Todo

Año 1 • Número 12 • **6,50** euros

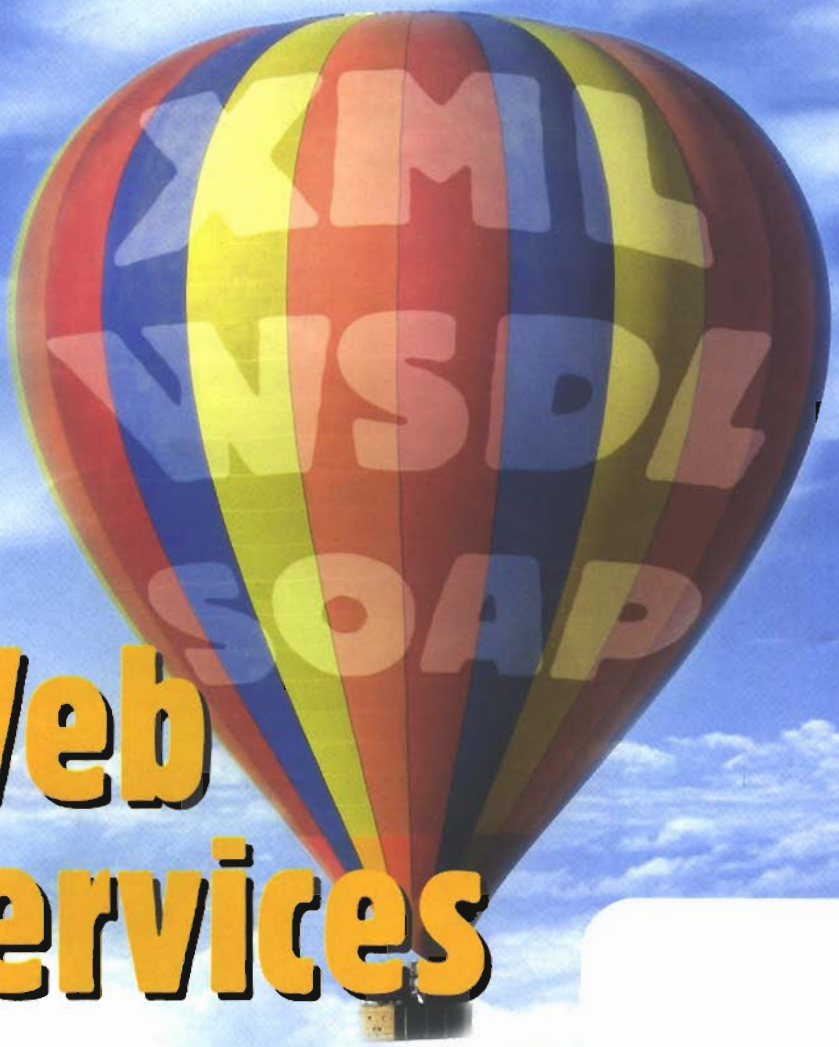
Studio Press

6 413042 751977 00012

Programación

La Revista bimestral para entusiastas de la programación

www.iberprensa.com



Web Services

Sesiones en PHP con PostNuke

Uso de Generics en el lenguaje Java

Diseño de interfaces gráficos con Glade

■ CD-ROM: ESPECIAL ENTORNOS DE PROGRAMACIÓN LINUX: ANJUTA, GLADE, KDEVELOP, ETC.

Zona Linux

- C#: Servicios Web, ejemplos con Google y Amazon
- Cómo crear repositorios con Subversion

Zona Windows

- F#: Programación funcional para .NET
- Programación basada en algoritmos genéticos

QUÉ ES LA PROGRAMACIÓN ROBUSTA Y CÓMO OBTENER CÓDIGO ROBUSTO EN JAVA

Todo
Programación
La Revista bimestral para entusiastas de la programación

DIRECTOR
Eduardo Toribio
etoribio@iberprensa.com

REDACCIÓN
Yenifer Trabadela
yenifer@iberprensa.com

COLABORADORES
Antonio M. Zugaldía
(azugaldia@iberprensa.com)
David Santo Orcero
(orcero@iberprensa.com)
Manuel Domínguez
(mdominguez@iberprensacom)
Fernando Escudero
(fescudero@iberprensa.com)
José Manuel Navarro
(jnavarro@iberprensa.com)
Marcos Prieto
(mprieto@iberprensa.com)
Guillermo "el Guille" Som
(elguille@iberprensa.com)
Santiago Márquez
(smarquez@iberprensa.com)
José Rivera
(jrivera@iberprensa.com)
Alejandro Serrano
(aserrano@iberprensa.com)
Jordi Massaguer
(jmassague@iberprensa.com)
Pedro Agulló
(pagullo@iberprensa.com)
Moisés Díaz
(mdiaz@iberprensa.com)

DISEÑO PORTADA
Antonio Gª Tomé

MAQUETACIÓN
Antonio Gª Tomé

DIRECTOR DE PRODUCCIÓN
Carlos Peropadre
cperopadre@iberprensa.com

ADMINISTRACIÓN
Marisa Cogorro

SUSCRIPCIONES
Tel: 91 628 02 03
suscripciones@iberprensa.com

FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: I. G. Printone
DUPLICACIÓN CD-ROM: M.P.O.

DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Studio Press
www.iberprensa.com

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, Nave 13.
Polígono "El Nogal"
28110 Algete (Madrid)
Tel.: 91 628 02 03*
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-13679-2004
Número 12 • Año 1

Eduardo Toribio

EDITORIAL

Web Services

Este mes dedicamos nuestro tema de portada a los Web services, un término que ya hace tiempo se emplea con frecuencia en distintos ámbitos. Estudiaremos desde una perspectiva teórica qué son los servicios web, para posteriormente pasar a ver algunos ejemplos de implementación práctica en la sección de Mono. Por lo demás seguimos incrementando nuestra oferta en el bloque Linux, ahora con programación para KDE con las librerías Qt, que también compilan para Windows, y las habituales secciones de Mono y GTK#, además del curso de Globus. En lo que a Windows se refiere, y concretamente a .NET, dedicamos un reportaje especial a la programación funcional con F#. Finalmente, y para que aparezcan todos los actores principales en este nuestro mundillo, incluimos dos tutoriales centrados en Java, el primero sobre el uso de Generics y el segundo sobre programación robusta.

SUSCRIPCIONES

Como oferta de lanzamiento existe la posibilidad de suscribirse durante 12 números a **Todo Programación** por solo 61 euros lo que significa un ahorro del 20% respecto el precio de portada. Además de regalo puedes elegir entre una de las dos guías que aparecen abajo.
Más información en: www.iberprensa.com

SERVICIO TÉCNICO

Todo Programación dispone de una dirección de correo electrónica y un número de Fax para formular preguntas relativas al funcionamiento del CD-ROM de la revista.
e-mail: todoprogramacion@iberprensa.com
Fax: 91 628 09 35

LECTORES

Comparte con nosotros tu opinión sobre la revista, envíanos tus comentarios, sugerencias, ideas o críticas.

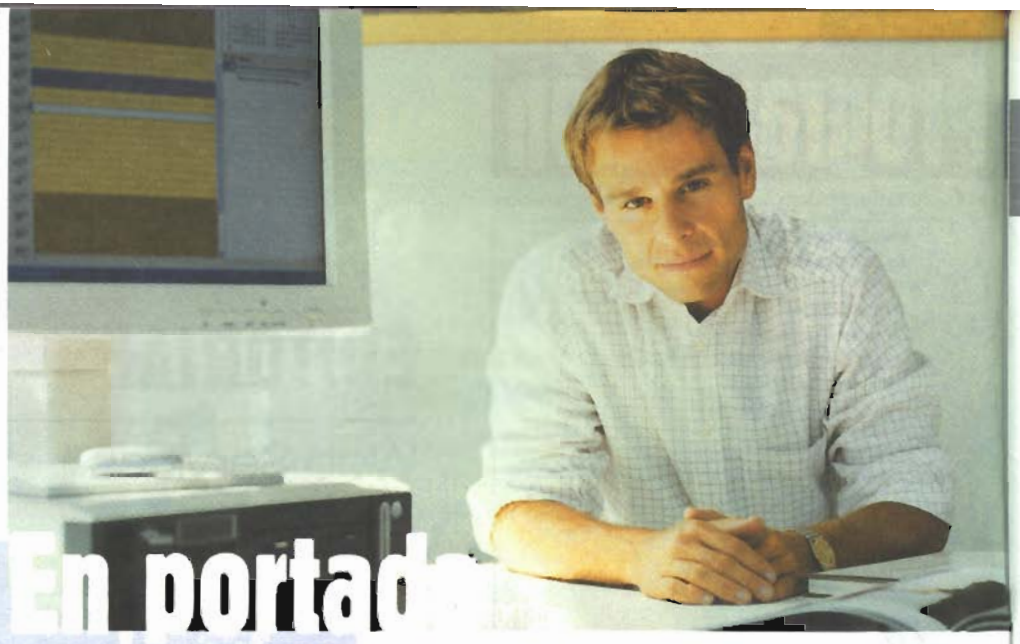
Studio Press
(Todo Programación)
C/ Del Río Ter, Nave 13
Pol. "El Nogal"
28110 Algete. Madrid

DEPARTAMENTO DE PUBLICIDAD

Si le interesa conocer nuestras tarifas de publicidad no dude en ponerse en contacto con nuestro departamento comercial:
- **Tel. 91 628 02 03**
- **e-mail: publicidad@iberprensa.com**

Sumario

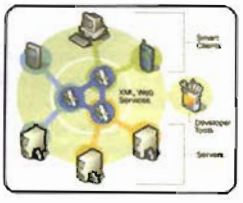
Número 12



En portada

Web Services: introducción y escenarios para su uso

10 Desde hace algún tiempo ha surgido una nueva tecnología que parece estar en todas partes: los servicios web. En el tema de portada de este mes vamos a tratar de definir qué son y su ambito de aplicación. Aprovecharemos también para dar algunas definiciones básicas, mientras en la sección habitual de Mono, dentro de la zona Linux, implementaremos varios ejemplos prácticos.



A quién vamos dirigidos

Todo Programación (TP) es una revista para programadores escrita por programadores y con un enfoque eminentemente práctico. Trataremos de ser útiles al programador, tanto al profesional como al estudiante. Si hay algo cierto en este sector es que nunca podemos parar, vivimos en un continuo proceso de reciclaje. Ahí es donde tratará de encajarse TP: información, actualidad, cursos y prácticas de los lenguajes más demandados y formación en sistemas.

ZONA LINUX

Actualidad: Novedades en lenguajes y compiladores >>

21 Analizamos la actualidad sobre lenguajes y herramientas para programadores Linux. Han sido novedades estos días la publicación de la nueva versión del IDE de Anjuta y algunas noticias relacionadas con los programas para control de versiones.

C#: Introducción a los servicios web >>

24 Enlazamos con nuestro reportaje del mes, ahora vamos a ver un par de ejemplos prácticos para la implementación de servicios web con C#. Aprovecharemos los APIs abiertos y disponibles para el programador de Google y Amazon. Manos a la obra.

GTK#: Diseño de interfaces con Glade >>

28 Finalizamos nuestra serie dedicada a GTK con un ejercicio práctico con Glade. También comentaremos algunas alternativas como Gazpacho, incluyendo una entrevista con el coordinador de dicho proyecto.

Ejemplos y código fuente

Cada CD-ROM de la revista incluye una carpeta denominada *fuentes* en la que se encuentra el material complementario para seguir cada uno de los cursos: los listados completos, los ejemplos desarrollados en diversos lenguajes, compiladores, editores, utilidades y en general, cualquier herramienta que se mencione o cite en la respectiva sección.
Todo con la finalidad de completar la formación y facilitar el seguimiento de cada artículo por parte del lector.

CONTENIDO DEL CD-ROM

Herramientas y recursos para el programador

65 En el CD de **Todo Programación** incluimos este mes una recopilación de herramientas para programadores en ambas plataformas (Windows/Linux), además del codigo fuente y los ejemplos de todos los artículos de la revista. Este número también incluye un especial dedicado a los entornos RAD de desarrollo para Linux, destacando entre todos Glade. También se incorporan diversas utilidades para desarrolladores Java como JDebugTool y ExcelsiorJet.

TALLER PRÁCTICO

Programación robusta en Java >>

36 Escribir código seguro o robusto en Java que ofrezca garantías cuando se produce una excepción no resulta tan sencillo como pueda parecer. Dedicamos este taller práctico a estudiar qué es el código robusto y cómo obtenerlo.

Programación de Grids: Arquitectura del Globus toolkit

41 Globus está muy fuertemente modularizado, lo que significa que no necesitamos dominar todo el API para desarrollar una aplicación que solo requerirá un subconjunto del mismo.

Extendiendo el framework de Qt: El juego Memory

45 Vamos a ver cómo crear nuestros propios widgets y también estudiaremos cómo desacoplar el diseño del código, todo ello mediante la realización de un ejemplo, en concreto el juego de cartas Memory.

Bases de datos en el cliente con JavaScript DB

48 JavaScript DB es un motor de bases de datos creado en JavaScript que permite generar una base de datos en local que puede ser consultada en SQL.

Generics en Java: uso y ejemplos prácticos

52 Los tipos de datos parametrizados o Generics es una de las novedades más importantes añadidas en la última versión de la plataforma Java 2. Vamos a estudiar su uso y funcionamiento con algunos ejemplos prácticos.

ZONA WINDOWS

F#: Programación funcional para .NET >>

32 F# es una adaptación del lenguaje funcional Ocaml a la plataforma .NET. Comenzamos un minicurso de dos entregas en el que introduciremos al lector primero en la programación funcional, para pasar a aprender a programar con esta herramienta.

REPORTAJE: Programación de algoritmos genéticos

16 ¿Qué son los algoritmos genéticos? ¿realmente funcionan o son mejores para solucionar determinado tipo de problemas? Dedicamos el reportaje de interior a este tema tan de moda últimamente, haremos una introducción a genética y evolución, y sus distintos ámbitos de aplicación en la informática.

Y ADEMÁS...

DESARROLLO WEB: sesiones en PHP con PostNuke >>

57 Vamos a analizar cómo gestiona PHP las sesiones mediante un ejemplo práctico. Para ello, vamos a utilizar PostNuke, un sistema de administración de contenidos open source similar a Mambo, el cual estudiamos en nuestro número anterior.

JUEGOS JAVA: Aspectos avanzados >>

61 Hemos estudiado aspectos concretos para el desarrollo de juegos con J2ME, pero hay un tema importante que apenas hemos tocado: la teoría de juegos. Dedicamos este número a abordar aspectos relacionados con dicha teoría en los distintos tipos de juegos.

NOTICIAS

6. **Java** Expo 2005.
6. **Séptima** edición del foro ENSA@WORK.
7. **Sun** renueva su convenio con la UC3M.
7. **JDBC** Manager será integrado en NetBeans.
8. **"Summer** of Code" de Google.
8. **Nuevas** herramientas de desarrollo para Longhorn.
8. **AMD** presenta "Pacifica".
8. **Borland** Day.
9. **Nuevos** servidores Bull NovaScale 9162 y 9322.
9. **Servidor** Sun Fire V40z.
9. **Dell** Precision 380.

NIVEL DE DIFICULTAD:

MATERIAL INCLUIDO EN EL CD-ROM:
db.core.zip ■ Ejemplos prácticos

Bases de datos en el cliente con JavaScript DB

MANUEL DOMÍNGUEZ
mdominguez@iberprensa.com

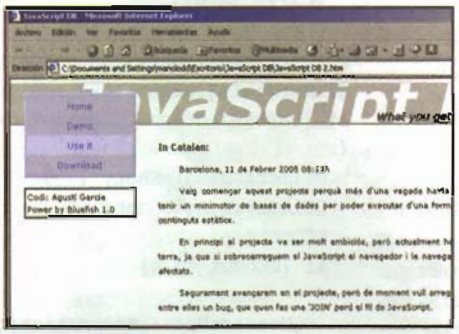
La carga de datos local en una aplicación web mediante el uso de JavaScript siempre ha sido una tarea complicada, por la necesidad de tener que programar en JavaScript una estructura de árbol o lista y un conjunto de funciones para su uso. Con JavaScript DB este proceso se simplifica enormemente, con lo cual como programadores tendremos una cosa menos de la que preocuparnos.

De vez en cuando todas las personas que programamos nos acabamos encontrando con problemas repetitivos que nos traen una y otra vez dolores de cabeza. Problemas lo suficientemente pequeños como para que no se promueva una solución en condiciones para ellos, pero tan frecuentes como para plantearse seriamente si dicha solución debería haber sido creada, y arrepentirnos una vez más de no haber solventado el fallo la última vez que apareció. Y, en ocasiones, nos encontramos de bruces con proyectos que resuelven de forma eficiente, sencilla y rápida estas dificultades.

Es el caso de JavaScript DB, un software creado para dar solución a un problema concreto: la carga de datos en local de una forma sencilla.

FILOSOFÍA DEL PROYECTO

JavaScript DB, como su propio nombre indica, es un motor de bases de datos (podríamos asemejarlo a un SGBD) íntegramente creado en JavaScript, que permite, desde el momento en que se utiliza en una página web, crear una base de datos en local con sus tablas, campos... y se puede consultar dicha base de datos mediante SQL. En principio, Agustí García, el autor, pretendía ser muy ambicioso y crear un motor que permitiese cualquier operación SQL, pero el proyecto ha quedado bastante reducido en cuanto a este propósito, según el propio autor *"porque sobrecargar a los navegadores con demasiado código JavaScript puede afectar negativamente a la navegación"*. Así que el proyecto mantiene toda la funcionalidad que realmente se puede necesitar en el contexto del que estamos hablando, y omite aquella que en un principio se pensaba incluir, pero que a posteriori no resultaba rentable. Pero... ¿qué utilidad puede tener? En las siguientes líneas veremos situaciones en las que JavaScript DB puede venirnos bien.



Sitio web del proyecto.

EL API DE JAVASCRIPT DB

El primer paso para usar JavaScript DB en una web es incluir la "librería" JavaScript correspondiente para ser usada. Esto es tan sencillo como añadir la siguiente línea en el fichero HTML:

```
<script type='text/javascript'
src='db_core.js'>
```

Donde *db_core.js* es el fichero que incluye toda la funcionalidad y que, para colmo, solo ocupa 3 KB, por lo que la web no se verá afectada por tener que cargar gran cantidad de datos. Podemos descargar *db_core.js* de la web del proyecto que se encuentra albergada en Source Forge (es software libre), en la URL *http://jsdb.sourceforge.net* o también en el CD-ROM de este número.

Así que realmente no hay instalación como tal, sino la inclusión de un fichero de código JavaScript en una página, del mismo modo que con cualquier otro código. Una vez realizado este paso, ya podemos usar los métodos definidos en la librería, que detallaremos a continuación.

Crear la base de datos

Crear una base de datos con JavaScript DB, en memoria por supuesto, es realmente sencillo. Basta con instanciar un objeto mediante *new*, como es habitual. Por ejemplo, si queremos crear una nueva base de datos llamada *BDEmpresa*, tendríamos que introducir la siguiente línea de código:

```
var BDEmpresa =
new Database("BDClientes");
```

Y a partir de entonces deberemos usar métodos de *BDEmpresa* para todas las operaciones que queramos crear sobre esta "base de datos". Lo que hemos hecho con la línea anterior es crear en memoria la estructura de la base de datos; un objeto que contiene un *array* de tablas, donde cada tabla a su vez es un objeto que contiene un *array* de registros. Además de todo ello, cada uno de los objetos tiene sus métodos y atributos (o propiedades).

Crear tablas

Ahora que ya tenemos la base de datos *BDEmpresa* creada, podemos hacer uso de uno de sus métodos para comenzar a generar tablas dentro de ella. Para ello debemos utilizar el método *CreateTable(Nombre, Array)*. El primero de los parámetros que acepta este método es el nombre de la tabla que queremos crear, y el segundo es un vector de cadenas de caracteres que son los nombres de los distintos campos (o columnas) de la tabla. Así, continuando con el ejemplo anterior donde habíamos creado una base de datos llamada *BDEmpresa*, podríamos crear una tabla como sigue:

JavaScript DB es un motor de bases de datos que crea una base de datos local con tablas, campos...

```
BDEmpresa.CreateTable("Clientes",
Array("id","nombre","DNI"));
```

lo cual generará en la base de datos *BDEmpresa* una tabla llamada *Clientes* donde cada registro almacenará un identificador de cliente, su nombre y su DNI. Como vemos, para la descripción de datos en JavaScript DB no se usa SQL sino métodos del objeto que representa la base de datos.

Insertar datos

La inserción de datos se realiza usando el método *Insert(Tabla, ArrayDatos)* de la base de datos. En este método, *Tabla* es una cadena de caracteres que especifica el nombre de la tabla de la base de datos, tal como se especificó en el método *CreateTable*, y *ArrayDatos* es un vector con los valores que se desean insertar en la tabla y que deben coincidir con el número de campos especificados durante la creación de la tabla. Siguiendo con el ejemplo que hemos comenzado en los puntos anteriores, supongamos la siguiente línea de código JavaScript:

```
BDEmpresa.Insert("Clientes",
Array(1, "Manuel D.", "12345678P"));
```

El resultado de la ejecución de esta línea haría que en la tabla *Clientes* de la base de datos *BDEmpresa* se insertara un nuevo registro para el cliente *Manuel D.*, con identificador de cliente 1 y el DNI 12345678P. Como se puede observar, por ningún lado se especifica el tipo de datos de cada campo. Esto es usual en los lenguajes de programación web, porque el tipo de las variables queda definido por el contexto en que se usa (por el valor que se le asigna).

Realizar consultas

Con lo visto hasta ahora podemos crear una base de datos con sus tablas e insertar en éstas los registros que deseemos. Lo habitual en el contexto en el que se usará normalmente JavaScript DB es que la carga de datos se realice desde un fichero JavaScript, de forma estática. No olvidemos que el entorno de ejecución será el navegador del cliente, por lo que no será necesario la mayor parte de las veces la actualización de datos, la inserción de datos de forma dinámica, el borrado de registros, etcétera. De hecho, estos cambios dejarían de existir en el momento en que el usuario volviese a cargar la web en su navegador. Sin embargo, lo que sí es usual, y es lo que proporciona flexibilidad y potencia a JavaScript DB, es la consulta de los datos que ya están insertados en las tablas. Realmente podríamos entender que la base de datos es una forma de almacenar los datos y consultarlos cómodamente en SQL, sin árboles, sin pilas, colas... Para ello debemos utilizar el método *Select(ConsultaSQL)* de la base de datos. Este método acepta una cadena de caracteres, que será la consulta SQL que deseamos hacer sobre las tablas de la base de datos; y a continuación devuelve una matriz bidimensional que contendrá los registros de la base de datos que sean resultados de la consulta, cada uno de ellos con los campos que se hayan seleccionado.

Volvamos a nuestro ejemplo: Supongamos que en la base de datos queremos consultar aquellos clientes con identificador "1". Las líneas de código que tendríamos que programar serían las siguientes:

```
var resultado;
resultado = BDEmpresa.Select
('select * from Clientes where id=1');
```

Lo cual es significativamente más sencillo y cómodo que recorrer un árbol y, además, permite que cualquier persona acostumbrada a la sintaxis de SQL sea capaz en poco tiempo de realizar consultas sobre los datos sin conocer demasiados aspectos de cómo están almacenados. En la variable *resultado* tendremos una matriz que en este caso solo contendrá un registro con todos los campos del cliente que hemos insertado líneas arriba.

Ahora bien... ¿qué tipo de consultas permite el método *Select* de JavaScript DB? Hasta la fecha de elaboración de este artículo, el autor ha implementado consultas tradicionales con *WHERE, ORDER BY, LIKE*, de todo los campos o algunos solo, *JOIN, JOIN* múltiples, *LEFT JOIN* y *RIGHT JOIN*, etc Es decir, más que de sobra para lo que utilizará normalmente este API.

Mostrar resultados

Generalmente, una vez que hemos realizado una consulta SQL a una base de datos, lo que necesitamos es mostrar los resultados. Como hemos visto, JavaScript DB devuelve los resultados en forma de matriz y para mostrarlos habría que ir recorriendo dicha matriz y construir una tabla HTML que se encargara de ello. Realmente no es necesario, porque la API de este proyecto incorpora un método que hace precisamente esto; es el método *View(Resultado, Documento)* de la base de datos, donde *Resultado* es la matriz que se obtiene tras la realización de una consulta mediante el método *Select(...)* y *Documento* es, valga la redundancia, el documento HTML donde se debe mostrar la salida. De este modo, por ejemplo, para mostrar los resultados obtenidos en el ejemplo que vamos siguiendo, habría que escribir el código:

```
BDEmpresa.View(resultado, document);
```

Queda claro que *document* es el típico objeto existente en *JavaScript* y sobre el cual podemos escribir con métodos como *document.write(...)*

Tabla 1. Resumen de la API

MÉTODO	PARÁMETROS	DESCRIPCIÓN
▶ CreateTable	**Nombre:** Nombre de la tabla. **Array:** Vector conteniendo los nombres de los campos.	Crea una nueva tabla en la base de datos.
▶ Insert	**Nombre:** Nombre de la tabla. **Array:** Vector conteniendo los datos a insertar.	Inserta un nuevo registro en una tabla de la base de datos.
▶ Select	**ConsultaSQL:** Cadena de texto. La consulta en SQL que se desea hacer a la base de datos.	Consulta datos de las tablas de la base de datos.
▶ View	**Resultado:** Matriz bidimensional con los registros consultados y su campos. **Documento:** objeto document, de JavaScript, donde se desea mostrar el resultado.	Muestra automáticamente el resultado de una consulta en una página HTML.

QUÉ NO PERMITE JAVASCRIPT DB

Y con lo anterior termina la explicación del API de JavaScript DB. Por tanto, se ve que ni hay operación *UPDATE*, ni *DELETE* ni otras similares. Como dijimos, JavaScript DB elimina todo lo que no es necesario para no sobrecargar de JavaScript el navegador en un entorno local, donde, en principio, los datos no van a ser actualizados sino que van a ser estáticos en el cliente, las operaciones que se usarán en el 99 por ciento de los casos serán las de inserción y búsqueda. La primera de ellas para poder realizar la carga de los datos, y la segunda para acceder a ellos de forma cómoda, eficiente y sencilla.

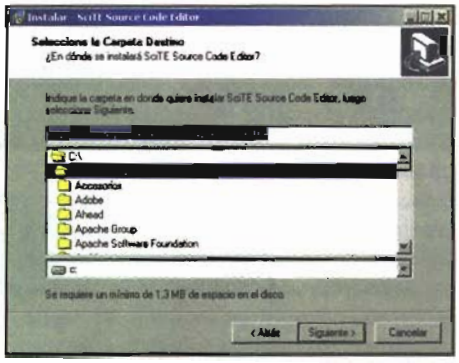
Los acostumbrados a gestores de bases de datos "de envergadura" pensarán que quizás JavaScript DB puede incorporar restricciones de integridad, posibilidad de programación de procedimientos almacenados o disparadores, u otras características avanzadas como las consultas anidadas, vistas, roles... pues rotundamente no. JavaScript DB es un software pequeño con funcionalidades muy reducidas, extremadamente útil para ocasiones muy específicas y, por las pruebas que hemos realizado, bastante estable y depurado, pero en ningún caso es, ni pretende ser, un sistema gestor de bases de datos; y ése es un concepto que hay que entender desde el principio.

POSIBLES ESCENARIOS PARA SU USO

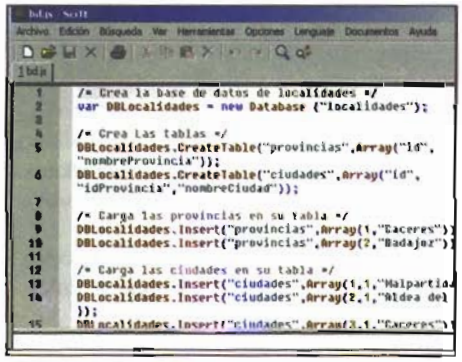
Dónde usar esta "librería" es el primer problema con el que puede encontrarse el programador. La verdad es que un API de base de datos que no permite que los cambios se almacenen, y que además generalmente cargará un número fijo de registros, no parece una buena opción. Sin embargo, nada más lejos de la realidad. Supongamos una web como Infojobs donde se nos permite introducir nuestro curriculum vitae. Imaginemos que entre los datos a introducir se encuentran el país, la provincia y la población. No parece lógico que si elegimos como país España, nos permita seleccionar como población Londres, por ejemplo. En estas ocasiones se suelen tener un número fijo de datos en memoria, y seleccionar solo los que queremos mostrar en las listas desplegables. Además, a medida que seleccionamos valores, por ejemplo, el país y la provincia, los datos a mostrar en la lista de poblaciones se restringen enormemente.

En este caso podríamos tener los datos en un árbol, y recorrerlo según van ajustándose los parámetros introducidos por el usuario. Pero el programador avanzado advertirá que usar JavaScript DB en lugar de un árbol en memoria sobre el que tengamos que realizar recorridos, es mucho más cómodo y sencillo, al permitirnos realizar la consulta en SQL.

Veamos otro caso. Imaginemos que tenemos un listado de productos pocos cambiantes, para que los clientes al visitar nuestra web puedan ver datos de ellos. Por ejemplo, coches y motocicletas antiguos, de coleccionista. Se podría mostrar un primer cuadro donde el usuario pudiese seleccionar entre motocicletas o automóviles, un segundo que permita elegir la marca y otro más para escoger el año de fabricación. Así, en refinamientos sucesivos, se guiará al visitante hasta la hoja de datos técnicos del modelo solicitado. Ambos ejemplos los desarrollaremos en las siguientes líneas. Este tipo de casos es muy común, puesto que en los formularios de entrada se intenta evitar a toda costa que el usuario deba introducir a mano los datos para evitar errores y en su lugar se le proporciona cuadros de selección, listas desplegables, etc con los valores posibles.



Instalación del editor de código fuente Scite.



Trabajando con el editor Scite.

PREPARACIÓN DEL ENTORNO

Para realizar los dos ejemplos, vamos a preparar un entorno de desarrollo adecuado. Necesitaremos un navegador web con capacidad de ejecución de guiones en JavaScript y un editor. Como editor utilizaremos Scite, un programa libre que colorea la sintaxis para multitud de lenguajes de programación y con versiones para Windows y Linux/UNIX. Como navegador utilizaremos cualquiera de los existentes en la actualidad, pues todos soportan JavaScript sin problemas.

La versión 1.54 de Scite se puede descargar de la web *http://www.scintilla.org* y cuenta con instaladores para Windows en castellano. La instalación se reduce a un par de clics de ratón sobre el ejecutable de instalación y seguir las sencillas instrucciones que aparecen.

EJEMPLOS PRÁCTICOS

Una vez instalado un navegador (con el que seguro ya contaréis en vuestro PC), y tras instalar Scite, dispondremos de un potente editor para programadores y podremos comenzar a desarrollar los ejemplos.

Provincias y ciudades

En este primer ejemplo, haremos uso de JavaScript DB para montar un sistema de selección del municipio para, por ejemplo, utilizarlo en una página de introducción de currículum. La idea es tener una base de datos con dos tablas, una para las provincias y otra para las ciudades. En la página web que permite la selección, habrá una lista desplegable con las provincias, y cuando se seleccione una de ellas, la lista de selección de la ciudad se rellenará con ciudades pertenecientes a dicha provincia.

Harán falta un par de funciones. La primera de ellas responderá ante un evento *onChange* de la lista de selección de provincias, y la segunda ante un evento *onChange* de la lista de selección de ciudades, para pasar a mostrar al usuario un mensaje de confirmación. En el **Listado 1** de la página siguiente podemos observar la función JavaScript que, haciendo uso del API de JavaScript DB, rellena la segunda lista de selección (de ciudades).

Captura del ejemplo 1.

La función *consultarCiudades()* obtiene el valor de la lista de selección de provincias para construir una consulta SQL que lanza contra la base de datos de JavaScript DB. De este modo obtiene el identificador de provincia que deben tener como clave externa todas las ciudades que buscamos. Posteriormente realiza una consulta a la base de datos y recorre el resultado, formando simultáneamente la segunda lista de selección.

Por su parte, la función *formarComentario()*, que podemos observar en el **Listado 2** de la página siguiente, actúa tras un cambio en la lista de selección de ciudades y se encarga de recolectar los datos de la listas de selección del formulario de entrada, formar un mensaje indicativo dirigido al usuario y mostrarlo en pantalla en el cuadro de texto correspondiente.

En el código fuente, que se encuentra en el CD de la revista, se podrá ver de forma más directa cómo se realiza la carga de datos y cómo se invocan desde HTML las funciones ne-

cesarias. Lo cierto es que es bien sencillo, y en pocas líneas se tendrá montado todo el sistema. Para ejecutar el ejemplo, solo será necesario un navegador con soporte de JavaScript y hacer doble clic en el fichero HTML.

Por supuesto, es imprescindible usar código JavaScript para programar la funcionalidad de este ejemplo, pero es significativamente menos trabajoso para el programador que usar un árbol binario que debería recorrer para realizar búsquedas. JavaScript DB nos abstrae por completo del almacenamiento de los datos y con SQL podemos consultarlos fácil y dinámicamente.

Búsqueda de modelo de vehículo

En este segundo ejemplo usaremos JavaScript DB para montar un sistema que permita mostrar productos, concretamente vehículos, para, por ejemplo, usarlo en una página de compraventa de coches de ocasión. Se trata de tener una base de datos con un par de tablas, una para los tipos de vehículos y otra para los modelos concretos, con información muy detallada de los mismos. En la página web que permite la selección, habrá una lista desplegable con los tipos de vehículos y cuando se seleccione uno de ellos, la lista de selección del modelo se rellenará con los modelos de vehículos que pertenecen a esos tipos.

La función *consultarModelos()* la podemos encontrar en el código fuente del ejemplo 2 (CD-ROM). Obtiene el valor seleccionado de la lista de selección de tipos de vehículos (básicamente vehículos de carga o turismos) para construir una consulta SQL que lanza contra la base de datos de JavaScript DB. Así obtiene el identificador del tipo de vehículo, que ha de coincidir con el que deben tener como clave externa todos los modelos de vehículos que buscamos.

Listado 1. Selección de ciudades

```
function consultarCiudades() {
  document.forms.formulario.comentario.value="Esperando selección..."
  if (document.forms.formulario.selectProvincia.value == "nada") {
    document.forms.formulario.selectCiudad.length=0;
    document.forms.formulario.selectCiudad[0]=new Option("- Seleccione la
    ciudad -","nada");
  } else {
    var consProvincia = 'select id from provincias where
    nombreProvincia="'+document.forms.formulario.selectProvincia.value+'"';
    var rconsProvincia = DBLocalidades.Select(consProvincia);
    var idBuscado=rconsProvincia[0];
    var consCiudades = 'select nombreCiudad from ciudades where
    idProvincia='+idBuscado;
    var rconsCiudades = DBLocalidades.Select(consCiudades);
    document.forms.formulario.selectCiudad.length=0;
    document.forms.formulario.selectCiudad[0]=new Option("- Seleccione la
    ciudad -","nada");
    for (var i=0;i<rconsCiudades.length;i++) {
      document.forms.formulario.selectCiudad[i+1]=new
      Option(rconsCiudades[i],i+1);
    } } }
```

Listado 2. Imprimir comentario

```
function formarComentario() {
  if (document.forms.formulario.selectCiudad.value == "nada") {
    document.forms.formulario.comentario.value="Esperando selección..."
  } else {
    var cElegida = document.forms.formulario.selectCiudad.value;
    var txt1 = "Usted vive en ";
    var txt2 = document.forms.formulario.selectCiudad[cElegida].text;
    var txt3 = ", (";
    var txt4 = document.forms.formulario.selectProvincia.value;
    var txt5 = ").";
    document.forms.formulario.comentario.value=txt1+txt2+txt3+txt4+txt5;
  } }
```

Visualización en navegador del ejemplo 2.

Posteriormente realiza una consulta a la base de datos y recorre el resultado, formando simultáneamente la segunda lista de selección, donde podremos escoger entre una lista de modelos concretos de vehículos del tipo seleccionado.

Captura del ejemplo 2 al seleccionar un modelo.

Para finalizar, la última de las funciones importantes del ejemplo es la llamada *mostrarDetalles()*, cuyo cometido es obtener de la base de datos de JavaScript DB todos los datos existentes para el vehículo escogido en la lista de selección de modelos, y mostrarlos en el navegador del usuario en un formato legible y amigable.

De nuevo se pueden consultar los fuentes en el CD de la revista y allí, en el código comentado, se encuentra detallado cómo cargar los datos y dónde se debe llamar a las funciones auxiliares en JavaScript que permitirán el buen funcionamiento de los ejemplos.

CONCLUSIONES

Una vez más comprobamos como en el mundo del software libre existen cada vez más soluciones para todo tipo de problemas; algunas extremadamente complejas y otras relativamente sencillas pero mucho más útiles para el día a día en el desarrollo de una aplicación. Éste es el caso del software JavaScript DB. En cualquier caso, lo importante es contar con alternativas que nos permitan crear software de calidad y JSDB es firme candidato a ser ese tipo de solución. Feliz programación.

1ª entrega, CD-ROM y Presentación de la obra por solo 5,99 euros

Vídeodigital

El primer curso para usuarios de cámaras digitales

- Aprende a trabajar con los programas de edición de vídeo más importantes.
- Cómo grabar, editar y producir vídeo digital paso a paso.
- Saca provecho a la cámara digital y al PC.
- No se requieren conocimientos previos para seguir esta obra.

Composición de la obra

20 coleccionables

divididos en:

Teoría
Sección donde se aborden las técnicas de grabación, el funcionamiento de las cámaras, el estudio de las principales aplicaciones profesionales de edición y en general todo lo relacionado con este campo.

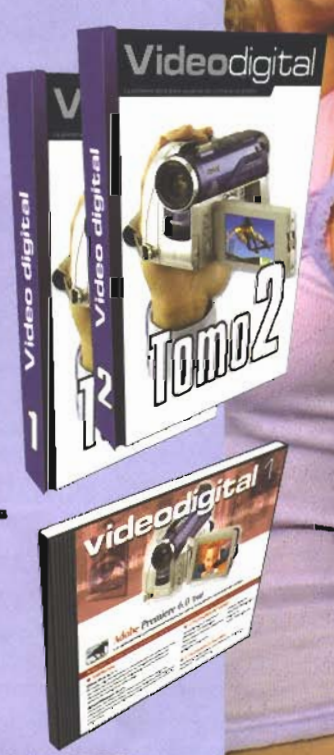
Práctica
Ejercicios para aprender técnicas de trabajo, cómo utilizar Adobe Premiere a la perfección y cómo montar los primeros clips de vídeo propios.

Tutorial
Áreas donde se complementa la parte práctica con el estudio de herramientas adicionales, como son: conversores, codecs, programas de retoque y efectos, aplicaciones de edición de audio, y todo tipo de utilidades relacionadas con la producción de vídeo.

Contenido CD-ROM
Descripción del contenido del CD-ROM y explicación de cómo instalar el software.

20 CD-ROMs

Todo el software actual para la edición de vídeo, demos de programas profesionales, aplicaciones completas, utilidades, etc.



Todos los CD-ROMs incorporan un asistente para mayor comodidad en la búsqueda de contenidos e instalación de programas.

Studio PRESS

C/ del Río Ter, Nave 13
Polígono Industrial "El Nogal"
28110 Algete (Madrid)
Tel.: 916280203
Fax: 916280935
e-mail: video@iberprensa.com
www.iberprensa.com

ya a la venta **semanalmente** en quioscos y centros comerciales