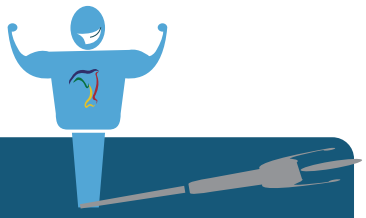
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

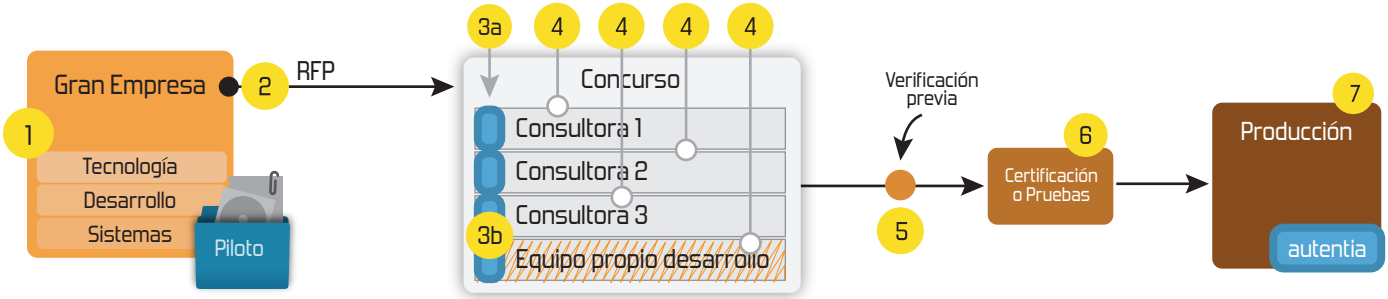
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por: Francisco Javier Martínez Páez

Puedes encontrarme en Autentia
Somos expertos en Java/J2EE
Contacta en info@autentia.com

autentia real business solutions

Descargar este documento en formato PDF cachedRowSetJDBC.pdf

Firma en nuestro libro de Visitas

CachedRowSet: JDBC y Java 5.

Los ejemplos de este tutorial están hechos con el siguiente entorno de desarrollo:

- Jboss Eclipse IDE Milestone 5.
- JDK 1.5
- MySQL 5.0
- MySQL Administrator (opcional)
- MySQL Connector/J (Driver tipo 4 que implementa la versión JDBC 3.0)

INTRODUCCIÓN

Entre las características nuevas introducidas en Java 5 se incluyen algunos interfaces para mejorar el API de JDBC, incluidas en paquete de extensiones javax.sql. Nos vamos a fijar más en concreto en los CachedRowSets. Evidentemente, al ser interfaces, será cuestión de cada fabricante de Base de datos implementar estas nuevas características en sus drivers.

¿Qué es un CachedRowSet?

Podríamos definir un CachedRowSet como un objeto para contener filas de datos o registros en memoria sin tener que estar conectado a la fuente de datos de donde han sido obtenidos (a diferencia de un ResultSet). Además, sigue el modelo de JavaBean y pueden ser serializados (al estar desconectados), además de permitirnos (si el driver lo permite) actualizar los datos (updatables) y recorrerlos hacia adelante y hacia atrás (scrollables). Lo más tipico, es rellenar un CachedRowSet a partir de un ResultSet, pero no es la única posibilidad, ya que también podemos rellenar el CachedRowSet a partir de otras fuentes de datos como ficheros tabulados, etc.

MANOS A LA OBRA

Para los ejemplos del tutorial vamos a hacer uso de la tabla usuarios, generada en el tutorial:
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=JDBCResultset

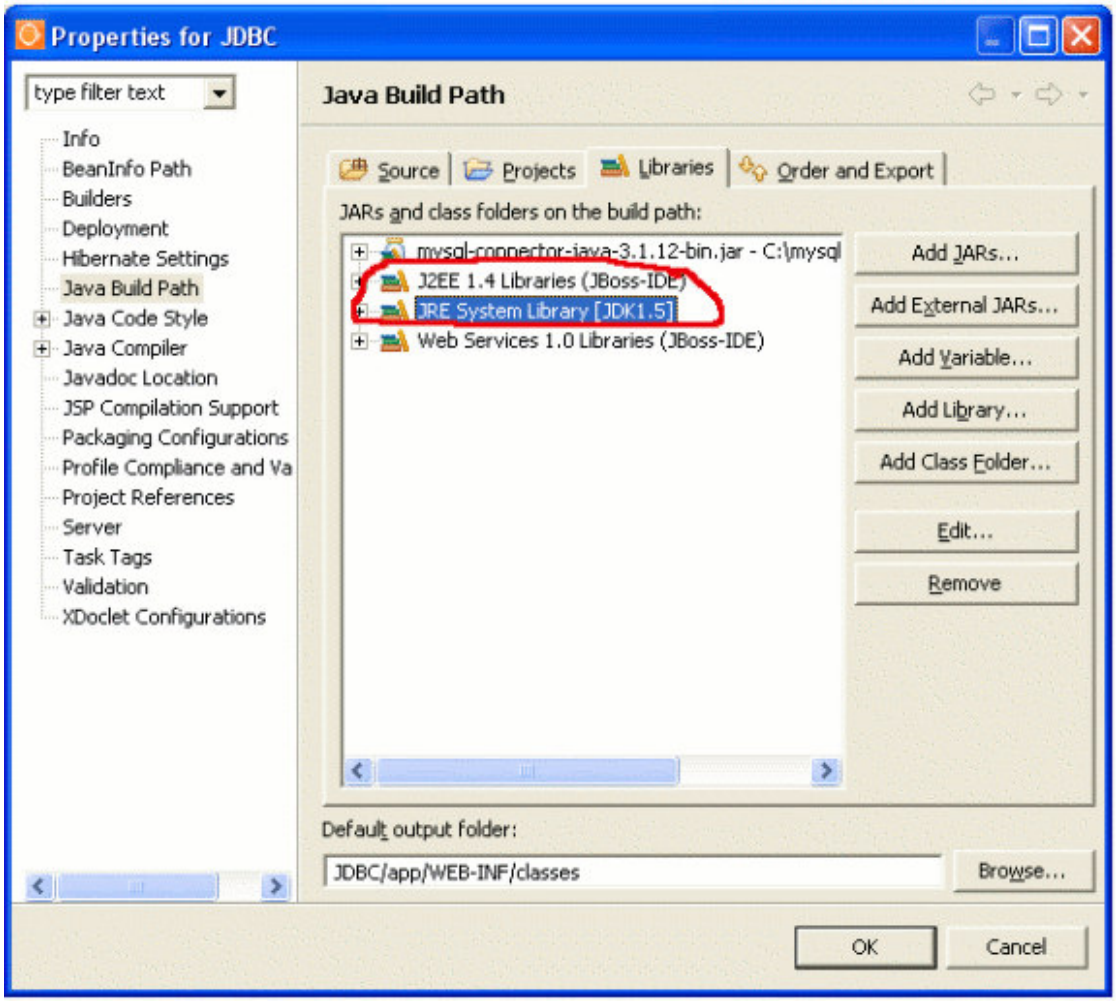
También usaremos el método getConnection():

```
protected static String dbClass = "com.mysql.jdbc.Driver";
protected static String dbUrl = "jdbc:mysql:///paco";

protected Connection getConnection() {
```

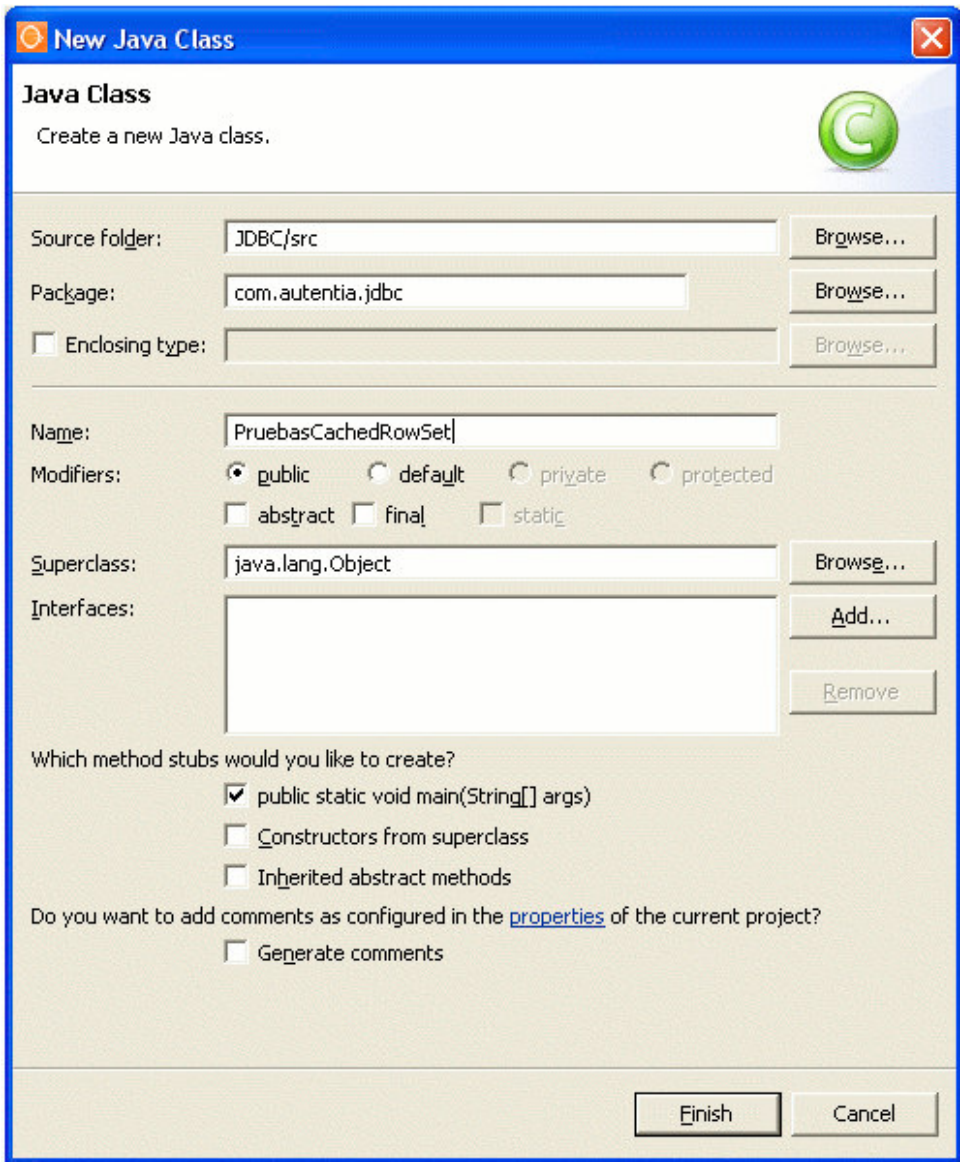
```
        Properties props = new Properties();
        props.put("user","<tu_usuario>");
        props.put("password","<tu_password>");
        Connection conBBDD = null;

        try {
            Class.forName(dbClass);
            conBBDD=DriverManager.getConnection(dbUrl, props);
        } catch(Exception e) {
             return null;
        }
        return conBBDD;
    }
```

Pues vamos a empezar con los ejemplos:

Lo primero será comprobar que nuestro proyecto está usando la JDK 1.5:



Ahora, crearemos una clase que denominaremos PruebasCachedRowSet:



Rellenando el CachedRowSet.

Vamos a crearnos un método que recibe una consulta y retorna un CachedRowSet con los datos de la consulta:

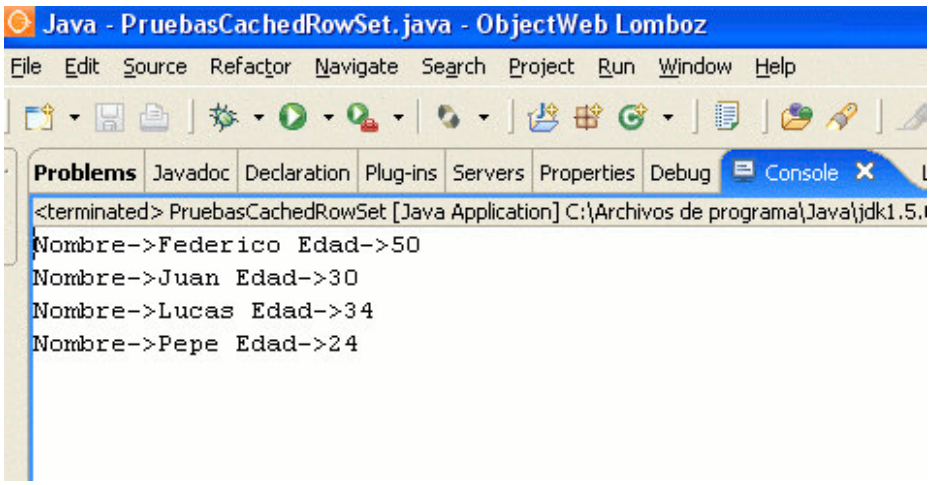
```
public CachedRowSet rellena(String query) throws Exception {

        CachedRowSetImpl crs = null;
        Connection conBBDD = null;
        Statement st = null;
        ResultSet rs = null;

        try {

            // Creamos la conexión a Base de Datos.
            conBBDD = getConnection();

            st = conBBDD.createStatement();

            // Obtenemos los datos:
            rs = st.executeQuery(query);

            // Cremos un objeto CachedRowSetImpl (implementación de CachedRowSet)
            crs = new CachedRowSetImpl();

 /* Rellenamos el CachedRowSet con los datos del ResultSet. Por defecto, un CachedRowSet  es scrollable y updatable, incluso
aunque el ResultSet desde donde ha sido rellenado no lo sea. */
            crs.populate(rs);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if(rs!=null) {
                try {
                    rs.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(conBBDD!=null) {
                try {
                    conBBDD.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }

            }
        }
        return crs;
    }
```

Invocamos el método desde main:

```
public static void main(String[] args) {

        PruebasCachedRowSet pcrs = new PruebasCachedRowSet();
        String query = "SELECT * FROM USUARIOS";
        try {
            CachedRowSet crs = pcrs.rellena(query);
            while (crs.next()) {
                String nombre = crs.getString(1);
                int edad = crs.getInt(2);
                System.out.println("Nombre->"+nombre+ " Edad->"+edad);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
```

Ejecutamos y mostramos la consola:



Pero, ¿ es necesario usar ResultSet ?

Vamos a crear un nuevo método que llamaremos rellena2(String query):

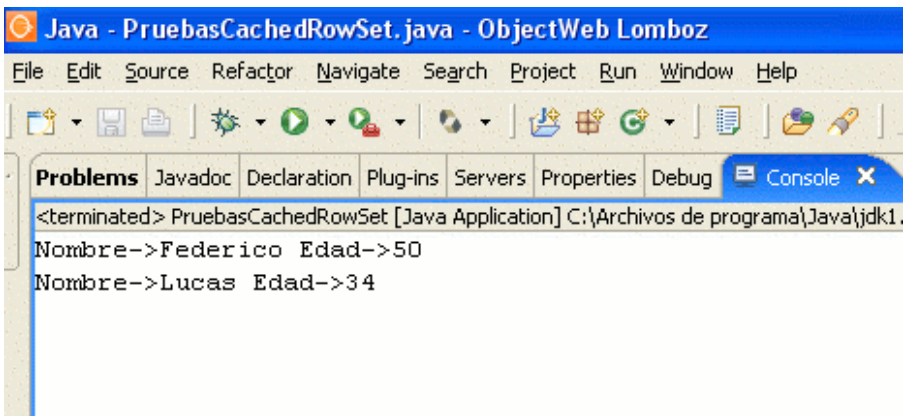
```
public CachedRowSet rellena2(String query) throws Exception {

        CachedRowSetImpl crs = null;
        Connection conBBDD = null;
        try {
```

```
            // Creamos la conexión a Base de Datos.
            conBBDD = getConnection();

            // Cremos un objeto CachedRowSetImpl (implementación de
            // CachedRowSet)
            crs = new CachedRowSetImpl();

            // Le decimos al CachedRowSet el comando SQL con el que se ha de
            // rellenar.
            crs.setCommand(query);

            // Ejecutamos la consulta usando la conexión creada previamente.
            crs.execute(conBBDD);

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (conBBDD != null) {
                try {
                    conBBDD.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
        return crs;
    }
```

Invocamos el método desde main (y modificamos la consulta):

```
public static void main(String[] args) {

        PruebasCachedRowSet pcrs = new PruebasCachedRowSet();
        String query = "SELECT * FROM USUARIOS WHERE EDAD > 30";
        try {
            CachedRowSet crs = pcrs.rellena2(query);
            while (crs.next()) {
                String nombre = crs.getString(1);
                int edad = crs.getInt(2);
                System.out.println("Nombre->"+nombre+ " Edad->"+edad);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
```

Ejecutamos y mostramos la consola:



He oido que implementa paginación, ¿es cierto?

Pues si, es cierto, pero antes, rellenemos la tabla con más datos, yo usaré un método del otro tutorial ya mencionado que me inserta filas en la tabla.

Una vez rellena la tabla, creemos un método que llamaremos:

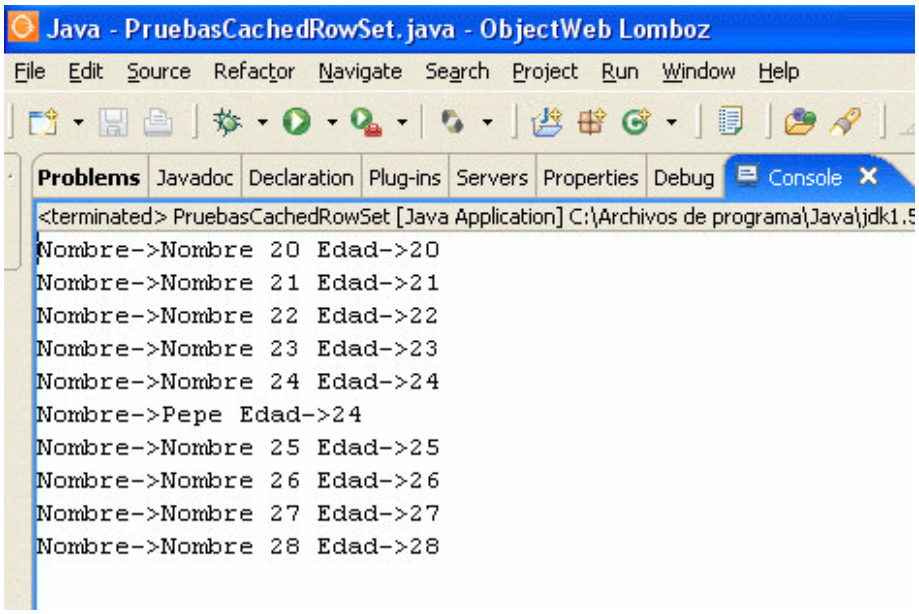
```
public CachedRowSet rellenaPaginando
```

```
        (String query, int numPagina, int numRegPagina) {

    CachedRowSetImpl crs = null;
    Connection conBBDD = null;
    try {

        // Creamos la conexión a Base de Datos.
        conBBDD = getConnection();

        // Cremos un objeto CachedRowSetImpl (implementación de
        // CachedRowSet)
        crs = new CachedRowSetImpl();

        // Le decimos al CachedRowSet el comando SQL con el que se ha de
        // rellenar.
        crs.setCommand(query);

        // Le indicamos el número de registros por página:
        crs.setPageSize(numRegPagina);

        // Ejecutamos la consulta usando la conexión creada previamente.
        crs.execute(conBBDD);

        int paginaActual = 1;

        // nos movemos hasta la página solicitada
        while(crs.nextPage() && paginaActual < numPagina) {
            paginaActual++;
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {

        if (conBBDD != null) {
            try {
                conBBDD.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }

        }
    }
    return crs;
}
```

Invocamos el método desde main (y modificamos la consulta).
Le indicaremos que nos muestre la página 3 paginando de 10 en 10:

```
public static void main(String[] args) {

    PruebasCachedRowSet pcrs = new PruebasCachedRowSet();
    String query = "SELECT * FROM USUARIOS ORDER BY EDAD";
    try {
        CachedRowSet crs = pcrs.rellenaPaginando(query,3,10);
        while (crs.next()) {
            String nombre = crs.getString(1);
            int edad = crs.getInt(2);
            System.out.println("Nombre->"+nombre+ " Edad->"+edad);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

Ejecutamos y mostramos la consola:

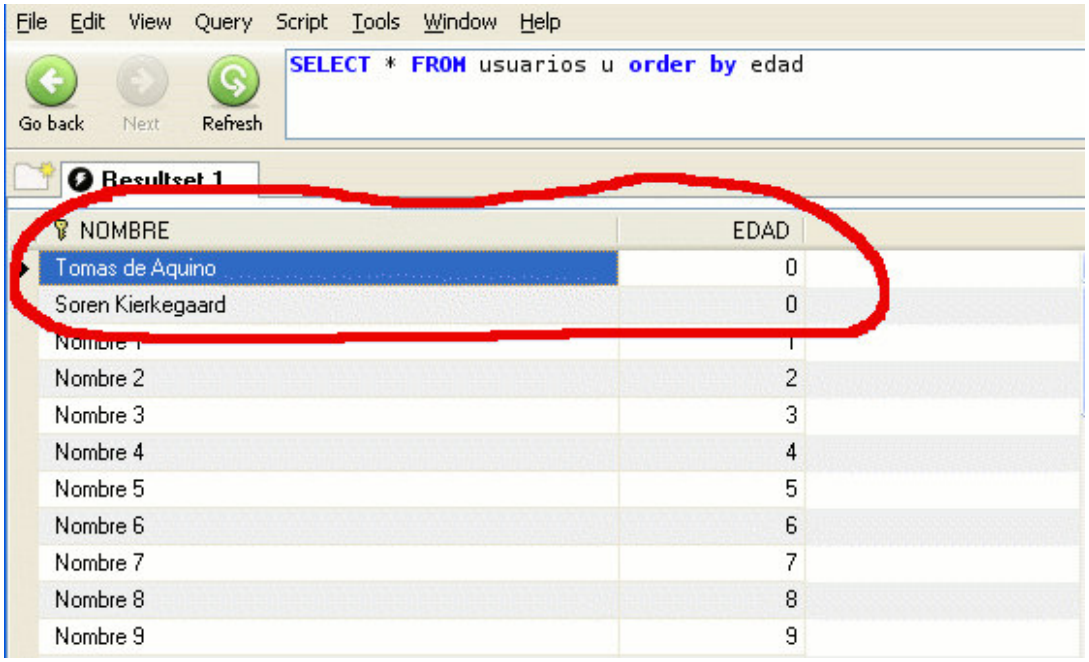


Además, son updatables:
Vamos a hacer uso de esta característica. Lo haremos directamente sobre la función main y comentaremos el código:

```
public static void main(String[] args) {

      PruebasCachedRowSet pcrs = new PruebasCachedRowSet();
     String query = "SELECT * FROM USUARIOS WHERE NOMBRE = 'Nombre 0' ";
      Connection conBBDD = null;

     try {
         // Obtenemos una conexión.
         conBBDD = pcrs.getConnection();

         // Obtenemos un CachedRowSet usando el método ya creado.
         CachedRowSet crs = pcrs.rellena2(query);

         // Nos situamos sobre el primer registro:
         crs.first();

         // Modificamos su nombre:
         crs.updateString(1,"Soren Kierkegaard");
         // Actualizamos el CachedRowSet.
         crs.updateRow();

         // Nos situamos en la "Insert Row"
         crs.moveToInsertRow();

         // Insertamos los datos del nuevo registro
         crs.updateString(1, "Tomas de Aquino");
         crs.updateInt(2, 0);
         // Actualizamos el CachedRowSet.
         crs.insertRow();

         // Volvemos a la fila inicial
         crs.moveToCurrentRow();

         // Enviamos los cambios a la base de datos.
         crs.acceptChanges(conBBDD);

     } catch (Exception e) {
         e.printStackTrace();
     } finally {
         if (conBBDD != null) {
             try {
                 conBBDD.close();
             } catch (SQLException e) {
                 e.printStackTrace();
             }

         }
```

```
    }
  }
```

Ejecutamos y vemos la tabla usuarios:



Bueno, ya hemos enseñado algunas de las características más importantes de los CachedRowSet. Espero que os haya servido de ayuda...

Lo de siempre, si necesitáis ayuda: http://www.autentia.com

Puedes opinar sobre este tutorial aquí

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	**Descripción**
Introducción a JDBC	En este tutorial os explicamos los fundamentos teóricos de JDBC
Generación automática de código JDBC	En este tutorial os enseñamos como, sin conocimiento de JDBC, crear vuestro programas en Java, gracias a JDBCTest.
JDBC y MySql	En el tutorial anterior vimos como instalar MySQL en Windows, ahora vamos a ver como acceder desde una aplicación Java.
Paginar un ResultSet	En este tutorial aprenderemos a paginar un ResultSet de manera que tengamos los datos tabulados y ordenados
Algunas características menos conocidas del api JDBC 2.0	En este tutorial veremos algunas de las caraterísticas que menos se conocen del api JDBC.