Año 1 • Número 3 • **5,98** euros

TODO Programación

La Revista mensual para entusiastas de la programación

www.iberprensa.com

Herramientas IBM

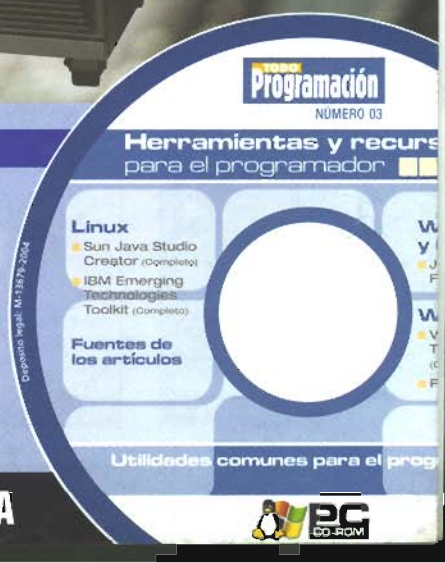
Introducción a JavaServer Faces

Programación práctica de plugins en Eclipse

Novedades en JbuilderX y FileMaker Pro 7

8 413042 751977 00003
Studio PRESS

■ EN EL CD-ROM: ESPECIAL HERRAMIENTAS IBM Y JAVA PARA WINDOWS/LINUX



Zona Linux

- Las declaraciones en el lenguaje C#
- Estado actual del proyecto Mono

Zona Windows

- Programación de un ejemplo con ASP. NET
- Delphi: programación de componentes visuales

CURSOS PRÁCTICOS DE C, PYTHON, SQL Y PROGRAMACIÓN BÁSICA

Programación

La Revista mensual 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)
Fernando Sánchez
(fsanchez@iberprensacom)
Fernando Escudero
(fescudero@iberprensa.com)
José Manuel Navarro
(jnavarro@iberprensa.com)
Marcos Prieto
(mprieto@iberprensa.com)
Guillermo "el Guille" Som
(elguille@iberprensa.com)
Luis Martín Caballero
(lmartin@iberprensa.com)
Manuel Domínguez
(mdominguez@iberprensa.com)
Lorenzo Gil
(lgil@iberprensa.com)
José Rivera
(jrivera@iberprensa.com)
Jaime Anguiano
(janguiano@iberprensa.com)
Javier Cano
(jcano@iberprensa.com)

DISEÑO PORTADA
Antonio Gª Tomé

MAQUETACIÓN
Antonio Gª Tomé

DIRECTOR DE PRODUCCIÓN
Carlos Peropadre
cperopadre@iberprensa.com

SUSCRIPCIONES
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, 7. 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 03 • Año 1

EDITORIAL

Eduardo Toribio

Herramienta de consulta

Bienvenidos a un nuevo número de **Todo Programación**. Este mes creo que os entregamos un número muy completo, primero por los artículos prácticos donde vemos desde cómo desarrollar plugins en Eclipse pasando por la programación de componentes visuales en Delphi hasta un ejemplo práctico con JavaServer Faces.

Además el análisis, hacemos un gran recorrido por las principales herramientas de desarrollo de IBM y luego analizamos productos de actualidad como JbuilderX o FileMaker 7. Y por supuesto las secciones habituales; SQL, Python, programación para principiantes, ASP.NET y la zona dedicada a desarrollo Linux, en concreto a Mono. Por otra parte, observad que todos los contenidos del CD-ROM guardan relación directa con los artículos y reportajes del mes, de esta forma convertimos la revista en una herramienta de consulta y de apoyo para el programador actual.

Sumario

Número 3

En portada

Herramientas IBM

10 IBM, con una historia basada tradicionalmente en hardware, se ha convertido en una compañía que dispone ahora de una enorme oferta de productos para desarrollo de software y en diversas plataformas. Dedicamos el cover del mes a realizar un recorrido por ellas y estudiar en lo sustancial la filosofía y posicionamiento de IBM en un momento tan crucial como el actual.

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.

.net ZONA WINDOWS

Borland JbuilderX >>

24 Jbuilder es la apuesta de Borland para el desarrollo de aplicaciones empresariales en Java, una plataforma que en la actualidad soporta aplicaciones críticas de todos los tipos. Analizamos las novedades de esta nueva versión.

Filemaker 7: muchas novedades >>

27 Filemaker es una de las bases de datos de mayor solera en el mundo Mac, en las últimas versiones ha ido ganando popularidad entre los usuarios Windows. Veámos que puede aportar esta nueva versión 7 que no nos aporten otras soluciones.

ASP.NET >>

30 Llegamos a la tercera y última entrega de nuestro minicurso, ASP .NET. En ella vamos a construir una aplicación web completa de ejemplo para ilustrar todo lo aprendido hasta la fecha.

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: por ejemplo, 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

64 En este número 3 fuerte contenido Java para ambas plataformas con Sun Java Creator y la edición Foundation de JbuilderX. Pero si lo que te interesa es la programación visual en C++ para Windows y .NET incluimos Visual C++ Toolkit 2003. Completan el CD alguna demo como la de FileMaker 7, fuentes y utilidades relacionadas con los artículos y aplicaciones de interés general.

TALLER PRÁCTICO

Desarrollo de componentes Delphi >>

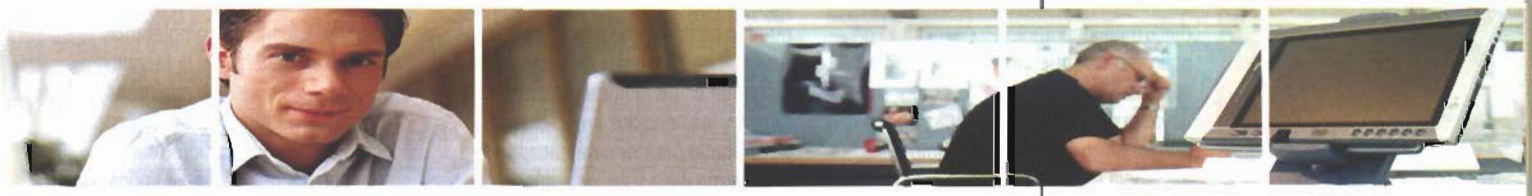
Programación de plugins con Eclipse 2.1 >>

Introducción a Java Server Faces >>

Y ADEMÁS...

C para Hackers: Pilas y listas enlazadas >>

Desarrollo Web: Python >>



ZONA LINUX

Actualidad Desarrollo >>

Mono: Declaraciones y C# >>

Bases de datos: Profundizando en SQL >>

NOTICIAS

CUADERNOS DE PRINCIPIANTES

Estableciendo la secuencia >>

NIVEL DE DIFICULTAD:

MATERIAL INCLUIDO EN EL CD-ROM:
Código fuente del ejemplo desarrollado

Aplicaciones web ASP

MANUEL DOMÍNGUEZ
mdominguez@iberprensa.com

El acceso a datos para almacenamiento y recuperación, el paso de parámetros al software web y la caracterización del usuario nos permiten, al fin, poder crear auténticas aplicaciones web, programas complejos accesibles a todos los usuarios desde el navegador, el cliente de acceso universal a contenidos a través de Internet. En este artículo construiremos una aplicación web completa.

PASO/RECUPERACIÓN DE ARGUMENTOS VIA URL

De poco sirve el hecho de disponer en VBScript de estructuras condicionales que nos permitan tomar decisiones, si las variables en las que se basan no pueden ser definidas en tiempo de ejecución. Por ejemplo, sería interesante que una misma página pudiese mostrar un saludo personalizado dependiendo de qué usuario se tratase, y que el nombre del usuario se pudiese pasar de alguna forma al código ASP desde fuera. Como hemos visto a lo largo de este curso, ASP proporciona objetos que permiten obtener y enviar información. En este caso, el objeto *Request* y su método *QueryString (nombre)* nos permiten obtener valores que se hayan pasado al fichero **.asp* en línea de comandos o, para ser más puristas, vía URL.

Así, si tenemos un fichero llamado *SaludoPorURL.asp*, y lo invocamos desde el navegador como *http://localhost/SaludoPorURL asp?nombre=pepe*, le estaremos diciendo en tiempo de ejecución que la variable *nombre* tiene el valor "pepe" y para acceder a dicha variable se ha de hacer desde dentro de *SaludoPorURL.asp* mediante el uso de *Request.QueryString("nombre")*. En este caso este método devolverá "pepe".

En el Listado 1 se encuentra el código de este ejemplo. Para probarlo, cópialo en un nuevo fichero del HAP Edit 3.0 y guárdalo en *C:\Inetpub\wwwroot* con el nombre *SaludoPorURL.asp*. Para probarlo teclea *http://localhost/SaludoPorURL.asp?nombre=texto* donde *texto* puede ser cualquier nombre que se te ocurra.

Listado 1. SaludoPorURL

```
<html>
<h1><center>Ejemplo SaludoPorURL.asp</center></h1><br><br>
<h3><center>
<%
  NombreEspecificado = Request.QueryString("nombre")
  TextoIzquierda = "Hola "
  TextoDerecha = ", ¿qué tal estás?"
  Response.Write(TextoIzquierda & NombreEspecificado & TextoDerecha)
%>
</center></h3><br><br><br>
</html>
```

Ejecución del ejemplo SaludoPorURL.asp.

Además se pueden pasar más de un parámetro al fichero **.asp* en la misma URL separándolas mediante el carácter *&*. El formato general es: *http:// /fichero asp? variable1=valor1&variable2=valor2* . Y, claro, posteriormente en *fichero.asp* deberemos hacer tantas llamadas a *Request QueryString(...)* como variables queramos consultar

Listado 2. SaludoPorFormulario

```
<html>
<h1><center>Ejemplo SaludoPorFormulario.asp</center></h1><br><br>
<h3><center>
<%
  NombreEspecificado = Request.Form("nombre")
  TextoIzquierda = "Hola "
  TextoDerecha = ", ¿qué tal estás?"
  Response.Write(TextoIzquierda & NombreEspecificado & TextoDerecha)
%>
</center></h3><br><br><br>
</html>
```

PASO/RECUPERACIÓN DE ARGUMENTOS VÍA FORMULARIOS

También se pueden enviar variables haciendo uso de formularios web. Para

ÍNDICE MINI-SERIE ASP

ASP proporciona objetos que permiten obtener y enviar información

ello debemos diseñar los formularios con un poco de cuidado e indicar que, una vez enviado el formulario, lo debe procesar un fichero **.asp*, el que nosotros programemos para este fin. Haremos uso de nuevo del objeto *Request* de ASP pero el método usado ahora será *Form*, aunque funciona exactamente igual que *QueryString*. *Request.Form(nombre)* nos devolverá el valor de la variable *nombre* siempre y cuando ésta se haya mandado desde un formulario.

En el **Listado 2** de la página anterior se encuentra el código necesario para recoger una variable llamada nombre que se haya enviado desde un formulario. Copia el código en *HAP Edit 3.0* y guárdalo en *C:\Inetpub\wwwroot* con el nombre *SaludoPorFormulario.asp*. No ejecutaremos este fichero directamente porque el único dato que podemos pasarle, debemos hacerlo a través de un formulario y no desde la URL. Simplemente vemos que el código es el mismo que el del fichero *SaludoPorURL.asp* salvo por que ahora se usa *Request.Form* en lugar de *Request.QueryString*.

Para enviar desde un formulario el nombre, vamos a crear otro fichero, cuyo código puedes observar en el **Listado 3** y que básicamente es un formulario que presenta un listado de texto cuyo nombre es "nombre" (el mismo nombre que la variable que queremos enviar) y que especifica que el fichero que recogerá los datos será *SaludoPorFormulario.asp*, el fichero que hemos creado antes. Guárdalo en *C:\Inetpub\wwwroot* con el nombre *FormularioDeSaludo.asp*. Ahora, pruébalo escribiendo en el navegador *http://localhost/FormularioDeSaludo.asp* y verás una caja de texto y un boton de envío. Al rellenarlo y pulsar el botón, el fichero *SaludoPorFormulario asp* se ejecutará, recibirá el nombre y lo mostrará

Igual que antes, podemos pasar a través de un formulario varias variables; simplemente debemos hacer un formulario

Entrada de datos a través de un formulario.

con más cajas de texto, áreas de texto, botones . Cada uno de los elementos que componen el formulario deberá tener un nombre (la propiedad *name*) y ese nombre será el de la variable que se podrá consultar en el fichero que recibe los datos.

MANEJO DE COOKIES

Las cookies son pequeñas porciones de texto que se pueden almacenar desde una página web en el ordenador del visitante con la intención, generalmente, de personalizarle la página sin tener que pedirle muchas veces que introduzca la información. Una cookie se caracteriza por un nombre, un valor y una fecha de caducidad, así podemos introducir una cookie llamada *Idioma* con el valor "Castellano" y fecha de caducidad "25/07/2008" en el ordenador de un visitante y cuando vuelva a visitar nuestra página, podremos consultar esa información.

Tanto el objeto *Request* como el objeto *Response* de ASP tienen un método *Cookie* que permite insertarlas o consultarlas.

```
Request.Cookies(nombre_cookie)
```

Nos devolverá el valor de la cookie especificada si ésta está almacenada en el ordenador del visitante y no ha caducado. Si no, devolverá una cadena vacía.

```
Response.Cookies(nombre cookie) =
"valor"
```

De esta forma, estamos colocando una cookie llamada *nombre_cookie* en la máquina del visitante de nuestra web. Además, damos a esa cookie el valor *valor*.

```
Response.Cookies(nombre cookie).
Expires = "2/3/2006"
```

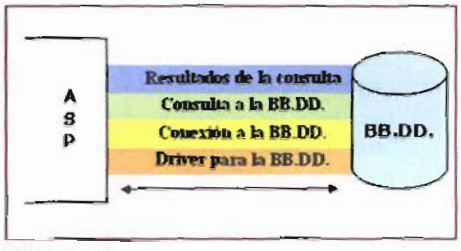
Así estamos diciendo que la cookie llamada *nombre_cookie* que hemos colocado en el ordenador del visitante caducará (se eliminará automáticamente del ordenador del visitante) el 2 de Marzo de 2006.

Realmente no hace falta decir mucho más de las cookies puesto que su manejo es trivial, y una vez se conoce el uso de los objetos *Response* y *Request* es todo muy similar.

CONEXIÓN A BASES DE DATOS

ASP tiene métodos para acceder a bases de datos desde páginas web, usando SQL, lo que nos dotará de todo lo necesario para crear auténticas aplicaciones ejecutables vía web.

Para acceder a una base de datos, debemos establecer una conexión con la misma. Para ello necesitamos que el servidor nos proporcione una conexión a una base de datos, ya veremos cómo. Necesitaremos especificar qué tipo de base de datos vamos a usar (en nuestros ejemplos serán bases de datos Access) para que IIS sepa cómo debe usarla, y una vez que tenemos la conexión con la base de datos establecida, realizaremos peticiones (consultas) y utilizaremos los datos que recibamos.



Capas necesarias para consultar una base de datos.

Tanto el lenguaje SQL (que se usará para consultar la base de datos) como Access, exceden al objetivo de este artículo, por lo que supondremos que el lector tiene ya estos conocimientos. Vamos a ver cómo conectar nuestra web dinámica a una base de datos en el siguiente ejemplo, de forma más práctica y clara.

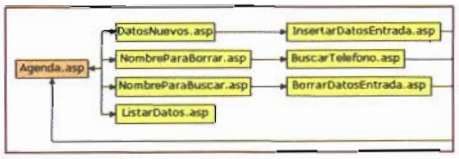
Listado 3. FormularioDeSaludo

```
<html>
<h1><center>Ejemplo FormularioDeSaludo.asp</center></h1><br><br>
<h3><center>
<form method="post" action="SaludoPorFormulario.asp">
      ¿Como te llamas? <input name=nombre>
      <br><br><input type="Submit" value="Contestar">
</form>
</center></h3><br><br><br>
<center>:: Formulario inspirado en código de CoRSA ::</center>
</html>
```

EJEMPLO: GUÍA DE TELÉFONOS PASO A PASO

Con lo que hemos visto a lo largo de este minicurso, tenemos los conocimientos mínimos necesarios para realizar aplicaciones web completas. Así que para poner en práctica lo aprendido, vamos a crear una aplicación web para almacenar, consultar, listar y borrar números de teléfono de personas de una base de datos Access. Un ejemplo sencillo pero completo en el que usaremos muchos de los conocimientos que hemos aprendido

Esta aplicación web va a estar formada por ocho ficheros **.asp* y uno **.mdb*. El fichero *mibd.mdb* es una base de datos Access que ahora definiremos. En cuanto a los fichero ** asp*, uno de ellos será la página inicio de la aplicación que hará las veces de menú Tres archivos más recogerán a través de formularios información para pasarla a otros ficheros que la utilizarán. De tal modo que al final tendremos todos los ficheros interconectados formando una aplicación única En el CD-ROM que acompaña a la revista, puedes encontrar todos los ficheros de este ejemplo listos para ser usados; aquí solo comentaremos los que tienen más relevancia. Vamos a suponer para todos los ficheros de este ejemplo, que los guardamos en *C.\Inetpub\wwwroot*.



Estructura de páginas de la aplicación web AGENDA.

La base de datos

La base de datos se llama *mibd mdb* y está realizada en Access. Tiene una única tabla llamada "agenda" con tres campos: *identificador*, de tipo autonumérico; *nombre*, de tipo texto y *telefono*, de tipo texto también. En un primer momento la base de datos tiene algunos datos de ejemplo

Formularios de entrada

Nuestra agenda telefónica va a permitir cuatro operaciones: Insertar nuevos datos (nombre, teléfono), eliminar los datos de una persona dado su nombre, buscar el número de teléfono de una persona por su nombre y ver todo el contenido de la base de datos. Las tres primeras operaciones necesitan datos de entrada que, como hemos visto anteriormente, los podemos hacer llegar a través de la URL o de un formulario. Elegiremos esta segunda opción Con respecto a ver el contenido de la base de datos, no se necesitan datos de entrada; ya que se van a ver todos los datos.

Por tanto creamos tres formularios *DatosNuevos.asp* es un formulario que recoge el nombre y el número de teléfono y se lo envía a un fichero *InsertarDatos Entrada asp*, encargado de realizar la inserción en la base de datos; *Nombre ParaBuscar asp* es otro formulario que recogerá un nombre y se lo hará llegar a otro programa *BuscarTelefono.asp* que será el encargado de buscar el teléfono asociado. *NombreParaBorrar.asp* es el último formulario que recogerá un nombre y se lo mandará al fichero *BorrarDatos Entrada.asp*, y éste eliminará de la base de datos todo rastro de esa persona. Para no resultar monótono no se va a explicar el contenido de dichos ficheros, dada la similitud con el que se puede ver en el listado FormularioDeSaludo al principio de este artículo.

Inserción de datos

Esta operación se realiza en *InsertarDatos Entrada asp* que recibe los datos *Nombre* y *Teléfono* del formulario *DatosNuevos.asp*. En el Listado 4 podemos ver el código que realiza esta operación. Examinemos el fichero con detenimiento:

- En las primeras catorce líneas del código no se hace nada extraño, definir el lenguaje, definir las variables que vamos a usar y empezar a componer una línea de texto que será la consulta SQL que usaremos para insertar los datos. Aquí podemos observar cómo se está haciendo uso de las variables *telefono* y *nombre* a través de *Request Form(...)*.
- En la línea quince nos encontramos con el código:

```
Server.CreateObject
("ADODB.Connection")
```

Con esta línea, que siempre es la misma, le decimos a IIS que nos cree un objeto a través del cual podamos establecer una conexión a una base de datos. Es decir, estamos diciendo a IIS que nos permita usar las funcionalidades especiales

Listado 4. InsertarDatosEntrada

```
<html>
<h1><center>Fichero InsertarDatosEntrada.asp<br></center></h1><br>
<h1><center>- Inserción de datos en la agenda -<br></center></h1>
<%@ LANGUAGE="VBScript"%>
<h3><center>
<%
  Dim Conexion
  Dim ResultadoConsulta
  Dim Consulta, Consulta1, Consulta2, Consulta3
  Dim TelefonoConsultado
  TelefonoConsultado=""
  Consulta1 = "insert into agenda values("
  Consulta2 = ""
  Consulta3 = ",'" & Request.Form("nombre") & "','" &
  Request.Form("telefono") & "')"
  Set Conexion = Server.CreateObject("ADODB.Connection")
  Conexion.Open "DRIVER={Microsoft Access Driver (*.mdb)};
  DBQ=C:\Inetpub\wwwroot\mibd.mdb"
  Set ResultadoConsulta = Conexion.Execute("select max(id)+1 as
  identificador from agenda")
  do while not ResultadoConsulta.EOF
     Consulta2 = ResultadoConsulta("identificador")
     ResultadoConsulta.MoveNext
  Loop
  Consulta = Consulta1 & Consulta2 & Consulta3
  Set ResultadoConsulta = Conexion.Execute(Consulta)
  Conexion.Close
%>
</form>
</center></h3><br><br>
<h3><center><a href="agenda.asp">Inicio</a></center></h3>
</html>
```

Imagen tras la inserción de un nuevo dato.

que tiene para acceso a datos. Este código devuelve una conexión (cerrada, en principio) a una base de datos que en este caso la asignamos a la variable *Conexion*. Pero una conexión es un objeto, no una variable simple como un número o una letra, y en VBScript para asignar un objeto a una variable hay que hacerlo anteponiendo la palabra reservada *Set* a la asignación. Así pues, la línea quince lo que hace es solicitar una conexión a una base de datos a IIS y cuando éste la devuelve, se la asigna a la variable *Conexion*.

La siguiente línea usa *Conexión Open* o sea, el método *Open* del objeto que acabamos de obtener, para abrir la conexión que hasta ahora estaba cerrada. Para que este proceso se haga correctamente hay que decir qué tipo de base de datos vamos a abrir (qué driver se hará cargo de la comunicación) y dónde exactamente está el fichero que contiene la base de datos (propiedad DBQ). En este caso, la línea:

```
Conexion.Open "DRIVER={Microsoft
Access Driver (*.mdb)};
DBQ=C:\Inetpub\wwwroot\mibd.mdb"
```

implementa que la conexión se haga a la base de datos *midb.mdb* que está en el directorio raíz de nuestro servidor web y que utilice el controlador para bases de datos Access.

Ya tenemos establecida la comunicación con la base de datos. Ahora hay que obtener resultados de la misma. El método *Execute* del objeto *Conexion* nos permite realizar una consulta en lenguaje SQL. Se invoca como *Conexion.Execute(ConsultaSQL)* y devuelve un objeto (se ha de asignar con *Set*, por tanto) que almacena una tabla con los resultados obtenidos. Al principio nos encontramos en el primer registro de esa tabla. En el ejemplo, *ResultadoConsulta* es la variable que almacenará el objeto resultante de ejecutar la consulta. Cada registro de la tabla contendrá los campos de la base de datos que hayamos consultado. Si tiene todos (*identificador, nombre, telefono*) se podrá consultar cada uno de ellos mediante *ResultadoConsulta (nombre_del_campo)* que devolverá el valor de ese campo para ese registro. Podemos pasar al siguiente registro mediante *ResultadoConsulta.MoveNext* y *ResultadoConsulta.EOF* nos indicará si se ha llegado al final de la tabla de resultados. De este modo podremos hacer bucles que recorran las tablas de resultados de principio a fin.

En el caso de la inserción de datos en la agenda, el programa, como se puede observar, hace lo siguiente: crea una conexión, abre la conexión a la base de datos Access, realiza una consulta para ver el mayor identificador existente en la base de datos, recorre los resultados, crea un identificador nuevo para el dato que vamos a insertar (los identificadores no se pueden repetir en la base de datos) y hace otra consulta para insertar los datos.

Podemos buscar más de una coincidencia.

Búsqueda de teléfono

Esta operación se lleva a cabo en *BuscarTelefono.asp* que recibe el dato "nombre" del formulario de entrada *NombreParaBuscar.asp*. En el Listado 5 vemos el código para esta operación.

Es más sencillo aún si cabe que el caso de la inserción de datos nuevos. En principio, se construye una consulta SQL. Para ello usamos el valor del parámetro "nombre" Lo obtenemos mediante *Request.Form(..)*. Posteriormente se crea

Listado 5. BuscarTelefono

```
<%@ LANGUAGE="VBScript"%>
<html>
<h1><center>Fichero BuscarTelefono.asp<br></center></h1><br>
<h1><center>- Búsqueda de teléfono -<br></center></h1>
<h3><center>
<%
Dim Conexion
Dim ResultadoConsulta
Dim Consulta
Dim TelefonoConsultado
TelefonoConsultado=""
Consulta = "select telefono from agenda where nombre='" &
Request.Form("nombre") & "'"
Set Conexion = Server.CreateObject("ADODB.Connection")
Conexion.Open "DRIVER={Microsoft Access Driver (*.mdb)};
DBQ=C:\Inetpub\wwwroot\mibd.mdb"
Set ResultadoConsulta = Conexion.Execute(Consulta)
do while not ResultadoConsulta.EOF
TelefonoConsultado = ResultadoConsulta("telefono")
Response.Write(Request.Form("nombre") & ": " & TelefonoConsultado & "<br>");
ResultadoConsulta.MoveNext
loop
if TelefonoConsultado="" then
Response.Write("No hay resultados")
end if
ResultadoConsulta.Close
Conexion.Close
%>
</form>
</center></h3><br><br>
<h3><center><a href="agenda.asp">Inicio</a></center></h3>
</html>
```

una conexión a la base de datos y se abre dicha conexión para poder realizar consultas. Se realiza la consulta SQL que hemos formado unas líneas más arriba en el código, mediante *Conexión.Execute (ConsultaSQL)*, y el resultado de la misma quedará en *ResultadoConsulta*. Es muy importante no olvidar que en VBScript los objetos se asignan con la palabra *Set*.

Se recorre en un bucle la tabla de resultados obtenida, pues puede haber más de un teléfono por persona y por tanto más de un registro por persona en la base de datos. En dicho bucle se pregunta constantemente por el valor del campo "telefono" y se muestra el resultado. Al finalizar, se cierran tanto el resultado como la conexión con *ResultadoConsulta.Close* y *Conexion.Close*, respectivamente.

Se puede mostrar el contenido completo de la agenda.

Página principal de la aplicación web.

Borrado de datos

El fichero *BorrarDatosEntrada.asp* se encarga de esta operación. Recibe un nombre de persona del formulario de entrada *NombreParaBorrar.asp*. El código de este fichero se encuentra en el Listado 6 que como se ve, es muy sencillo.

Primero crea una consulta SQL que, usando el parámetro de entrada "nombre", al ejecutarse hará que se borre de la base de datos las entradas cuyo nombre sea coincidente. Posteriormente se crea una conexión a la base de datos, se abre dicha conexión, se ejecuta la consulta y se cierra. En este caso no hace falta recorrer los resultados porque, de hecho, no hay resultados. No se pregunta por unos datos, se dice que un dato se borre. Se borra y ya está.

Listado de todos los datos

Se puede consultar el contenido del fichero *ListarDatos.asp* en el CD-ROM que acompaña la revista. Este trozo de la aplicación es el encargado de mostrar todo el contenido de la base de datos de la agenda. No necesita de ningun parámetro de entrada puesto que no necesita hacer ninguna consulta condicional, va a mostrar todos los datos.

No vamos a ver su funcionamiento porque es prácticamente idéntico a los ya vistos. De hecho los programas, tanto aplicaciones web como aplicaciones tradicionales, que realizan accesos a bases de datos, tienen un aspecto común: crear una conexión, abrir la conexión a una base de datos concreta, ejecutar una consulta, recorrer los datos, cerrar la consulta y cerrar la conexión. *ListarDatos.asp* sigue al completo estos pasos y en ese orden.

Página principal

Para que el acceso a las distintas opciones resulte mas sencillo y cómodo, creamos una página cuyo único cometido sea enlazar todo. La página *agenda.asp* se encarga de ello. Su contenido es HTML. No incluye código ASP aunque tenga extensión *.asp*. El intérprete de ASP comprobará que no hay marcas de inicio y fin de código interpretable, y por tanto traspasará todo el contenido al navegador del usuario directamente.

Para iniciar la aplicación escribiremos en el navegador *http://localhost/agenda.asp* y podremos probar todas las funciones. Ya tenemos nuestra primera aplicación web completa, con las tres capas mínimas que explicamos al comienzo de este curso.

CONCLUSIONES

Los ejemplos de esta práctica han sido muy sencillos, enfocados principalmente a aprender. Se han separado todas las operaciones por ficheros, para hacerlas más legibles, incluso la entrada de datos se hace en ficheros aparte. Sin muchos cambios y con un poco de paciencia y habilidad se puede desarrollar aplicaciones web mucho más complejas y compactas, puesto que tenemos los ingredientes básicos: métodos para hacer llegar datos a la aplicación, para recibirlos, formas de unir las partes del programa, métodos de almacenaje permanente de datos para su tratamiento posterior, caracterización de usuarios... Con todo esto se puede programar aplicaciones de cualquier tipo: facturación, contabilidad, ERP, CMR ... o montar una intranet corporativa.

Listado 6. BorrarDatosEntrada

```
<%@ LANGUAGE="VBScript"%>
<html>
<h1><center>Fichero BorrarDatosEntrada.asp<br></center></h1><br>
<h1><center>- Elimina datos de la base de datos -<br></center></h1>
<h3><center>
<%
  Dim Conexion
  Dim ResultadoConsulta
  Dim Consulta
  Dim TelefonoConsultado
  TelefonoConsultado=""
  Consulta = "delete from agenda where nombre='" & Request.Form("nombre") & "'"
  Set Conexion = Server.CreateObject("ADODB.Connection")
  Conexion.Open "DRIVER={Microsoft Access Driver (*.mdb)};
DBQ=C:\Inetpub\wwwroot\mibd.mdb"
  Set ResultadoConsulta = Conexion.Execute(Consulta)
  Conexion.Close
%>
</form>
</center></h3><br><br>
<h3><center><a href="agenda.asp">Inicio</a></center></h3>
</html>
```

Este mes con Guitarrista

¡DOS REVISTAS AL PRECIO DE UNA!

GRABACIÓN
HAZLO TU MISMO

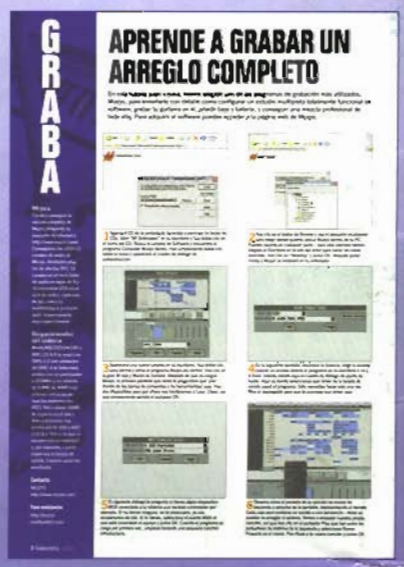
GUÍA ESENCIAL PARA GUITARRISTAS

TODO LO QUE NECESITAS PARA GRABAR TU GUITARRA EN TU ORDENADOR
GRABA UN ARREGLO COMPLETO
PRUEBAS DE EQUIPOS DE GRABACIÓN DIGITAL

MULTIPISTAS DIGITALES PORTÁTILES

Fostex VF80

Yamaha AW166



APRENDE A GRABAR UN ARREGLO COMPLETO

ESPECIAL GRABACIÓN

PREGUNTAS FRECUENTES

GRABACIÓN: HAZLO TU MISMO

Lo que necesitas para grabar música con tu ordenador

Hardware

Este mes comprando Guitarrista te regalamos otra revista:
¡una práctica guía para grabar tu música en casa!

¡Corre a tu quiosco!
o llama al 916 637 349

www.rdmeditorial.com