Año 1 • Número 7 • **5,98** euros

TODO Programación

La Revista mensual para entusiastas de la programación

www.iberprensa.com

Acceso a Bases de Datos: JDBC Vs. ODBC

El entorno de desarrollo integrado Eclipse 3.0

Programación en C del microcontrolador PIC16F84

Sistemas gestores de Bases de Datos

8 413042 751977 00007

Studio PRESS

■ EN EL CD-ROM: POSTGRESQL, MYSQL, DBXML, BORLAND TOGETHER, ETC.

Zona Linux

- **Mono 1.0:** Funciones de librerías externas
- **C#:** Ventajas de la serialización de objetos

Zona Windows

- **POO en .NET:** Ejemplos prácticos en C#
- **Breeze y Flex:** Reviews de las últimas herramientas de Macromedia

PROGRAMACIÓN DE SUBPROGRAMAS EN ORACLE PL/SQL

TODO Programación

La Revista mensual para entusiastas de la programación

DIRECTOR
Eduardo Toribio
etoribio@iberprensa.com

REDACCIÓN
Fernando Escudero
fescudero@iberprensa.com

COLABORADORES
Antonio M. Zugaldía
(azugaldia@iberprensa.com)
David Santo Orcero
(dorcero@iberprensa.com)
Manuel Domínguez
(mdominguez@iberprensa.com)
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)
Lorenzo Gil
(lgil@iberprensa.com)
José Rivera
(jrivera@iberprensa.com)
Jaime Anguiano
(janguiano@iberprensa.com)
Alejandro Serrano
(aserrano@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 07 • Año 1

EDITORIAL

Eduardo Toribio

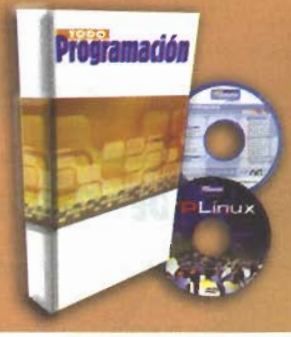
BB.DD

La importancia que poseen los sistemas gestores de bases de datos en el mundo empresarial actual es enorme. Ya no basta con aplicaciones robustas que aseguren la integridad de los datos, ahora se requiere almacenar los datos pero sobre todo la posibilidad posterior de gestionarlos de manera inteligente, obteniendo de ellos información significativa para la actividad comercial. Nadie duda, por ejemplo, que la empresa que pueda saber más sobre sus posibles clientes está en franca ventaja respecto sus competidores.

Hoy, el núcleo de muchas compañías son sus bases de datos y es por ello que nosotros hemos decidido dedicar en este número el cover a realizar un recorrido por las aplicaciones más importantes comerciales y libres que encontramos en el mercado. Además en el CD-ROM que acompaña la revista incluimos muchas de ellas, las que son libremente distribuibles, tanto para plataforma Windows como Linux. No quiero despedirme sin resaltar la importancia que viene adquiriendo en los últimos meses un proyecto al que también reservamos espacio este mes, me refiero a Eclipse, ya en su versión 3.0 y con el respaldo de compañías tan importantes como Novell, IBM, HP, etc.

SUSCRIPCIONES

Como oferta de lanzamiento existe la posibilidad de suscribirse durante un año (12 números) a **Todo Programación** por solo 61 euros lo que significa un ahorro del 15% respecto el precio de portada. Además de regalo se incluye un archivador para coleccionar y guardar las revistas con sus CD-ROMs.
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

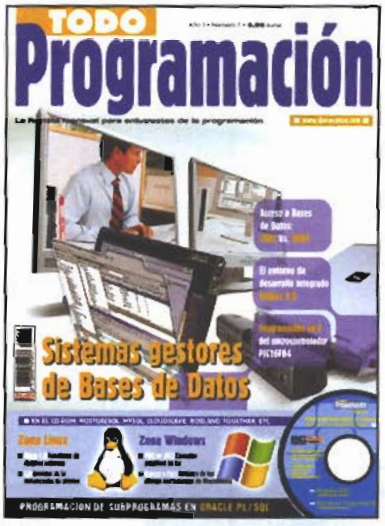
Comparte con nosotros tu opinión sobre la revista, envíanos tus comentarios, sugerencias, ideas o críticas.

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 7

En portada

Bases de datos

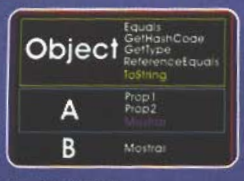
12 Dedicamos nuestro tema de portada a realizar un recorrido por los principales sistemas gestores de bases de datos que implementan la mayoría de las empresas. Conoceremos los actores más destacados de este segmento, tanto en el lado comercial como en el boyante libre, donde MySQL gana cuota de mercado de manera exponencial. Analizaremos ventajas, desventajas y nivel de implantación para poder extraer una idea muy aproximada de cuál es la realidad de este mercado.

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

Programación orientada a objetos en .NET >>



25 En nuestro número anterior vimos conceptos teóricos de la POO respecto los lenguajes de .Net Framework. Ahora veremos algunas características mediante ejemplos prácticos, utilizando para ello Visual Basic .NET.

ANÁLISIS DE SOFTWARE
Breeze: Presentaciones y proyectos >>

31 Breeze es una herramienta basada en Flash y Powerpoint que, gracias a sus variadas posibilidades de configuración y funcionamiento ofrece comunicación web, presentaciones en tiempo real, colaboración y training.

ANÁLISIS DE SOFTWARE
Programación Flash con Flex >>

34 Macromedia Flex es un servidor de presentaciones basado en XML y Java que puede desplegarse en cualquier servidor de aplicaciones bajo Windows o Linux. Actualmente se está desarrollando una versión para .NET.

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 Este mes nuestro CD está dedicado a las bases de datos open source, incluimos entre otras MaxDB, FireBird, dbXML, MySQL, etc. Además y disponible para ambas plataformas (Windows/Linux) el entorno de desarrollo Eclipse en su nueva versión 3.0 y el entorno de modelado Borland Together Designer Community Edition. Además como todos los meses una recopilación de utilidades para el programador.

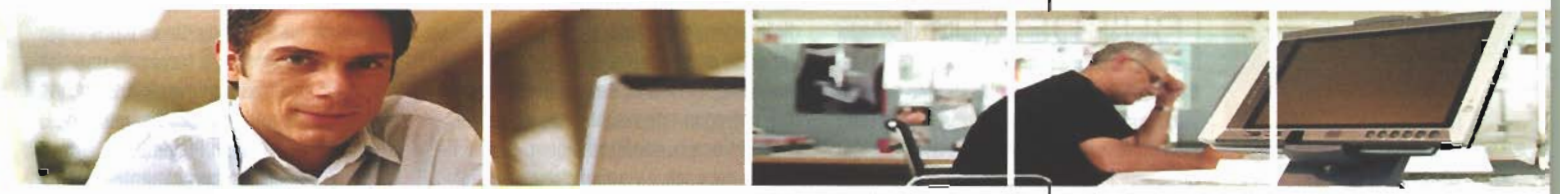
TALLER PRÁCTICO

Acceso a BB.DD. desde aplicaciones Java >>

37 Segunda y última entrega de esta miniserie dedicada a JDBC. En esta ocasión vamos a repasar lo aprendido hasta ahora con ejemplo práctico. Veremos las ventajas respecto a conectores ODBC.

El entorno de desarrollo integrado Eclipse 3.0 >>

41 Eclipse es un entorno de desarrollo multipropósito y multiplataforma fácilmente extensible mediante plugins que está desarrollado conjuntamente por diversas empresas de la talla de IBM, HP, Novell, Borland, etc.



LIBROS

10 Os presentamos las ultimas novedades editoriales del mercado sobre programación. Publicaciones especializadas en el sector para todos los niveles, desde los que están dando sus primeros pasos en este mundo hasta los programadores avanzados.

ZONA LINUX

Actualidad Desarrollo >>

18 Dedicamos nuestra sección de programación Linux a la serialización de objetos mediante algunos espacios de nombres que nos proporciona la librería de clases de .NET. Todo con ejemplos prácticos.

Mono: C, funciones de librerías externas >>

22 Acabamos el estudio de los métodos viendo de forma sencilla cómo acceder a las funciones de librerías externas para poder reutilizar el código lo máximo posible.

NOTICIAS

6. **Flex** Builder y Flash Video Kit
6. **Borland** lanza JBuilder 2005
7. **Microsoft** lanzará el sistema operativo cliente Windows Longhorn en 2006
7. **Abucasis II** se desarrolla sobre Sun.
8. **Data** Migrator Real Time.
8. **BEA** WebLogic Platform ISV Edition.
8. **Herramientas** Macromedia de gestión web.
8. **Acuerdo** HP-Yahoo!
9. **Servidor** Fujitsu PRIMERGY Econel40.
9. **Portátiles** HP Compaq nx9020 y nx9030.
9. **Reproductor** Gmini 400.

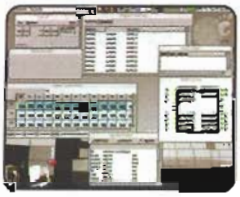
Y ADEMÁS...

Juegos Java: El interfaz gráfico >>

49 En la entrega de este mes y en la de nuestro próximo número vamos a ver cómo trabajar con el interfaz gráfico sobre un sistema móvil, y es que sin un interfaz gráfico ningún juego puede triunfar.

C para Hackers: Programación de sistemas >>

55 Llegamos a la última parte de nuestro curso práctico de C con una iniciación a la programación directa de hardware, en concreto utilizaremos el PIC16F84 de Microchip.



Bases de datos: Subprogramas en Oracle PL/SQL >>

59 Existen dos tipos de subprogramas en PL/SQL, los procedimientos y las funciones, vamos este mes a estudiar mediante ejemplos prácticos cómo implementar ambos tipos.

CUADERNOS DE PRINCIPIANTES

Buscando los datos >>

44 Estudiamos de manera práctica, basándonos en algunos de los más famosos y clásicos algoritmos, los métodos de ordenación de listas y de búsquedas, tanto en listas ordenadas como en no ordenadas. Es un tema fundamental.

NIVEL DE DIFICULTAD:

MATERIAL INCLUIDO EN EL CD-ROM:
Ejemplos ■ Código SQL

Acceso a BB.DD. desde aplicaciones Java JDBC 2.0

MANUEL DOMÍNGUEZ
mdominguez@iberprensa.com

En la anterior entrega comprobamos lo sencillo que resultaba acceder a bases de datos utilizando la API JDBC1.0 de Java.

Sin embargo, con lo visto el rendimiento de nuestra aplicación no era excesivamente bueno y por ello sucesivas versiones de la API JDBC han ido añadiendo funcionalidades extra, que facilitan aún más la integración entre Java y las bases de datos.

MODIFICACIÓN E INSERCIÓN DE DATOS

Para terminar con la serie de ejemplos sobre JDBC 1.0, iniciados en la entrega anterior de **Todo Programación**, vamos a ver cómo realizar inserciones, borrados y modificaciones en la base de datos. Básicamente, la única forma que tenemos de modificar los datos en JDBC 1.0 es mediante el uso de las tradicionales consultas *UPDATE*, *DELETE* e *INSERT* y la utilización de los métodos *ResultSet.executeQuery(SQL)* y *ResultSet.updateQuery(SQL)*. Esto implica que en JDBC 1.0 hemos de hacer una consulta por cada operación que deseemos realizar: modificar, insertar, borrar, etc. Esto se traduce en la realización de muchas comunicaciones con el SGBD que, en caso de estar a mil kilómetros de distancia, se traduciría en un importante retardo en nuestra aplicación.

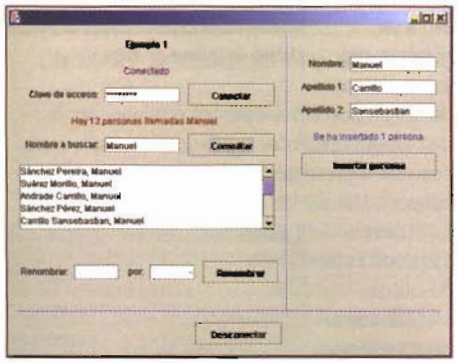
Para aquellos que se incorporen a esta serie en esta entrega, se incluye en el CD de la revista el código SQL necesario para crear la base de datos MySQL de los ejemplos.

En el fichero *Ejemplo1.java* se modifica el *Ejemplo6.java* de la anterior entrega, añadiéndole la posibilidad de insertar una nueva persona a la base de datos. No hay demasiado que resaltar en cuanto al código de esta nueva característica, cuyos aspectos importantes se pueden ver en el *Listado Ejemplo 1*; realmente es una consulta *INSERT SQL* tradicional. Esto a la vez que cómodo, pues permite usar SQL a nuestro antojo, tiene el inconveniente de que no todos los SGBD implementan el estándar SQL de igual forma y por ello la consulta para MySQL podría variar con respecto a la necesaria para PostgreSQL u Oracle (teóricamente no debería, pero en la práctica es así).

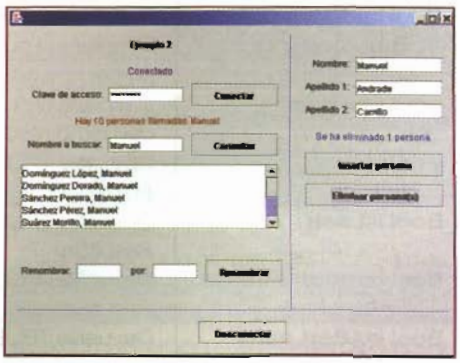
Como último ejemplo de JDBC 1.0, se incluye el fichero *Ejemplo2.java* que modifica *Ejemplo1.java*, añadiéndole la opción de eliminar de la base de datos las personas que cumplan unos ciertos criterios. No haremos más comentarios dado que el código es un calco del escrito para *Ejemplo1.java* salvo que, de nuevo, la cade-

Listado 1. Ejemplo 1

```
private void clicEnInsertar(java.awt.event.ActionEvent evt) {
...
consulta = conexion.createStatement();
int id=0;
ResultSet r = consulta.executeQuery("select max(id) as maxid from
personas ");
if (r.next()) {
id = r.getInt("maxid") + 1;
}
String n = this.jTextField4.getText();
String a1 = this.jTextField5.getText();
String a2 = this.jTextField6.getText();
int afectados = consulta.executeUpdate("insert into personas
values('"+id+"', '"+n+"', '"+a1+"', '"+a2+"')");
...
}
```



Vista de Ejemplo1.java.



Vista de Ejemplo2.java.

ÍNDICE CURSO C PARA HACKERS

- Entrega 1: Acceso a BB.DD. desde aplicaciones Java: JDBC 1.0
- Entrega 2: Acceso a BB.DD. desde aplicaciones Java: JDBC 2.0

En JDBC 1.0 hemos de **consultar** cada operación a realizar: modificar, insertar, borrar, etc.

na de texto SQL es una sentencia *DELETE*, del SQL tradicional. Se presenta este ejemplo para servir de guía a quienes quieran tener una aplicación de referencia que realice todas las operaciones básicas.

INTRODUCCIÓN A JDBC 2.0

JDBC 2.0 trajo muchas mejoras con respecto, sobre todo, al rendimiento y a la abstracción del lenguaje de consulta SQL. Imaginemos una situación en la que tras obtener datos de una base de datos, hemos de modificar ciertos registros de una forma periódica; por ejemplo, un concurso de televisión en el que los concursantes de la semana anterior vuelven a participar ésta. Nuestra aplicación se encarga de gestionar las puntuaciones obtenidas por estos concursantes (tres, por ejemplo) que cambiarán en cada prueba (cada cinco minutos). Con JDBC 1.0 obtendríamos las puntuaciones acumuladas de la semana anterior al principio del concurso, y cada vez que uno de los concursantes modificara su puntuación, se debería hacer una consulta de actualización de la base de datos y otra para volver a recuperar un *ResultSet* con los valores actualizados. Parece más razonable que se pudiesen gestionar en local todas las modificaciones que van ocurriendo de las puntuaciones y, esporádicamente (por seguridad y coherencia) volcar las modificaciones acumuladas a la base de datos JDBC 1.0 no lo permite de forma intrínseca, pero JDBC 2.0 sí.

Por supuesto, poder hacer esto implica que los datos reales y los existentes en la base de datos pueden no estar sincronizados en un momento dado, y por lo tanto es un elemento que se debe valorar bien.

Tratamiento de resultados en JDBC 2.0

Para poder hacer uso de algunas de las nuevas funcionalidades que ofrece JDBC 2.0 con respecto a la clase *ResultSet*, debemos especificarlo al crear el objeto *Statement* que será quien lo devolverá. Esto se hace con una línea como la que sigue (con la nomenclatura seguida en la entrega anterior):

```
Statement consulta =
MiConexion.createStatement
(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
```

Esto permite que a la hora de obtener un *ResultSet*, éste sea acorde a la especificación JDBC 2.0, es decir que puede ser accedido hacia delante, hacia atrás, de forma directa... y que además es sensible a las modificaciones realizadas concurrentemente (hay más opciones). A partir de ahí, cuando obtengamos los resultados de una consulta podremos hacer uso de los métodos añadidos a la clase *ResultSet* por la especificación 2.0. Un listado muy reducido de estos métodos se puede observar en la tabla 1. Profundizaremos en ellos a lo largo de este artículo.

COMENTARIOS SOBRE JDBC 3.0

La API JDBC 3.0 incorpora un abanico de nuevas posibilidades, casi todas encaminadas a la eficiencia y la redundancia de los sistemas. Entre ellas podemos encontrar capacidades para crear *pool* de conexiones, *pool* de transacciones con la base de datos, etcétera, así como funciones para ir convergiendo poco a poco de la arquitectura SPI (*Service Provider Interfaz*, Interfaz de Proveedor de Servicios) a la nueva arquitectura propuesta por Javasoft: la arquitectura de conector. Estas mejoras son opciones avanzadas que para el usuario novel en el funcionamiento JDBC quedan muy alejadas. No es el cometido de esta miniserie profundizar con tal nivel de detalle, sino orientar a todo aquél que desee incluir el acceso a bases de datos en sus desarrollos Java. Dejamos por tanto al lector el profundizar en este tema, para lo cual encontrará abundante información en la web de Javasoft (*http://java.sun.com*).

APLICACIÓN COMPLETA: LIBRERÍA

Queremos hacer una aplicación, llamada *Librería* que nos permitirá insertar, modificar, eliminar, buscar y listar registros de libros para, por ejemplo, tener clasificada nuestra biblioteca personal. Usaremos este ejemplo real para ver las nuevas características de JDBC 2.0.

Listado de la base de datos.

Tendremos una base de datos MySQL llamada *librería* con una única tabla llamada *libros*, el usuario, *usuarioTP*, y la clave que deseemos. El código SQL para la creación de la base de datos se adjunta en el CD de la revista en un fichero llamado *SQL_Libros.zip*.

Vista del interfaz de Librería 1.0.

Tabla 1. Resumen de nuevos métodos de ResultSet

Método	Descripción
Bool absolute(int fila)	Desplaza el puntero del ResultSet a la fila especificada.
Bool relative (int posiciones)	Desplaza el puntero del ResultSet hacia delante o atrás las posiciones especificadas.
Bool previous()	Desplaza el puntero del ResultSet una posición hacia atrás.
Bool first()	Desplaza el puntero del ResultSet a la primera posición.
Bool last()	Desplaza el puntero del ResultSet a la última posición.
Void beforeFirst()	Desplaza el puntero del ResultSet una posición antes del primer resultado.
Void afterLast()	Desplaza el puntero del ResultSet una posición después del último resultado
Bool isFirst()	Devuelve TRUE si se está en el primer resultado del ResultSet.
Bool isLast()	Devuelve TRUE si se está en el último resultado del ResultSet.
Bool isBeforeFirst()	Devuelve TRUE si se está antes del primer resultado del ResultSet.
Bool isAfterLast()	Devuelve TRUE si se está después del último resultado del ResultSet.
Int getRow()	Devuelve la posición actual del puntero del ResultSet.
Void updateXXX (columna, valor) (XXX = tipo de datos)	Actualiza la columna indicada, con el valor indicado, en la fila del ResultSet apuntada por el puntero.
Void updateRow()	Actualiza en la base de datos la fila apuntada por el puntero del ResultSet (con las modificaciones que se hayan hecho).
Void deleteRow()	Elimina del ResultSet y de la base de datos la fila apuntada por el puntero del ResultSet.

Por sencillez, se utilizará el mismo interfaz para todas las opciones de la aplicación; lo que nos interesa es lo que se hace por dentro. En la siguiente figura anterior se muestra una vista del interfaz de la aplicación.

Listado de libros

En el Listado 2 que muestra el código importante de la opción *Listar* de la aplicación, observamos que utiliza las clases y los métodos JDBC que hemos visto para JDBC 1.0. JDBC 2.0 especializa lo planteado por JDBC 1.0 por lo que hereda sus métodos y atributos, que siguen funcionando. Así, en esta opción se hace una consulta SQL tradicional y se recorre de principio a fin el *ResultSet* obtenido mediante el uso de *ResultSet.next()*. Al crear la conexión se hace uso de *MiURL*, *MiUsuario*, *MiClave*, tres atributos de la clase que se definen en el constructor de la misma, para no tener que hacerlo en todos los sitios en que se use. En realidad, al hacer clic sobre cualquiera de las opciones, no se ejecuta la función en el momento, sino que se marca la función que se desea y es luego al pulsar sobre el botón de abajo a la izquierda (que cambia de texto según el contexto) cuando realmente se ejecuta lo deseado.

Buscar libros

La búsqueda se lleva a cabo de forma similar al listado. Se realiza una consulta SQL basada, en este caso, en lo que exista en los campos de texto donde se ha debido introducir el criterio de búsqueda. Gran parte del código de la aplicación se invierte en el interfaz de usuario (pese a ser muy pobre), y otra gran parte en validar el contenido de los cuadros de texto para formar las consultas. En la búsqueda, la consulta genera un *ResultSet* acorde a la API JDBC 2.0, como se puede apreciar en el Listado 3 que contiene el código importante de la operación de búsqueda.

Listado 2. Código importante de la opción Listar

```
private void clicEnListar(java.awt.event.ActionEvent evt) {
...
String sql = new String("select * from `libros` order by id");
conexion = DriverManager.getConnection(MiURL, MiUsuario, MiClave);
consulta = conexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
           ResultSet.CONCUR_UPDATABLE);
resultado = consulta.executeQuery(sql);
if (resultado.next()) {
identificador.setText(resultado.getString("id"));
titulo.setText(resultado.getString("titulo"));
autor.setText(resultado.getString("autor"));
editorial.setText(resultado.getString("editorial"));
anio.setText(resultado.getString("anio"));
mensaje.setText(sql);
} else {
mensaje.setText("¡No hay libros en la base de datos!");
}
...
}
```

Listado 3. Código importante de la operación de búsqueda

```
private void realizarBusqueda() {
...
conexion = DriverManager.getConnection(MiURL, MiUsuario, MiClave);
consulta = conexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
           ResultSet.CONCUR_UPDATABLE);
resultado = consulta.executeQuery(sql);
if (resultado.next()) {
identificador.setText(resultado.getString("id"));
titulo.setText(resultado.getString("titulo"));
autor.setText(resultado.getString("autor"));
editorial.setText(resultado.getString("editorial"));
anio.setText(resultado.getString("anio"));
} else {
mensaje.setText("¡No se han encontrado coincidencias!");
}
...
}
```

Modificar libros

La modificación sí tiene aspectos a destacar. Esta función realiza un listado completo de la base de datos y permite modificar los datos de aquel registro que se muestre en ese momento en el interfaz de usuario. En JDBC 1.0 tendríamos que haber modificado el registro mediante la confección de una sentencia SQL de tipo *UPDATE*; en JDBC 2.0 no es necesario. Tenemos el método *ResultSet.updateXXX(columna, nuevoValor)* que nos permite modificar el valor de una columna del registro del *ResultSet* al que apunta el puntero (del *ResultSet*, claro) en ese momento; *XXX* puede ser *String*, *Float*, *boolean*... Por ejemplo:

```
r.updateString("titulo", "Java 2:
Manual de referencia");
```

Modificaría el registro apuntado del *ResultSet* "r" de tal forma que el campo "título" quedase con el valor "Java 2: Manual de referencia".

Este método no modifica la base de datos, solo el *ResultSet*. Para cambiar realmente todo en la base de datos tenemos el método *ResultSet.updateRow()* que actualiza el registro que está siendo apuntado en ese momento del *ResultSet*. El código es muy sencillo y ni siquiera hacer falta saber SQL. Se puede observar en el Listado 4 de la página siguiente. Siguiendo con el ejemplo:

```
r.updateRow();
```

Actualizaría la base de datos con los datos del registro apuntado actualmente en el *ResultSet*.

La ventaja que ofrece JDBC 2.0 es que podemos hacer tantos *updateXXX(...)* como deseemos, sin que ello conlleve ninguna comunicación con la base de datos y luego, en un momento dado, recorrer el *ResultSet* entero actualizando los datos nuevos mediante consecutivos *updateRow()*.

Eliminar libros

Eliminar datos es aún más sencillo que la modificación. En este caso la aplicación uti-

liza el método *ResultSet deleteRow()* proporcionado por la API JDBC 2.0. En JDBC 1.0 tendríamos que haber usado una sentencia SQL de tipo *DELETE*. Aquí no es necesario saber SQL. En el código asociado a la opción de eliminar, que se encuentra en el CD de la revista, se hace uso de este método de la siguiente forma:

```
resultado.deleteRow();
this.mensaje.setText("¡Entrada
eliminada correctamente!");
```

Donde incluso la segunda línea es simplemente una ayuda visual. Con ello se ha eliminado del *ResultSet* y de la base de datos un registro: el que se mostraba en ese momento en el interfaz de la aplicación.

Vista tras eliminar un libro.

Insertar libros

La inserción en la aplicación Librería 1.0 se podría haber realizado con JDBC 2.0, donde la clase *ResultSet* proporciona unos métodos para realizar esta operación. Concretamente estos métodos son *ResultSet.moveToInsertRow()*, *ResultSet.insertRow()* y *ResultSet.moveToCurrentRow()*. El primero de ellos inserta un nuevo registro en el *ResultSet* y coloca el puntero apuntando a él; mediante los conocidos *updateXXX()* podemos dar valores a este nuevo registro. El segundo inserta el registro recién creado en la base de datos. El tercero vuelve a colocar el puntero apuntando al lugar donde estaba antes de llamar a *ResultSet.moveToInsertRow()*.

En nuestro caso lo hemos hecho en SQL puro y duro, con JDBC 1.0, demostrando así dos cosas: que JDBC 1.0 y JDBC 2.0 pueden coexistir en la misma aplicación, y que JDBC 2.0 facilita enormemente la tarea de programar consultas a bases de datos.

Segundos antes de insertar un nuevo registro.

Avanzar

Cómo moverse hacia delante es una operación que está soportada en el *ResultSet* desde JDBC 1.0 y heredada en las posteriores especificaciones a través del método *ResultSet.next()*; así es como lo realiza nuestra aplicación. En el Listado 5 se puede observar el código de esta opción en Librería 1.0.

Retroceder

Moverse hacia atrás por el *ResultSet* es una cosa que no estaba contemplada en JDBC 1.0, de tal forma que si cargabas un *ResultSet* en memoria no podías recorrerlo más que una vez antes de tener que realizar la consulta de nuevo, o bien construir una estructura en memoria alternativa. En JDBC 2.0 esta situación está más que superada y se proporcionan muchos métodos para desplazarse en cualquier dirección y de cualquier modo por el *ResultSet*. En concreto, nuestra aplicación utiliza *ResultSet.previous()* para retroceder en el *ResultSet*. De hecho la función es exactamente igual que la de avanzar pero cambia la llamada a este método, que ahora es:

```
if (resultado.previous()) {
...
// Hago cosas con resultado...
...
}
```

Listado 4. Código JDBC para modificar

```
private void realizarModificar() {
...
resultado.updateString("titulo", this.titulo.getText());
resultado.updateString("autor", this.autor.getText());
resultado.updateString("editorial", this.editorial.getText());
resultado.updateString("anio", this.anio.getText());
resultado.updateRow();
this.mensaje.setText("¡Entrada actualizada correctamente!");
...
}
```

Listado 5. Código importante para avanzar

```
private void clicEnSiguiente(java.awt.event.MouseEvent evt) {
if (resultado != null) {
if (resultado.next()) {
identificador.setText(resultado.getString("id"));
titulo.setText(resultado.getString("titulo"));
autor.setText(resultado.getString("autor"));
editorial.setText(resultado.getString("editorial"));
anio.setText(resultado.getString("anio"));
}
}
}
```

CÓMO DISTRIBUIR NUESTRA APLICACIÓN

Hay que tener en cuenta que cuando nuestra aplicación esté terminada y la vayamos a instalar en casa del cliente, es necesario que éste tenga instalada dos cosas: Java Runtime Enviroment, con la MVJ que ejecutará el software, y *Connector/J*, el driver JDBC para MySQL, tal y como se explicó en la entrega anterior ¿Es un engorro? Bueno, en realidad no mucho; más tedioso aún es utilizar ODBC en lugar de JDBC y tener que configurar conexiones ODBC en cada sistema.

CONCLUSIONES

JDBC es muchísimo más extenso de lo que se ha comentado en esta miniserie de dos entregas. Sin embargo, el desarrollador de software de pequeña envergadura (la gran mayoría) no usará más de lo aquí expuesto. Con estos conocimientos es sencillo tener una base de datos como *backend* de nuestra aplicación. Esto nos permitirá realizar aplicaciones Java que conecten con esos datos, aplicaciones web (PHP, ASP...) que hagan lo mismo, etcétera. Éstas son las bases para la construcción de sistemas de información modernos.

La mejor formación técnica apta para todos

FORMACIÓN 100% GARANTIZADA

1 Aprende a programar tus propios juegos

Programación y diseño de Videojuegos

- La primera obra para aprender a desarrollar un videojuego 3D completo incluyendo gráficos, animaciones, menús y toda la programación.
- Formación en la herramienta de programación Blitz 3D, incluida en el primer CD-ROM.
- Se incluye el desarrollo paso a paso con todo el código fuente de un juego de acción 3D: Zone of Fighters.

COMPOSICIÓN: 20 coleccionables en dos tomos de 200 páginas y 20 CD-ROMS
NIVEL: Principiantes a nivel medio
TEMARIO: Zona gráficos, Zona desarrollo, Blitz 3D, Historia del videojuego, cuestionario y desarrollo del juego Zone of Fighters
REQUERIMIENTOS: Pentium 200 o superior. 64 Mb RAM. Windows 98/Millenium/XP
PVP: ~~120 euros~~.
OFERTA SUSCRIPCIÓN: 107,82 euros
Incluye las tapas archivadoras de regalo.

2 Creatividad publicitaria aplicada al diseño gráfico y web

Diseño publicitario

- Una obra práctica que cubre todos los aspectos y temas necesarios para dominar el diseño gráfico publicitario actual.
- Aprende las técnicas, los métodos y los trucos que utilizan los profesionales del diseño en el día a día de una forma sencilla, práctica y eficaz.
- Realiza los ejercicios paso a paso, empezando a diseñar de forma profesional con un estilo actual.

COMPOSICIÓN: 20 coleccionables en dos tomos de 200 páginas y 20 CD-ROMS
NIVEL: Principiantes a nivel medio
TEMARIO: Teoría de la publicidad, diseño editorial, diseño corporativo, diseño web y zona de tutoriales
REQUERIMIENTOS: Pentium 200 o superior. 64 Mb RAM. Windows 98/Millenium/XP
PVP: ~~120 euros~~.
OFERTA SUSCRIPCIÓN: 101,85 euros
Incluye las tapas archivadoras de regalo.

3 El primer curso para usuarios de videocámaras digitales

Vídeo digital

- Curso orientado a usuarios de videocámaras digitales y entusiastas de la edición digital.
- Aprende a trabajar con los programas más importantes de tratamiento de vídeo.
- Cómo grabar, editar y montar correctamente tus propios videos.
- No son necesarios conocimientos previos para seguir este curso.

COMPOSICIÓN: 20 coleccionables en dos tomos de 200 páginas y 20 CD-ROMS
NIVEL: Principiantes
TEMARIO: Teoría de vídeo digital, formatos, compresiones, técnicas de edición, montaje, efectos, transiciones. Prácticas de producción de vídeo. Tutoriales sobre las herramientas que se incluyen en los CD-ROMs
REQUERIMIENTOS: Pentium 200 o superior. 64 Mb RAM. Windows 98/Millenium/XP
PVP: ~~120 euros~~.
OFERTA SUSCRIPCIÓN: 99,95 euros
Incluye las tapas archivadoras de regalo.

Cupón de pedido (Fotocopie y rellene el siguiente cupón de pedido)

Nombre: ..... Apellidos: .....

Dirección: ..... Tel.: ..... E-mail: ..... @ .....

Población: ..... C.P.: ..... Provincia: .....

Corta por la línea de puntos, pégalo y envíalo al Apartado F.D. 1 (28110 Algete). (No necesita sello).

(No olvides firmar tu cupón). (Para menores se requiere firma paterna). Oferta válida durante 6 meses desde la publicación. La información que nos facilites será incluida en un fichero automatizado, tienes el derecho de acceso, rectificación y cancelación dirigiéndote a Studio Press, S.L. C/ del Río Ter. Nave 13. Polígono "El Nogal". 28110 Algete (Madrid). (Ley de protección de datos).

- ❑ Adjunto cheque a nombre de STUDIO PRESS, S.L.
- ❑ Domiciliación bancaria, código de la cuenta:

❑❑❑❑ ❑❑❑❑ ❑❑ ❑❑❑❑❑❑❑❑❑❑

❑ Visa ❑ 4B ❑ Master Card Fecha de caducidad: ❑❑ ❑❑

(Visa electrón no válida) ❑❑❑❑ ❑❑❑❑ ❑❑❑❑ ❑❑❑❑

Firma

Pedido de:
- ❑ Programación y diseño de Videojuegos 107,82 €
- ❑ Diseño publicitario 101,85 €
- ❑ Vídeo Digital 99,95 €

Total Pedido ..... €
Marque con una "X" su pedido.

Por Teléfono: 916280203 (de 9 a 14h y de 15h. a 18 h.) Por Fax: 916280935 Por Correo electrónico: suscripciones@iberprensa.com