UD9

BD Relacionais

Marcia Fernández

IES Antón Losada Diéguez 21/22

Actividade

Acceso básico a bases de datos

Supondo que xa sabemos conectarnos a unha base de datos, os nosos programas poden comezar a facer consultas de lectura e actualización.

O obxecto Connection converterase nun enlace directo coa base de datos a través do cal enviar ordes utilizando instancias do obxecto Statement.

Un obxecto Statement será unha porta aberta para enviar todo tipo de instrucións á base de datos. Só no momento de envialo ao servidor estará claro que tipo de sentenza estamos pedindo que se execute. Nese momento será imprescindible distinguir entre as sentenzas de lectura que devolverán información, e para as que haberá que ter preparado un obxecto ResultSet e as sentenzas de modificación, que devolverán simplemente un número enteiro.

As clases fundamentais para manexar o tratamento de datos almacenados en SXBDR serán, ademais de Connection, serán polo tanto, Statement e ResultSet.

Clase Statement

A clase Statement é a máis básica das que existen para representar sentenzas SQL.

Proporciona tres métodos de execución, que se utilizarán dependendo do tipo de sentenza;

- executeQuery() utilizarase para as sentenzas de lectura, aquelas que esperamos que nos devolvan información que estaba almacenada na base de datos;
- executeUpdate() será o método para as sentenzas de actualización da base de datos, das que só esperamos como resultado un número enteiro que indique a cantidade de filas que se viron afectadas polo cambio;
- execute() xenérico que se utilizará nas situacións nas que o programador non sabe a priori que tipo de sentenza se executará, senón que se trata de sentenzas xeradas dinamicamente.

Clase ResultSet

Un obxecto ResultSet está preparado para conter as filas de datos devoltas por unha consulta enviada a unha base de datos a través dunha conexión válida. Trátase dunha clase sinxela que só proporciona métodos simples para acceder á información coñecendo ou ben o nome o ben a posición da columna que nos interese en cada momento. No nome destes métodos vai incluído o tipo de datos que queremos obter como resultado, ao que será convertido, se é posible, a información orixinal da base de datos. Por exemplo, e como veremos máis adiante, este código

```
rs.getString(1)
```

recuperará o contido da columna na primeira posición (número 1) dentro da fila actual e, se é preciso e posible, vaino transformar nun String.

A lista de métodos para pedirlle a un RecordSet a recuperación dun dato é longa, pero o que nos interesa agora é recalcar a diferenza entre a información almacenada na base de datos e a que se manexará dentro do programa. Normalmente haberá unha relación moi directa entre eles, pero non necesariamente ten por que ser así.

Metadatos

En moitas ocasións, ademais dos datos introducidos nun ResultSet pódenos interesar tamén manexar información acerca do seu contido, o que chamamos metadatos, datos sobre os datos.

Un obxecto da clase ResulSetMetaData, devolto polo método getMetaData(), conterá información útil sobre o contido e estrutura dun ResultSet que pode ser importante para o desenvolvemento do noso programa.

Vexamos, por exemplo, os seguintes catro métodos:

getColumnCount()	Devolve o número de columnas que hai no ResultSet.
getColumnLabel(int col)	Devolve o título dunha columna, para usalo directamente na presentación de datos.
getColumnName(int col)	Devolve o nome dunha columna.
getColumnTypeName(int col)	Devolve o nome do tipo de dato SQL dunha columna.

Con eles podemos ter suficiente información para amosar os resultados de calquera consulta dun xeito razoable.

Moverse polos datos

O procedemento para recuperar información dunha base de datos consistirá en determinar a consulta SQL que é preciso executar, asociala cun obxecto Stament e enviala ao servidor utilizando un obxecto Connection. O conxunto de datos que conforma a resposta non se incorporarán directamente ao noso programa, senón que recibiremos os recursos para manexalos en forma de obxecto ResultSet. Unha peza fundamental deste obxecto é o punteiro que indica cal será a seguinte fila a ser tratada, é dicir, que datos deberán recuperar os métodos get que víamos no apartado de introdución (getInt, getString, etc.).

Aínda que é posible moverse polo contido dun ResultSet en ambos sentidos, o tratamento máis habitual inclúe sucesivas chamadas ao método next() para mover o punteiro á seguinte fila a ser utilizada, como veremos no seguinte exemplo.

Primeiro exemplo básico

```
package films;
import java.sql.*;
 public class Select {
   static final String MYSQLUSER = "root";
   static final String MYSQLPASS = "root";
    public static void main(String args[]) {
      String mysqlUrl = "jdbc:mysql://localhost/peliculas";
      String driver = "com.mysql.jdbc.Driver";
      Connection mysqlCon = null;
      try {
      Class.forName(driver).newInstance( );
      } catch( Exception e ) {
      System.out.println("Non se atopou o driver de MySQL.");
      return;
      }

      try {
      mysqlCon = DriverManager.getConnection(mysqlUrl, MYSQLUSER,
MYSQLPASS);
      Statement mysqlSelect = mysqlCon.createStatement( );
      ResultSet mysqlResult = mysqlSelect.executeQuery("SELECT * FROM
films");
```

```
            System.out.println("Listado de películas");
            int contador = 0;
        while(mysqlResult.next( )) { // procesa cada fila do resultado
              System.out.print(++contador + ": " + mysqlResult.getInt(1));
              System.out.print(", " + mysqlResult.getString(2));
              System.out.println(", " + mysqlResult.getInt("ano"));
        }
        } catch( SQLException e ) {
              System.err.println(e.getMessage( ));
        } finally {
        if( mysqlCon != null ) {
              try { mysqlCon.close( ); }
              catch( Exception e ) { e.printStackTrace( ); }
        }
        }
    }
}
```

Neste primeiro exemplo inclúese todo o necesario para un tratamento básico de información extraída da base de datos. A consulta SQL que se precisa é simplemente un String para o noso programa. Para que esa consulta se poda enviar á base de datos a través da conexión previamente establecida, precísase un obxecto Statement, creado polo método createStatement() da conexión. O novo obxecto terá a capacidade de enviar consulta de lectura utilizando o método executeQuery() e haberá que ter preparado un obxecto ResultSet no que recibir a información sobre o resultado da consulta.

O punteiro que indica cal será a seguinte fila a tratar queda situado antes da primeira, de tal xeito que a seguinte chamada que se faga a next() permita movelo aos primeiros datos a utilizar. En caso de que non houbese ningunha fila no resultado, esta chamada a next() xa alcanzaría o final do ResultSet.

O tratamento de cada unha das filas consistirá en extraer os datos que precisemos, converténdoos ao tipo que creamos máis conveniente. Determinaremos a columna que queremos utilizar ben referíndonos a ela pola súa posición ou ben polo nome que teña na base de datos e dependendo do tipo de dato que queiramos obter chamaremos ao correspondente método *get*.

No exemplo anterior recuperamos o título de cada unha das películas da base de datos con

```
mysqlResult.getString(2)
```

porque sabemos que está na segunda columna e queremos tratalo como un String, mentres que o ano da película recupérase con

```
mysqlResult.getInt("ano")
```

porque sabemos que ese é o nome da columna na base de datos e queremos tratalo no noso programa como un número enteiro.

Quizais sexa recomendable a utilización da posición da columna, porque desa forma o noso programa quedará illado dos posibles problemas que causarían os cambios no nome das columnas. De tódolos xeitos haberá que seguir tendo en conta os posibles cambios no número de columnas ou na súa posición.

Outros tipos de ResultSet

Neste primeiro exemplo acabamos de ver a versión por defecto dos ResultSet, un conxunto de información que se percorre no sentido que vai dende a primeira á última fila e no que só se fan operacións de lectura, pero estes obxectos teñen outras posibilidades. Coñecelas todas escaparíase do alcance desta actividade pero incluímos a continuación un exemplo no que usaremos un ResultSet de características diferentes.

```
Statement mysqlSelect =
mysqlCon.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);
```

O método createStatement, que no exemplo anterior invocamos sen argumentos para conseguir as características por defecto, aceptará indicacións en forma de números enteiros para determinar exactamente o comportamento requirido. Neste caso estamos facendo que se trate dun conxunto de resultados que permite moverse adiante e atrás polas filas individuais (SCROLL) aínda que sen ter en conta os posibles cambios nos datos dende o momento en que foron recuperados (INSENSITIVE) e que non acepta operacións de cambio dos datos que contén (READ_ONLY).

Para ilustrar o uso dun ResultSet así, crearemos un programa no que despois de recuperar os datos dunha táboa con información sobre películas permitiremos escoller se o percorrido se fai dende o primeiro ao último dato ou á inversa. Ademais incluiremos esa elección dentro dun

bucle de xeito que se poda repetir o percorrido dos datos cantas veces se queira en calquera dos dous sentidos.

```
package films;

import java.sql.*;
import java.util.Scanner;

public class Select {
   static final String MYSQLUSER = "root";
   static final String MYSQLPASS = "root";

   public static void main(String args[]) {

      String mysqlUrl = "jdbc:mysql://localhost/peliculas";
      Connection mysqlCon = null;
      try {
      String driver = "com.mysql.jdbc.Driver";
      Class.forName(driver).newInstance( );
      }
      catch( Exception e ) {
      System.out.println("Failed to load MySQL driver.");
      return;
      }

      String input = null;
      Scanner entrada = new Scanner(System.in);
      System.out.println("Orde alfabética dos títulos ascendente (asc) ou
descendente (desc)" );
      input = entrada.nextLine();

      try {
      mysqlCon = DriverManager.getConnection(mysqlUrl, MYSQLUSER,
MYSQLPASS);

      Statement mysqlSelect =
mysqlCon.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);

      ResultSet mysqlResult = mysqlSelect.executeQuery("SELECT * FROM films
ORDER BY titulo ASC");
```

```
System.out.println("Listado de películas");

   while(!input.equals("sair")) {
      int contador = 0;
      if (input.equals("desc")) {
      mysqlResult.last();
      while(!mysqlResult.isBeforeFirst()) { // procesa cada fila
            System.out.print(++contador + ": ");
            System.out.print(mysqlResult.getInt(1) + " - ");
            System.out.print(mysqlResult.getString(2) + " - ");
            System.out.println(mysqlResult.getInt(3));
            mysqlResult.previous();
      }
      } else if (input.equals("asc")) {
           while(mysqlResult.next()) { // procesa cada fila da táboa
            System.out.print(++contador + ": ");
            System.out.print(mysqlResult.getInt(1) + " - ");
            System.out.print(mysqlResult.getString(2) + " - ");
            System.out.println(mysqlResult.getInt(3));
      }
      }
      input = entrada.nextLine();
}

}
catch( SQLException e ) {
while( e != null ) { //bucle que trata a cadea de excepcións
      System.err.println("SQLState: " + e.getSQLState( ));
      System.err.println(" Code: " + e.getErrorCode( ));
      System.err.println(" Message:");
      System.err.println(e.getMessage( ));
      e = e.getNextException( );
}
}
finally {
if( mysqlCon != null ) {
      try { mysqlCon.close( ); }
      catch( Exception e ) { e.printStackTrace( ); }
}
}
```

```
    }
}
```

A principal diferenza entre o percorrido cara adiante e cara atrás será a condición que mantén o bucle en execución. Cando o percorrido e cara adiante, facer sucesivas chamadas ao método next() será dabondo para detectar cando se acabaron os datos, cando no sexa posible dar un novo salto.

Pero cando o percorrido é cara atrás aparece o problema de que despois de chegar ao primeiro dato aínda é posible dar un novo salto, porque existe un espazo antes da primeira fila onde se sitúa inicialmente o punteiro de situación. Por iso a condición de continuación do bucle deberá ser que non nos atopemos aínda nesa posición anterior ao primeiro dato.

```
while ( !mysqlResult.isBeforeFirst( ) )
```

Exemplo empregando obxectos

```
package films;
public class Film {
   private int id;
   private String titulo;
   private int ano;

   public Film() {
   }
   public Film(int id, String titulo, int ano) {
      this.id = id;
      this.titulo = titulo;
      this.ano = ano;
   }
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public String getTitulo() {
      return titulo;
   }
   public void setTitulo(String titulo) {
      this.titulo = titulo;
   }
   public int getAno() {
      return ano;
   }
   public void setAno(int ano) {
      this.ano = ano;
   }
   public String toString() {
      return this.titulo + " (" + this.ano + ")";
   }
}
```

O que estamos conseguindo neste exemplo é que unha vez recuperados os datos do SXBDR, e xa aproveitadas as avantaxes que proporciona un sistema así, agora será a linguaxe orientada a obxectos a que entre en funcionamento coas súas particularidades e fortalezas.

Neste caso, a transformación que se fai é mínima, porque cada fila da táboa da base de datos peliculas converterase nun novo obxecto Film e para cada columna terase un atributo no obxecto dun tipo moi semellante, porque só estamos a utilizar número enteiros e cadeas de caracteres.

```
Film novoFilm = new Film(mysqlResult.getInt(1), mysqlResult.getString(2),
mysqlResult.getInt(3));
```

Despois cada un dese obxectos incorpórase a un ArrayList

```
peliculas.add(novoFilm);
```

de xeito que agora, unha vez incorporados os datos en forma de obxectos, poderemos utilizar as ferramentas e modos de traballo propios da linguaxe.

```
for(Film f : peliculas) {
   System.out.println(f.toString());
}
```

Tarefa

Debes escoller unha entidade da túa base de datos, que teña como mínimo 4 campos, referiblemente de diferentes tipos (númericos, alfanuméricos...).

Debes crear un JFrame, que empregarás tanto para engadir datos da túa entidade como para amosalos.

Comeza por engadir unha row na tua base de datos.

Cando remates, fai o inverso. Amosa unha row da base de datos, a partir da sua clave introducida polo usuario. Por exemplo, se queremos mostrar unha moto, o usuario introducirá a sua matrícula e pulsará enter ou un botón de procura.

Pensa nun deseño axeitado para os usuarios.