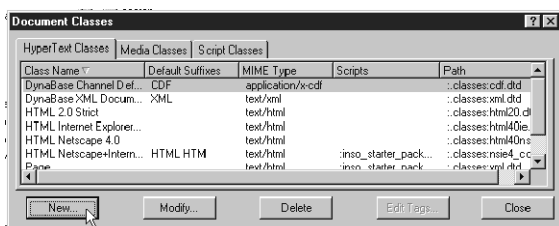


Figura 3

convierte en una tarea mucho más fácil, y se hace realidad el concepto de la reutilización. Al eliminarse la presentación de una página html estándar se reduce su tamaño en un 50 ó 80%, con lo que se minimiza parte de su complejidad y, en consecuencia, su coste de conservación.

Además, con ambos conceptos separados, las hojas de estilo controlan la apariencia del web. Por lo tanto sólo hay que modificarlas para rediseñar una parte o el total del mismo. La reutilización del contenido es cuestión solamente de cambiar la hoja de estilo a la que estaba asociado. Para ello, su distribución es tan simple como copiar ficheros, puesto que no hay necesidad de eliminarlos definitivamente, al igual que tampoco es obligatorio para las hojas de estilo.

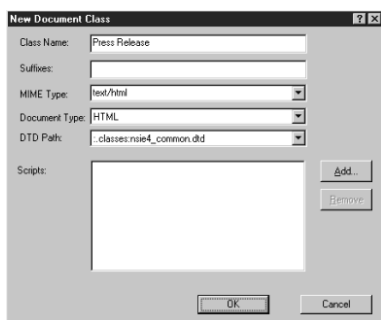


Figura 4

### Cómo implementa DynaBase las hojas de estilo

La manera mediante la cual da formato al contenido es a través de la creación de “métodos” html. Hay que recordar que se trabaja en un entorno orientado a objetos, y “método” se refiere a cierto comportamiento añadido a un objeto, en este caso a un documento html, y que se localiza en ficheros aparte: scripts DynaBase con la terminación .wbs.

El sistema maneja, al menos, tres documentos interrelacionados por cada página web servida al navegador: un fichero con el contenido (.html, .xml), una hoja de estilo (.html) y un script (.wbs). Todos ellos y sus componentes son obje-

tos y como tales deben ser establecidos en una clase para cumplir su función. DynaBase les asigna cada uno a una clase diferente, sin dejar por eso de mantener su interrelación.

Es interesante profundizar brevemente en por qué y cómo DynaBase obliga a que todos sus documentos/objeto pertenezcan a una clase. En relación a la primera cuestión, es obvio si se tiene en cuenta que se trata de un contexto orientado a objetos. El cómo es algo más difícil de explicar. En la imagen de la figura 2 puede verse el menú del cliente DynaBase web manager desde el cual se accede a las clases de los documentos que están actuando en el web. En la figura 3 se aprecia que existen clases para documentos (HyperText), media y script. Casi todas ellas están instaladas por defecto pero, en algunos casos, existe la posibilidad de añadir nuevas o modificar las existentes.

**«El sistema maneja tres documentos interrelacionados por cada página web servida al navegador: un fichero con el contenido, una hoja de estilo y un script»**

Se define una nueva clase de documentos cada vez que se desea que un grupo particular con contenido específico se comporte de una

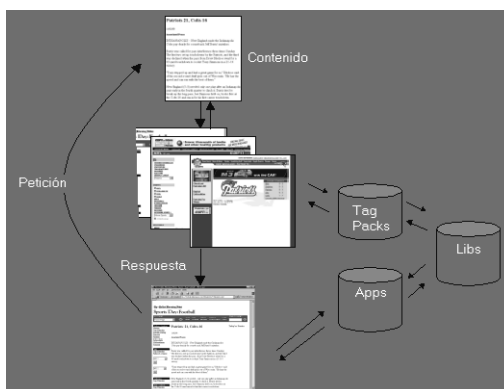


Figura 6

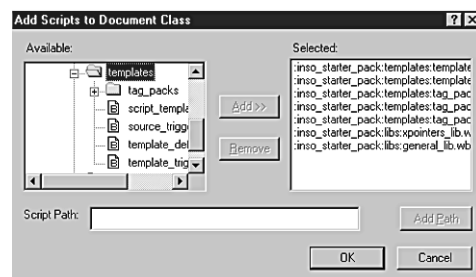


Figura 5

forma determinada. Dos tipos posibles podrían servir de ejemplo: “informes” para documentos de contenido o “templates” para hojas de estilo. Se puede definir una nueva clase de hipertexto siguiendo el siguiente proceso:

— Cuando se selecciona “New” en la imagen superior izquierda, aparece el menú de la figura 4:

— En el primer cuadro de diálogo se asigna el nombre de la nueva clase.

— Como tipo *Mime* (*multipurpose internet mail extensions*) se debe elegir el adecuado (tercer recuadro): text/html, text/xml, etc.

— El tipo de documento debe ser acorde a la elección anterior: html, xml.

— Si no existe una DTD específica (como en el caso de documentos xml) se puede seleccionar una por defecto, por ejemplo: :classes:nsie4\_common.dtd (la DTD que obedecen las versiones 4 y superiores de *Explorer* y *Netscape navigator*).

— Por último es necesario seleccionar en el cuadro de diálogo grande los scripts que van a interactuar con los documentos (objetos) de la clase que se está definiendo, que, a su vez, son objetos de su propia clase, generalmente de *DynaBase scripts*. La figura 5 muestra el menú desde el cual se localizan y eligen.

Una vez completados estos pasos se obtiene una nueva clase (informes o template), y ahora será necesario asignarle los documentos que le pertenecen. El

**Tabla 3.** Algunos de los objetos *DynaBase* más destacados.

Request:	Attrs	
Cookies	EventType	
Headers	NodeType	
http version	TagName	
IPAddress		
Item	Este objeto interacciona bidireccionalmente a través de la estructura arbórea de los documentos xml y html posibilitando así la localización del contenido de los elementos xml o html. Usando las propiedades y métodos de este objeto se puede navegar por el código de un documento de una etiqueta a otra, entre bloques de texto, etc. Así es posible especificar con total precisión aquello que se desea recuperar o generar.	
Method		
URL		
UserAgent		
UserCacheKey		
UserName		
Web		
Representa un mensaje http de petición. Se usa para acceder a la información contenida en este mensaje.		
Response:		Parameter:
Headers		Name
StatusCode	Value	
ReasonPhrase	Es un par nombre/valor. Un ejemplo claro de parámetro es uno de consulta y su valor basados en el encabezamiento de una petición http. Otro ejemplo es un nombre de atributo (en un elemento xml o html) y su valor.	
Representa un mensaje http de respuesta.		
	DocItem:	
URL:	PathName	
Host	Properties	
Params	Scanner	
Path	URL	
Port	URLs	
Query-Params	Version	
Scheme	VersionCount	
Server-URL		
Value	Representa un artículo (ítem) dentro de un web DynaBase. Es un fichero que puede contener: texto formateado en html o xml, un fichero script DynaBase (.wbs), imágenes en varios formatos (.gif, .jpg, .tiff), sonido (.wav) o vídeo (.avi, .mpeg, .mov). Un ítem puede ser también un directorio o subdirectorio del web.	
Recupera los parámetros (también los de consulta, si el que se ha recuperado se ha generado gracias a una) incluidos en la URL que contiene el mensaje http de petición.		
Scanner:		

sistema debe saber que los informes son de la clase “informes”, tipo *Mime* text/xml, tipo de documento xml, perteneciente a la *DTD :classes:informes.dtd* y con los scripts a, b y c asociados. De este modo el modelo consigue servir un documento cuya presentación depende de la actuación conjunta de varios componentes.

**«La recuperación, proceso y definición de objetos URL juega un papel muy importante a la hora de generar contenido dinámicamente»**

Centrándose en la presentación y en xml, cuando un navegador hace una petición al servidor *DynaBase*, éste localiza y sirve la hoja de estilo asociada con el documento referido en la URL en lugar de él mismo (contenido). Al procesarse ciertas etiquetas xml presentes en esta hoja —aquellas especiales incluidas en el *Inso tag pack* de *DynaBase* (ver tablas 1 y 2)—, recuperan el contenido apropiado y lo suman a la respuesta que se va a generar. En otras palabras, si desde el navegador se observa el código fuente de la página ofrecida, lo que se ve es una estructura (hoja de estilo), rellena dinámicamente (contenido), cuyo comportamiento depende de una serie de scripts *DynaBase* (script). En la figura 6 se esquematiza todo este proceso.

### Ejemplo práctico

El entorno *DynaBase* tiene muchos componentes e interacciones. En un artículo de estas dimensiones ya es complicado intentar dar una idea global precisa. La mejor forma de, al menos, destacar algunas de sus posibilidades y tal vez despertar el interés de los lectores (no por el producto, sino por la aplicación de la idea) puede ser desarrollando un ejemplo concreto.



sintaxis xml en general y por una *DTD* particular que determina su posición en la jerarquía, el número de elementos que pueden contener, el orden y, en general, la estructura arbórea del documento en cuestión, es decir, lo que aparece en la imagen. En resumen, información muy estructurada.

2. El segundo componente es la hoja de estilo (figura 9). Se trata de un fichero html que define la estructura de la tabla que aparece en el documento generado para el navegador. Lo que se debe destacar del código es la presencia de una serie de etiquetas xml especiales: fundamentalmente `<include/>` y `<Xpointer/>` (ver tablas 1 y 2). Las primeras sirven para procesar inline una serie de ficheros: la cabecera, la barra de navegación lateral y el formulario de búsqueda de la página:

```
<include
src="templates/header.txt"></include>

<include
src="templates/bovenkant.txt"></include>

<include
src="templates/linkerbalk.txt"></include>
```

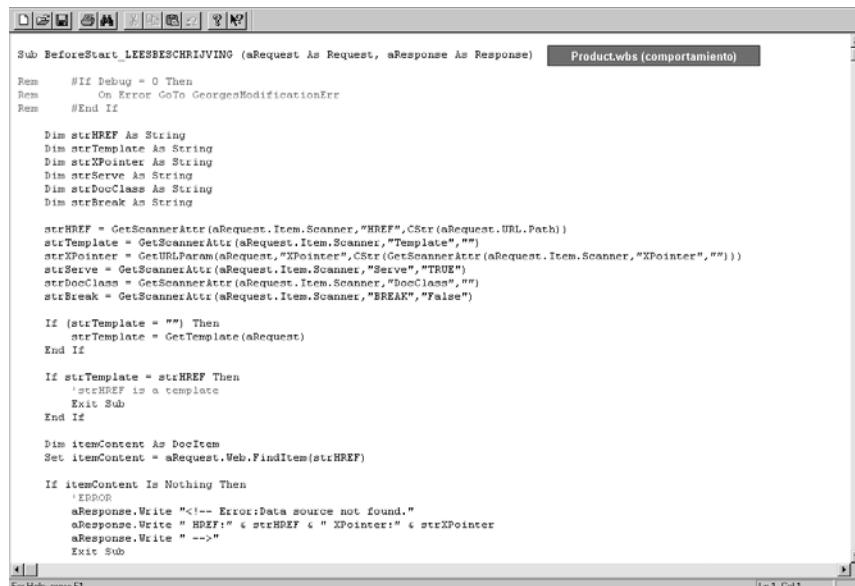


Figura 10

Las etiquetas *Xpointer* sirven para llamar ciertas partes del documento xml donde está el contenido e incluirlas en una determinada posición de la página:

```
<Xpointer
Xpointer="child(1,titel)"></Xpointer>

<Xpointer
Xpointer="child(1,definitie)"></Xpointer>

<Xpointer
Xpointer="child(1,dienst_van)"></Xpointer>
```

```
<Xpointer
Xpointer="child(1,dienst_voor)"></Xpointer>
```

El código anterior simplemente localiza el contenido de las etiquetas xml mencionadas (`<titel>` `<definitie>` `<dienst_van>` `<dienst_voor>`) y lo introduce en el lugar oportuno de la tabla. Por ejemplo, el de `<titel>`, *Global one*, es incluido dentro de la celda verde.

¿Y el resto? Cuando el navegador procese la etiqueta `<leesbeschrijving/>` recurrirá al tercer componente de la página vista al principio. Se trata del script *DynaBase* que opera sobre ella y, en este ejemplo, se llama *product.wbs*, que declara un conjunto de subrutinas que utilizan los objetos, colecciones de objetos, propiedades y métodos *DynaBase* mediante un lenguaje de *scripting* para implementar el comportamiento de la propia página. Por ejemplo: generar dinámicamente enlaces, organizar la estructura o añadir a la página una interfaz de búsqueda. El aspecto que presenta en *WebDeveloper* (el cliente *DynaBase* para editar scripts) es el que se puede apreciar en la Figura 10.

Excede los límites de este artículo comentar el código fuente de este fichero. Simplemente destacar

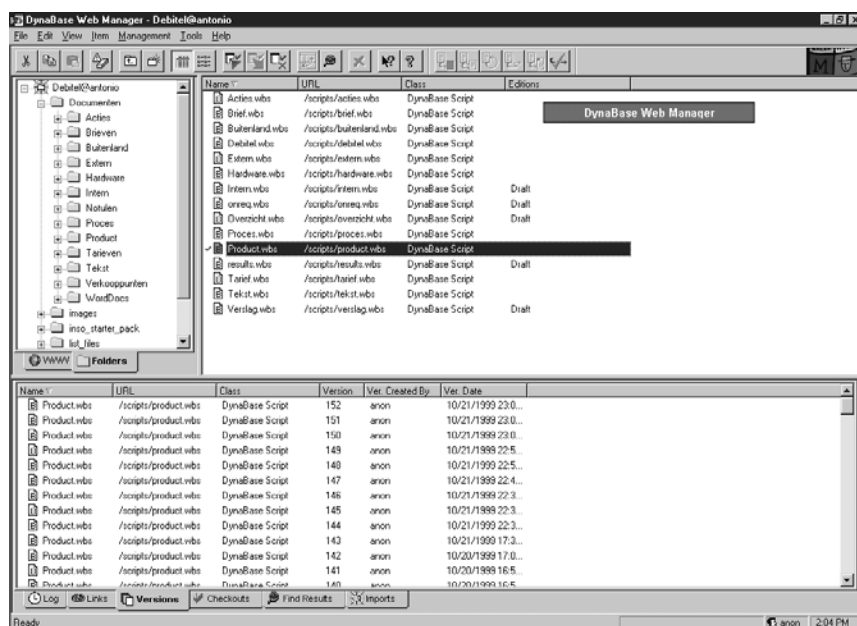


Figura 11

la llamada que se produce en la primera línea:

*Sub BeforeStart\_leesbeschrijving (aRequest As Request, aResponse As Response)*

Define una subrutina que se iniciará y cumplirá su función entre las etiquetas de inicio y final del elemento “*leesbeschrijving*”. De alguna forma sería como incluir todo lo que se encuentra entre *Sub* y *End Sub* (la subrutina) situadas dentro de las etiquetas de la hoja de estilo `<leesbeschrijving></leesbeschrijving>`. La ventaja es que comportamiento, contenido y presentación se implementan en tres ficheros diferentes que funcionan conjuntamente para generar una página. Esto significa ahorro en mantenimiento, mayor precisión en todos los niveles, posibilidad de reutilización y simplificación.

**«La ventaja es que comportamiento, contenido y presentación se implementan en tres ficheros diferentes que funcionan conjuntamente para generar una página»**

Para finalizar, se analizará brevemente cómo se gestiona todo este procedimiento desde el cliente *DynaBase web manager* tal y como se encuentra en la figura 11. En la parte superior izquierda de la imagen se muestra un árbol de directorios muy parecido a *Windows explorer* (solapa “Folders”), y que representa la sede web desde la que el servidor está suministrando páginas a los navegadores. Si se selecciona la solapa “www” aparecerá una lista con aquellas que mantenga en ese momento, *Debitel@antonio* entre ellas. Cada una debe estar asignada a un puerto distinto, y es recomendable que se usen carpetas distintas como memoria temporal de cada web. Es

necesario indicar que todas estas características deben especificarse durante la instalación del servidor *DynaBase*.

En la parte derecha de la imagen aparecen los artículos contenidos en la carpeta elegida a la izquierda, que en este caso es la de scripts, y los artículos son todos los ficheros *.wbs*. Es decir, aquellos que contienen las rutinas que controlan el comportamiento de los documentos del web y, entre éstos, se halla seleccionado *product.wbs*. La marca roja a la izquierda significa que ha sido seleccionado (checkout) del servidor para editarlo o revisarlo (esto produciría la imagen de *product.wbs* en *WebDeveloper*). Los ficheros que se extraen del servidor se almacenan en memoria temporal. Cada vez que se quiere editar un artículo del web es necesario hacer una comprobación, efectuar los cambios e integrarlo de nuevo mediante checkin.

En la parte inferior se ha seleccionado la solapa de versiones. En ella se pueden ver y activar o desactivar todas las establecidas (en este caso *product.wbs*) desde la primera vez que se creó el documento. El mantenimiento de esta estructura ofrece una idea del gran uso que hace *DynaBase* de la memoria. La parte positiva es que se puede seguir paso a paso el desarrollo de un documento y una sede web, con todas las implicaciones que esto tiene para la gestión — sobre todo a nivel documental y subyacente a esta práctica se halla el concepto del documento como entidad en continuo desarrollo—. La parte negativa es la enorme utilización de recursos y los problemas que a veces existen para recuperar en el navegador la última versión y no una anterior.

Además de estas solapas hay otras que se utilizan para: ver los resultados cuando se ha efectuado una búsqueda en la sede a través de

*Web manager* (se puede consultar por documento, texto, objeto, clase, etiqueta etc.), importación y exportación de ficheros, enlaces, checkouts de los documentos seleccionados y mensajes de gestión de *Web manager*.

## Conclusiones

Este trabajo no ha pretendido sólo señalar las virtudes de un producto como *DynaBase* sino sobre todo resaltar cómo es posible llevar a la práctica ciertas ideas: separación de contenido y presentación, documentos y etiquetas tratados como objetos, transclusión de contenidos, enlaces sofisticados, utilización de las posibilidades de los navegadores para descargar al servidor, gestión de versiones de un documento, etc. Estas nociones son las que continuamente se investigan en lugares como el W3C, las mismas que la mayoría suele tachar de poco realistas y nada rentables.

Pues bien, como se acaba de ver, existen compañías punteras que orientan sus productos hacia la aplicación práctica de esas ideas, lo que hace suponer que tienen grandes perspectivas de rentabilidad. En consecuencia se deberían tener más presentes las propuestas de *TI* que nos hacen reír. ¿Quién sabe? De aquí a unos años pueden dar dinero cosas como las interfaces *storytelling* o la investigación de la relevancia en recuperación de información.

## Notas

1. Algunos de esos ficheros son *javascript* que se ejecutan al cargarse la página.

## Bibliografía

Inso corporation. *DynaBase programming guide*. Revisión B. 1998, August.

<http://www.inso.com>

**Antonio de la Rosa**, Consultor/investigador, *Wisdom. Nieuwe Herengracht, 113. 1011 SB, Amsterdam. Holanda.*

[antonio@wisdom.nl](mailto:antonio@wisdom.nl)