Año 2 • Número 14 • 6,50 euros

Todo Programación

La Revista bimestral para entusiastas de la programación

www.studiopress.es

NUEVO Visual Studio .NET 2005

Depurar errores de gestión de memoria en Linux con Valgrind

OpenGL: diseñando nuestro primer programa paso a paso

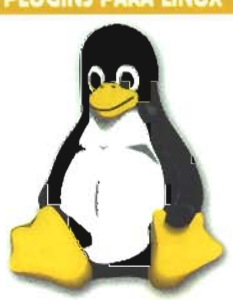
Programación práctica en Qt: un mezclador de sonidos

Acceso a bases de datos en PHP mediante ADOdb

CD-ROM: ECLIPSE 3.1.1 + PLUGINS PARA LINUX Y WINDOWS, STRUTS 1.2.7, NEMERLE 0.9.0, VALGRIND 3.0.1...

Zona Linux

- Nemerle: Nuevo curso "desde cero"
- Mono: Así se programará con C# 3.0



Zona Java

- Introducción al framework Struts
- Creación de ayudas Java online con JavaHelp

DOSSIER: ECLIPSE 3.1 Y SUS HERRAMIENTAS PARA C/C++

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 (jmavarro@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)
Fernando Marín (fmarin@iberprensa.com)
Fabian Seoane (fseoane@iberprensa.com)

DISEÑO PORTADA Y 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.studiopress.es

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 14 • Año 2
Copyright 1/01/06
PRINTED IN SPAIN

EDITORIAL

Eduardo Toribio

Actualidad manda

Muchas novedades se han producido estos últimos días en nuestro pequeño gran mundo del desarrollo, y tratamos en este número de dar cobertura a todas ellas. Por ejemplo, Visual Studio 2005, donde de la mano de Guillermo Som nos adentramos en las nuevas características del entorno de Microsoft. También en la sección Mono dedicamos un completo reportaje a analizar técnicamente algunas de las nuevas caracteristicas del lenguaje C# en su versión 3.0. Pero no acaba aquí la parte que dedicamos en este número de **Todo Programación** a la actualidad, ya que además tenemos nueva versión de Eclipse, la 3.1, con una sustancial mejora, sobre todo a nivel de funcionalidad, y algunas novedades en lo que se refiere a capacidades Veremos también cómo marcha el plugin para trabajar en C/C++.
En cuanto a la parte más práctica de la revista, hemos elegido una serie de tutoriales orientados a programadores en Java y a herramientas más específicas de la plataforma Linux. Tanto Java como Linux tendrán ya desde este número y en lo sucesivo una mayor presencia en la revista. Digamos que, junto a .NET, configuran el esqueleto sobre el cual elaboramos nuestros contenidos. Esperemos que os gusten, de momento, ya somos la revista especializada en desarrollo de mayor difusion en España, es un hecho y no una frase hecha

SUSCRIPCIONES

Como oferta de lanzamiento existe la posibilidad de suscribirse durante 12 números a **Todo Programación** por solo 66,30 euros lo que significa un ahorro del 15% respecto el precio de portada. Además de regalo puedes elegir entre una de las dos guías que aparecen abajo.
Más información en: www.studiopress.es

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 14

En portada

Dossier: El nuevo Visual Studio 2005

10 Desde la aparición de la primera versión de Visual Studio .NET todo lo relacionado con la programación ha cambiado debido al gran impacto que esta plataforma ha tenido en el mercado. Ahora con la nueva versión que se presenta, la 2005, se refuerza y se facilita la creación de aplicaciones basadas en la nueva versión de .NET Framework 2.0. En el dossier que hemos elaborado analizamos las nuevas características que convierten a este entorno en una herramienta muy útil a la hora de desarrollar aplicaciones de escritorio, web o dispositivos móviles.

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: Eclipse 3.1 y sus herramientas para C/C++ >>

16 Eclipse 3.1 supone la culminación de un año de trabajo de la Fundación Eclipse, una de las uniones más prósperas entre software libre, empresa y Java. Para esta última versión no se han introducido cambios radicales, pero sí una serie de pequeñas mejoras que hacen el trabajo con esta herramienta mucho más productivo.

Avance de las nuevas características para C# 3.0 >>

19 Recientemente Microsoft ha publicado un documento donde se describen las principales novedades que incorporará C# en su versión 3.0. Dedicamos este mes nuestra habitual sección de desarrollo Mono a analizar estas nuevas características.

.net WINDOWS

Trucos C++: Lanzando aplicaciones >>

23 Dedicamos un tutorial para ver de manera muy rápida y práctica cómo lanzar y ejecutar aplicaciones desde nuestros programas en C++ y es que son muchas las ocasiones en que necesitamos realizar ciertas operaciones sin delegar en el sistema operativo.

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 el CD de **Todo Programación** puedes encontrar en cada número las herramientas de programación más útiles y que conseguirán simplificar tu trabajo. Además, tendrás a tu disposición todas las aplicaciones y los ejemplos que se requieran para seguir correctamente los artículos y cursos que presentamos en la revista. En este número el protagonista es Eclipse y sus plugins.

TALLER PRÁCTICO

Acceso a bases de datos en PHP mediante ADOdb >>

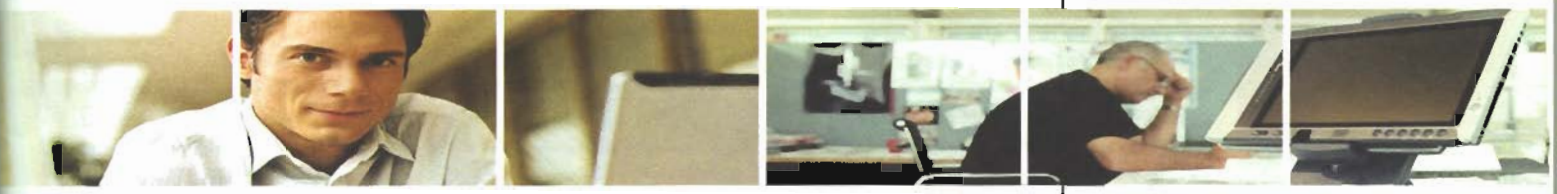
36 ADOdb es un conjunto de clases PHP con soporte para multitud de sistemas gestores de bases de datos, que permite abstraer al programador del acceso a los datos de una manera similar al que se produce con JDBC en Java.

Programación multimedia en Qt : mezclador de sonidos >>

40 Vamos a realizar una práctica para seguir progresando en nuestros conocimientos de las librerías multiplataforma Qt, en esta ocasión vamos a estudiar la programación de un mezclador de sonido, nos apoyaremos también en el uso de OpenGL.

Introducción al framework Struts >>

44 En dos entregas vamos a abordar el que probablemente se ha convertido en el estándar de desarrollo de aplicaciones web sobre Java, Struts. Introducimos esta herramienta a nivel teórico y veremos un primer contacto práctico.



Creación de ayudas Java online con JavaHelp >>

49 Es de vital importancia que seamos capaces de construir un buen sistema de ayuda para el usuario que utilizará la aplicación que desarrollamos. Si lo hacemos en Java, Sun pone a nuestra disposición JavaHelp, descubrámoslo.

OpenGL: nuestro primer programa >>

53 En nuestro anterior número tuvimos la primera toma de contacto con OpenGL. Ahora vamos a realizar nuestro primer programa para ir viendo los mecanismos fundamentales para trabajar con esta librería multiplataforma.

Valgrind, localizando errores de gestión de memoria >>

57 Existen muchas aplicaciones que analizan código de forma estática y depuradores que funcionan a buen nivel, pero no detectan errores causados por el uso incorrecto de la memoria dinámica. Es ahí donde entra en juego el sistema de depuración Valgrind.

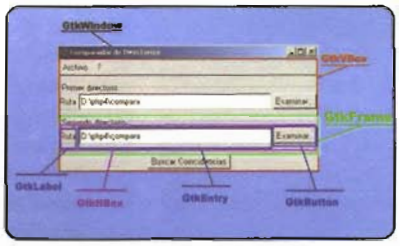
NOTICIAS

6. **Madrid** acoge la sesión inaugural del HP Integrity Summit 2005.
6. **Nueva** edición de BEA World en California.
7. **IBM** y Red Hat se alían para impulsar Linux en los nuevos mercados.
7. **CA** cede patentes a la comunidad open source.
8. **Oracle** Open World.
8. **SGI** incorpora tecnología reconfigurable RASC
8. **HP** firma acuerdos con la ONCE y la SGFD.
8. **Aula** SUN en la UC3M.
9. **Archos** lanza sus nuevos reproductores multimedia portátiles.
9. **Nuevo** portátil Airis PRAGMA 845.
9. **Equipos** multifunción Dell All-in one.

Y ADEMÁS...

DESARROLLO WEB: Diseño con PHP-GTK >>

62 PHP-GTK es una extensión para PHP que nos permite la posibilidad de crear interfaces gráficos de usuario (GUI) para que nuestras aplicaciones interactuén con el usuario a través de su entorno gráfico (Windows, KDE, GNOME, etc.)



CUADERNOS DE PRINCIPIANTES

Introducción a Nemerle >>

28 Nemerle es un lenguaje de alto nivel para la plataforma .NET que vamos a utilizar en este curso para principiantes. El objetivo es aprender a programar para .NET de una manera sencilla, a nivel práctico, paso a paso aprovechando las virtudes de este lenguaje.

Visual Basic para Aplicaciones: macros en Excel >>

33 Excel es una aplicación muy extendida y usada, no hay duda, pero vamos a dar un salto importante en su manejo mediante la introducción de macros que nosotros mismos podemos programar utilizando Visual Basic para Aplicaciones.

NIVEL DE DIFICULTAD:

MATERIAL INCLUIDO EN EL CD-ROM:
Ejemplos ■ ADOdb463.zip ■ scite1.66.tgz

Accesos a bases de datos en PHP mediante ADOdb

MANUEL DOMÍNGUEZ
mdominguez@iberprensa.com

Este mes vamos a estudiar ADOdb, un conjunto de clases PHP para gestionar bases de datos. El acceso a bases de datos desde las aplicaciones, como ya se ha comentado en otros tutoriales de Todo Programación, es esencial.

Desde Java, vía JDBC, se abstrae en gran medida la lógica de acceso a los datos, de la base de datos usada. En PHP no ocurre así, sino que este lenguaje tiene una API distinta para cada base de datos a usar y esto provoca problemas de portabilidad de los datos de un SGBD a otro. ADOdb viene a rellenar este hueco, abstrayendo al programador PHP del acceso a los datos de forma similar a JDBC en Java.

Efectivamente, son muchos los casos en que un proyecto o aplicación con varios años de desarrollo e implantación requieren un cambio en el SGBD que maneja los datos. Por ejemplo, cada vez son más las Comunidades Autónomas que apuestan por el software libre, LinEx, Guadalinex, Molinux... y las Consejerías de dichas regiones van portando poco a poco sus proyectos existentes en, por ejemplo, MS SQL Server, a MySQL o PostgreSQL. Esto que parece trivial, no lo es cuando las aplicaciones están realizadas en lenguajes como PHP, usando la API nativa de este lenguaje. En este caso suele ser casi más barato y menos problemático crear de nuevo las aplicaciones, que la propia adaptación. Con el uso de ADOdb este proceso de migración es mucho más sencillo en estas situaciones tan drásticas, pero tampoco es mala idea usarlo por costumbre en los proyectos para abordar futuras actualizaciones de un mismo SGBD, por ejemplo de Oracle 8i a Oracle 9i o 10g. De forma que, simplificando mucho, solo habría que quitar un SGBD, poner otro y retocar en lugares muy concretos el código de la aplicación.

EL PROYECTO

ADOdb comenzó su andadura en el año 2000, por lo que ya se puede considerar que tiene la estabilidad y la solera suficiente para confiar en él. Además muchos proyectos, algunos relevantes como PostNuke, lo usan, y aunque esto suene a publicidad barata no lo es. Su autor, John Lim, puso el código fuente bajo licencia dual BSD-LGPL, indicando además que en caso de conflicto debe prevalecer la licencia BSD con respecto a la LGPL. Esto se traduce en que se puede usar la librería para proyectos de software libre o propietarios, comerciales o no, con bastante flexibilidad y muy pocas limitaciones.

Se trata de un conjunto de clases PHP que se deben importar en el proyecto desde el que deseemos hacer uso de la API que proporciona. Actualmente tiene soporte para multitud de SGBD, por ejemplo MySQL, Oracle, Microsoft SQL Server, Sybase, Sybase SQL Anywhere, Informix, PostgreSQL, FrontBase, Interbase, Foxpro, Access, ADO, etcétera.

Para cualquier información, documentación adicional, etcétera, la página del proyecto es *http://adodb.sourceforge.net*.

PREPARACIÓN DEL ENTORNO

Si se toma la decisión de usar ADOdb, aunque no es necesario, es muy recomendable instalar un editor de código PHP que al menos coloree la sintaxis. Hay muchos y muy buenos, para Windows y GNU/Linux. En este artículo se utilizará Scite, un editor sencillo que incluimos en el CD-ROM.

Por supuesto, se presupone que se tiene bien instalado y configurado un servidor web como Apache, PHP 4.x o posterior y un SGBD, por ejemplo MySQL, con alguna base de datos creada. De esta forma podremos ejecutar los ejemplos que acompañan al artículo y, en cualquier caso, nos harán falta para desarrollar de forma normal una aplicación usando ADOdb.

En nuestro caso, trabajaremos con una base de datos MySQL llamada *mibd*, con nombre de usuario y contraseña *usuario/todoprogramacion*. El listado DDL para la creación de la tabla (una sola) *clientes*, existente en la base de datos, se puede consultar en el Listado 1. Usaremos PHP 4.3.7, MySQL 4.0.15 y Apache 2.

En cuanto a ADOdb, se presenta como un fichero comprimido que contiene una estructura de carpetas/directorios conteniendo ficheros PHP donde se implementan las clases proporcionadas por este proyecto. El modo de utilizarla es hacer accesible dicha estructura de directorios desde el lugar donde vayamos a crear nuestra aplicación PHP.

Si tenemos todos los componentes instalados, la base de datos creada y todo funcio-

Listado 1. Creación de una tabla

```
# Servidor.....: localhost
# Base de datos: mibd
# Tabla.....: 'clientes'
#
CREATE TABLE `clientes` (
  `ID` int(11) NOT NULL auto_increment,
  `NOMBRE` varchar(100) NOT NULL default '',
  `APELLIDO1` varchar(100) NOT NULL default '',
  `APELLIDO2` varchar(100) NOT NULL default '',
  `DNI` varchar(100) NOT NULL default '',
  PRIMARY KEY  (`ID`)
) TYPE=MyISAM;
INSERT INTO clientes VALUES (1,'Manolo','Domínguez','Dorado','1234567');
INSERT INTO clientes VALUES (2,'José
Luis','Carracedo','Pérez','3216548');
INSERT INTO clientes VALUES (3,'María
José','Lorente','Millán','1597535');
```

na, podemos ponernos en marcha a desgranar las ventajas de ADOdb. ¡Manos a la obra!

LA API DE ADODB

ADOdb está compuesto por dos clases principales: *ADOConnection* y *ADORecordSet*. La primera de las clases contiene métodos para la conexión a las bases de datos de distintos modos, obtención de fechas de la base de datos de una forma unificada, para la concatenación, para establecer inicios y fin de transacciones, etcétera. La segunda almacena en memoria los datos obtenidos de la consulta a una base de datos. Permite recorrer los datos obtenidos, métodos para obtener información sobre los campos y sus tipos y métodos para presentar al usuario los datos de distintas maneras. En las siguientes líneas iremos viendo los métodos más importantes de estas dos clases.

Conexión a una base de datos

Lo primero que hay que hacer en todos los lugares donde se desee usar ADOdb es importar el fichero *adodb.inc.php*, que está en el raíz del directorio que se crea al descomprimir el proyecto. La siguiente línea sería un ejemplo:

```
include('adodb/adodb.inc.php');
```

Ahora podemos hacer uso de la API de ADOdb. El siguiente paso suele ser la creación de una instancia de la clase *ADOConnection*; esto se realiza con la función *ADONewConnection(String TipoSGBD)*, donde el parámetro indica el tipo de SGBD al que nos vamos a conectar. Viene especificado en la web del proyecto y es una lista bastante larga; por ejemplo *'mysql'* para MySQL, *'access'* para Microsoft Access u *'oci8'* para Oracle 8/9, aunque hay muchos más, por supuesto:

```
$bd = ADONewConnection('mysql');
```

En este punto tendremos un objeto *bd*, de tipo *ADOConnection*, pero no estaremos conectados aún a la base de datos que deseamos. Antes de nada, es importante aclarar que aunque en los ejemplos de este artículo se especificará en el propio código el tipo de SGBD, es mejor idea especificar todos estos parámetros en un fichero de configuración, por ejemplo *configuracion.php* (creado por nosotros) e importarlo en cada lugar que se necesite. Así para el cambio a un nuevo SGBD se necesitarán retocar menos ficheros.

Ahora sí, para realizar la conexión a la base de datos podemos seleccionar varios métodos. Los más usados son *PConnect(servidor, usuario, clave, base_datos)* y *Connect*, con los mismos parámetros. En el primero de los casos se crea una conexión permanente a la base de datos, lo cual permite que posteriormente sea más rápida la ejecución de todas las operaciones, pero a costa de estar ocupando recursos en el servidor de bases de datos. *Connect*, normalmente más usado, crea una conexión no permanente que se restablece con cada operación a realizar sobre dicha conexión. Para nuestro ejemplo, sería algo como lo que muestra la siguiente línea:

```
$bd->Connect('localhost', 'usuario',
'todoprogramacion', 'mibd');
```

Y ya estaríamos conectados a la base de datos, ¿sencillo no? Además vemos como ya se pueden apreciar las ventajas de ADOdb, puesto que en PHP las funciones para conectar con la base de datos difieren de un SGBD a otro.

Desconectando de la base de datos

Todo lo que comienza, finaliza. Así pues, cuando ya no vayamos a usar más la conexión con la base de datos, debemos cerrarla. Con las últimas versiones de PHP realmente no es necesario realizar esta operación, pero tampoco pasa nada por hacerlo y se puede englobar dentro del *'manual de buenas costumbres'* de un programador. Para ello tenemos el método *Close()* de *ADOConnection*.

En el Listado 2 podemos observar el contenido de *ejemplo1.php*, que realiza una conexión a nuestra base de datos y desconecta de la misma.

Select, insert, update y delete

Existen varios métodos para estas sentencias de manipulación de datos. En otras API's de otros proyectos suele haber un método para *select* y otro para *insert, update* y *delete*. En ADOdb no es así, y los métodos existentes para lanzar SQL contra una base de datos valen para las cuatro sentencias anteriores. El método más usado es *Execute(SQL, ArrayEntrada)*, de la clase *ADOConnection*. El primero de los parámetros es una cadena de caracteres que será la consulta SQL que se desea lanzar. El segundo parámetro es opcional y permite asociar una variable de nuestro código PHP a un alias que insertemos dentro de la sentencia SQL.

La forma normal de usar este método, que requiere la conexión previa con la base de datos es, por ejemplo, como se muestra en las siguientes líneas:

```
$rs = $bd->Execute('select nombre
from clientes');
```

o:

```
$rs = $bd->Execute('update clientes
set nombre='Pepe');
```

En todos los casos, *Execute(...)* devuelve un *ADORecordSet* si el SQL que se ha lanzado contra la base de datos ha tenido éxito, mientras que si ha ocurrido cualquier error, se devolverá *FALSE*. Esto nos permitirá controlar que lo que vamos haciendo está saliendo como se esperaba. Si el SQL lanzado es una operación *insert*, *update* o *delete*, *ADOConnect* nos proporciona un método llamado *Affected_Rows()* que nos indicará el número de filas que han sido afectadas por el SQL que hemos especificado. Por su parte, y sin adelantar demasiado porque se verá en las siguientes líneas, el *ADOResultSet* devuelto nos proporciona un método para contar el número de registros que se han recogido tras una sentencia SQL de tipo *select*. Para comprobar el funcionamiento de lo visto hasta ahora, el fichero *ejemplo2.php* (Listado 3) recoge un ejemplo en el que se usan los cuatro tipos de sentencia SQL y se inspecciona el número de registros que entran en juego en cada caso.

Listado 2. Conexión y desconexión a la BB.DD.

```
<?php
  include('../adodb/adodb.inc.php');
  $bd = ADONewConnection('mysql');
  if (!$bd->Connect('localhost', 'usuario', 'todoprogramacion', 'mibd')) {
    print("<center><h1>No se pudo conectar</h1></center>");
  } else {
    print("<center><h1>Se pudo conectar</h1></center>");
    $bd->Close();
    print("<center><h1>Se ha cerrado la conexión.</h1></center>");
  }
?>
```

RESULTADO DE UNA CONSULTA

Ya adelantábamos en los párrafos anteriores que otra de las clases principales de ADOdb es *ADORecordSet*. Como su nombre indica (a todos aquellos familiarizados con las conexiones a bases de datos), este objeto permite almacenar registros obtenidos de una base de datos mediante una sentencia SQL, de tal forma que se pueden recorrer cada uno de ellos y tratarlos de forma individual.

Obtención del número de registros

Lo más normal que alguien puede querer hacer tras obtener un conjunto de registros es saber cuántos ha recogido. Como comentábamos líneas atrás, esta clase nos ofrece un método para ello, *RecordCount()* Siguiendo el ejemplo que vamos viendo durante todo el artículo, nosotros deberíamos hacer algo como:

```
$NumReg = rs->RecordCount();
```

para poder obtener en la variable *NumReg* el total de registros que hemos conseguido mediante la ejecución de una sentencia SQL.

Moviéndose por los resultados

Podemos entender un *ADORecordSet* como una tabla donde cada fila es un registro de la base de datos obtenido mediante una *select* SQL y donde cada columna coincide con un campo de dicho registro en la base de datos. Existe un cursor o puntero interno que en principio está apuntando al primero de los registros, y que podemos mover y desplazar adelante o atrás a placer para ir obteniendo los datos requeridos del *ADORecordSet*

El primer método para esta acción es *MoveFirst()*, que posiciona el cursor de ADORecordSet en el primero de los registros contenidos. *MoveLast()* mueve el cursor al último registro existente en *ADORecordSet* y por último, el método más usual de todos, *MoveNext()*, que avanza el puntero hasta el siguiente registro (a partir del actual) y es uno de los método que se suele utilizar más para realizar bucles que recorran *ADORecordSet*. En relación a esto, *ADORecordSet* proporciona un atributo público, *EOF*, que es verdadero si se ha llegado al final, es decir, si ya no quedan más registros que visitar porque todos se han recorrido. Es esencial para crear bucles con el método *Move()*. Se utilizaría de la siguiente forma.

```
if ($rs->EOF) {
    ...
}
```

Para mayor claridad, en el ejemplo que se puede ver en el Listado 4 de la página siguiente se hace el recorrido de un *RecordSet*. No se muestran los valores aun, solo se recorre el resultado de una consulta.

Accediendo a los resultados

Generalmente cuando se recorren los registros de un *ADORecordSet* se espera acceder a ellos en algún momento. Existen varias formas de hacerlo. Si utilizamos la familia *Move'''()* para movernos por los registros, podemos acceder al atributo público de la clase *'fields'*, que es el array de campos que conforman el registro completo que actualmente está apuntado por el cursor. Además, mediante el método *FieldCount()* de *ADORecordSet* podemos saber qué número de campos contiene cada registro. De este modo para cada registro recorrido con *Move'''()* es posible obtener cada uno de los campos

Sin embargo, existe un método más sencillo, que permite recorrer el conjunto de registros a la vez que accede a cada uno de ellos. Son los métodos *Fetch'''()*. Por ejemplo, *FetchRow()*, tendría el mismo resultado que acceder a *'fields'* y posteriormente hacer un *MoveNext()*. O el más interesante todavía, *FetchNextObject(boolean Mayusculas)*, que realiza la misma operación pero devuelve el registro como un objeto, de forma que a cada campo se puede acceder mediante el operador ->, por ejemplo *$rs->NOMBRE*, *$rs->APELLIDO1*, etcétera. Es mucho más cómodo. En este último metodo, el parámetro *Mayúsculas* es lógico. Puesto a *TRUE* indica que los nombres de los campos, para luego ser llamados con el operador ->, se pasarán a mayúsculas todos. Si está puesto a *FALSE*, estará como lo devuelva el SGBD al que nos estemos conectando. Por comodidad y por hacer las cosas siempre de un modo correcto, es mas lógico hacerlo con *Mayúsculas=TRUE*, aunque no hace falta especificarlo porque es el valor por defecto.

El Listado 5, correspondiente al ejemplo 4, muestra cómo se recorrería un

Listado 3. Inspección de registros

```
<?php
  include('../adodb/adodb.inc.php');
  $bd = ADONewConnection('mysql');
  $inserciones = 0;
  !$bd->Connect('localhost', 'usuario', 'todoprogramacion', 'mibd');
  $rs = $bd->Execute("insert into clientes values
  (NULL,'José','Ruiz','Lasso','0000000')");
  if ($rs) {
    print ('<h3>Se han insertado '.$bd->Affected_Rows().' registros</h3>');
  } else {
    print ('<h3>No se han insertado registros</h3>');
  }
  $rs = $bd->Execute("update clientes set nombre='Xavier' where nombre='José'");
  if ($rs) {
    print ('<h3>Se han modificado '.$bd->Affected_Rows().' registros</h3>');
  } else {
    print ('<h3>No se han modificado registros</h3>');
  }
  $rs = $bd->Execute("delete from clientes where nombre='Xavier'");
  if ($rs) {
    print ('<h3>Se han eliminado '.$bd->Affected_Rows().' registros</h3>');
  } else {
    print ('<h3>No se han eliminado registros</h3>');
  }
  $rs = $bd->Execute('select * from clientes');
  if ($rs) {
    print ('<h3>Existen '.$rs->RecordCount().' registros en mibd.</h3>');
  } else {
    print ('<h3>Ocurrió un error al ver los registros que hay</h3>');
  }
  $rs->Close();
  $bd->Close();
?>
```

ADORecordSet con *FetchNextObject()*, es igual con *FetchRow()* salvo que se debe acceder a los campos como un array y no como un objeto, con el operador ->.

Cierre de un ADORecordSet

Los resultados obtenidos tras una consulta son almacenados en un *ADORecordSet* en memoria dentro del servidor que ejecuta la aplicación. Esto significa que esos datos ocupan memoria. Y esta cantidad de memoria puede ser importante según el número de registros y el tipo de los mismos, por lo que cuando ya se han terminado de usar los resultados, lo lógico y lo que se debe hacer es liberarla. Para ello, *ADORecordSet* nos ofrece el método *Close()*, cuya finalidad es precisamente esa, liberar la memoria, haciendo inaccesibles desde ese momento los resultados.

TRANSACCIONES

Todos hemos oído hablar de las transacciones. Ese conjunto de operaciones que se consideran atómicas; o bien se realizan todas o no se realiza ninguna. Unos SGBD las soportan y otros no. ADOdb nos permite usar una API común siempre, y bastante sencilla, para el manejo de transacciones. Para ello contamos con varios métodos de *ADOConnection*: *StartTrans()*, *CompleteTrans()*, *FailTrans()* y *HasFailedTrans()*

Inicio y fin de una transacción

Para indicar a ADOdb que se van a realizar una serie de operaciones atómica, hay que invocar al método *StartTrans()*. Para finalizar la transacción, o sea, para indicar a ADOdb dónde termina el conjunto de operaciones atómicas, se usa *CompleteTrans()*. Entre ambas se pueden realizar diversas operaciones de tipo *Execute(,)*, todas ellas formando una sola transacción.

CompleteTrans() hace algo más que determinar el final de la transacción, realiza un commit de la misma si no ha habido ningún error desde que se llamó a *StartTrans()* y realiza un rollback de todas las operaciones en caso contrario. Es muy cómodo usar este modo de trabajo, sobre todo en grandes proyectos, porque evita al desarrollador tener que controlar todos los posibles casos de error. Controlar todos los casos de error provoca un código farragoso y de difícil mantenimiento que de esta forma se evita. Un ejemplo de transacción sería el siguiente:

```
$bd->BeginTrans();
$bd->Execute("update padres set
CODIGO=123 where CODIGO=321");
$bd->Execute("update hijos set
C_PADRE=123 where C_PADRE=321");
$bd->CompleteTrans();
```

Si cualquiera de los *Execute(...)* diese problemas, *CompleteTrans()* desharía los dos *updates*, si no, éstos se asentarían en la base de datos.

Errores en la transacción

Lo anterior está muy bien, pero ¿qué pasa si el SGBD no da un error pero nosotros sabemos que sí se ha producido un fallo, por ejemplo, de tipo lógico? Pues que contamos con el método *FailTrans()* que provoca un error de transacción. Insertado dentro de la transacción produce que cuando se llame a *CompleteTrans()*, este método haga rollback. De este modo habremos forzado un rollback. Por otro lado, cuando termina la transacción... ¿cómo sabemos si acabó correctamente o no? Pues para ello contamos con el método *HasFailedTrans()*, que llamado tras *CompleteTrans()* devuelve *TRUE* si se hizo commit de la transacción o *FALSE* si se hizo rollback. Siempre debe haber un *CompleteTrans()* por cada *StartTrans()*, aunque se pueden anidar. Por último, hay que saber que el SGBD al que nos conectamos no soporta transacciones, *CompleteTrans()* siempre hará commit (lógico).

Listado 4. Recorrido de un *RecordSet*

```
<?php
    include('../adodb/adodb.inc.php');
    $bd = ADONewConnection('mysql');
    $inserciones = 0;
    !$bd->Connect('localhost', 'usuario', 'todoprogramacion', 'mibd');
    $rs = $bd->Execute("select * from clientes");
    print('<H2>Se han extraido '.$rs->RecordCount().' registros.</H2>');
    $contador = 0;
    while (!$rs->EOF) {
      print('<H3>Puntero en el registro '.$contador.'</H3>');
      $contador++;
      $rs->MoveNext();
    }
    $rs->MoveFirst();
    print('<H3>Puntero en el primer registro</H3>');
    $rs->MoveLast();
    print('<H3>Puntero en el último registro</H3>');
    $rs->Close();
    $bd->Close();
?>
```

Listado 5. Ejemplo de recorrido

```
<?php
    include('../adodb/adodb.inc.php');
    $bd = ADONewConnection('mysql');
    $inserciones = 0;
    !$bd->Connect('localhost', 'usuario', 'todoprogramacion', 'mibd');
    $rs = $bd->Execute("select * from clientes");
    print('<H1>Se han extraido '.$rs->RecordCount().' registros.</H1>');
    $contador = 0;
    while ($o = $rs->FetchNextObject()) {
      print('<H2>Registro '.$contador.'</H2>');
      print('<H3>---> Nombre: '.$o->NOMBRE.'</H3>');
      print('<H3>---> Apellido 1: '.$o->APELLIDO1.'</H3>');
      print('<H3>---> Apellido 2: '.$o->APELLIDO2.'</H3>');
      print('<H3>---> DNI: '.$o->DNI.'</H3>');
      $contador++;
    }
    $rs->Close();
    $bd->Close();
?>
```

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

USUARIO LINUX

- *Usuario Linux es una obra compuesta por doce entregas de periodicidad semanal, cuyo objetivo es formar al lector en el manejo del sistema Linux y sus principales aplicaciones*
- *Aprende a trabajar de manera práctica, con tutoriales guiados y explicaciones sencillas, con el sistema operativo de mayor futuro, la plataforma que ya utilizan las empresas*
- *Linux es el sistema más moderno y eficiente para PC, con él olvídate de cuelgues, virus, spyware y demás problemas*
- *Con el número 1 se incluye un DVD con Debian Linux Sarge, un sistema Linux completo y en español que incluye más de 7000 aplicaciones.*

NVU	OpenOffice.org	Thunderbird	Evolution	Inkscape	Firefox	Mono	Gimp
Diseño Web	Suite de Ofimática	Gestión de Correo	Suite de Comunicaciones	Autoedición de Textos	Navegador Web	Desarrollo .NET	Tratamiento y y Retoque Digital

Composición de la obra

12 coleccionables divididos en:

Teoría
Conoce el sistema Linux a nivel profesional, cómo se instala en un PC, cómo se administra y cómo se trabaja con los principales entornos gráficos.

Software
Descubre todas las aplicaciones libres que hay para Linux, desde la suite de ofimática OpenOffice.org a las herramientas de desarrollo compatibles con .NET.

Tutorial Práctico
Aprende lo que realmente se necesita hoy en día: Cómo montar un servidor web, cómo configurar una red local, cómo programar una web dinámica o cómo instalar un cortafuegos profesional.

Oferta de suscripción

Usuario Linux está disponible en tu quiosco, pero también puedes suscribirte directamente llamando al 91 628 02 03 y recibir la obra en tres entregas mensuales por solo 66,90 euros.

Además conseguirás de regalo la tapa para encuadernar la obra y una minisuscripción de tres números a la revista Todo Linux.

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

ya a la venta en quioscos y centros comerciales