Java DataBase Conectivity

Marciel de Liz Santos

Java DataBase Conectivity

- Classes para gerenciamento disponíveis no pacote java.sql;
- Interface padronizada para acesso a banco de dados;
- Cada fabricante de banco de dados implementa esta interface

Interfaces JDBC

- Connection – serve para estabelecer uma conexao (sessão) com o banco de dados;
- Statement – Meio para executar queries SQL;
- ResultSet – Classe que armazena o conjunto de dados de uma seleção

Principais Classes JDBC

- DriverManager – Manipulador de drivers para conexão;
  - Em especial o método estático getConnection() retorna um objeto do tipo Connection;
- SQLException – exceções que são geradas por diversos métodos da API JDBC

Drivers para acesso a BD

- Driver ODBC, nativo do J2SE
  - sun.jdbc.odbc.JdbcOdbcDriver
- Drivers não nativos precisam ser importados como biblioteca primeiro e logo após fazer a carga do driver que é o caso do MySQL
  - com.mysql.jdbc.Driver

Efetuando uma Conexao com BD

- Seqüência para efetuar uma conexão:
  - Carregar o Driver
  - Criar uma conexão
- Exemplo ODBC:
  - Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  - Connection conexao = DriverManager.getConnection("jdbc:odbc:pessoas“,[usuario],[senha]);

Selecionando Dados usando uma Conexão

- Seqüência para efetuar uma seleção:
  - Criar um Statement usando a conexão
  - Criar um ResultSet usando um statement
- Exemplo usando uma conexão:
  - Statement stmt = conexao.createStatement();
  - ResultSet rs = stmt.executeQuery("SELECT * FROM PESSOA");

Interface Connection

- Estabelece uma sessão com o SGBD
- Criados usando o DriverManager.getConnection()
- Tem o comportamento AutoCommit por default
  - Pode ser alterado pelo método setAutoCommit(bool)

Interface Connection

- Principais métodos:
  - public Statement createStatement() throws SQLException
    - Cria um statement para executar comandos SQL
  - public void setAutoCommit(bool) throws SQLException
    - Segue ou não os comandos SQL por commit;
  - public void commit() throws SQLException
    - Valida a transação (aplica comandos SQL)
  - public void roolback() throws SQLException
    - Invalida a transação (não aplica os comandos SQL)
  - punlic void close() throws SQLException
    - Fecha a conexão com o banco

Interface Statement

- Representação de Comandos SQL
- São criados pelo objeto de conexão através do método createStatement()

Principais Métodos

- Principais Métodos
  - public ResultSet executeQuery(String sql) throws SQLException
    - Executa o sql passado como parâmetro e retornado sob forma de um resultset
    - Usado para executar SELECTs
  - public int executeUpdate(String sql) throws SQLException
    - Executa o comando sql informado e retorna o número de linhas afetadas (alteradas / criadas)
    - Usado para comandos INSERT/UPDATE/DELETE
  - public void Close() throws SQLException

Interagindo com um ResultSet

- São criados através de um objeto statement
- Representação do resultado de uma seleção em um Objeto
- Ao criar o cursor está apontando para a posição antes da primeira linha

Interagindo com um ResultSet

- Principais Métodos
  - public boolean Next() throws SQLException
    - Move para o próximo registro, se este existir retorna verdadeira senão falso
  - public [Tipo] get[Tipo](String columnName) throws SQLException
    - Retorna o valor da coluna de nome passado por parâmetro
    - [Tipo] é o tipo de retorno da função
  - public boolean wasNull() throws SQLException
    - Retorna true se o último método get[Tipo]() retornou uma coluna com valor SQL null
  - public void close() throws SQLException

Método get[Tipo]()

	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	FLOAT	DOUBLE	DECIMAL	NUMERIC	BIT	CHAR	VARCHAR	LONGVARCHAR	BINARY	VARBINARY	LONGVARBINARY	DATE	TIME	TIMESTAMP
getByte	X	x	x	x	x	x	x	x	x	x	x	x	x						
getShort	x	X	x	x	x	x	x	x	x	x	x	x	x						
getInt	x	x	X	x	x	x	x	x	x	x	x	x	x						
getLong	x	x	x	X	x	x	x	x	x	x	x	x	x						
getFloat	x	x	x	x	X	x	x	x	x	x	x	x	x						
getDouble	x	x	x	x	x	X	X	x	x	x	x	x	x						
getBigDecimal	x	x	x	x	x	x	x	X	X	x	x	x	x						
getBoolean	x	x	x	x	x	x	x	x	x	X	x	x	x						
getString	x	x	x	x	x	x	x	x	x	x	X	X	x	x	x	x	x	x	x
getBytes														X	X	x			
getDate											x	x	x				X		x
getTime											x	x	x					X	x
getTimestamp											x	x	x				x	x	X
getAsciiStream											x	x	X	x	x	x			
getUnicodeStream											x	x	X	x	x	x			
getBinaryStream														x	x	X			
getObject	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

"x" indica que o método **getXXX** pode ser usado para obter o tipo JDBC.
" X " indica que o método **getXXX** recomendado para obter o tipo JDBC.

Exemplo de Acesso

```
package não_visual;

import java.sql.*;

public class ExemploSimples {

    public static void main(String argumentos[]) {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection conn = DriverManager.getConnection("jdbc:odbc:pessoas","","");
            Statement stmt = conn.createStatement();
            ResultSet rsPessoas = stmt.executeQuery("SELECT NOME,EMAIL FROM PESSOA");
            while (rsPessoas.next()) {
                System.out.println(rsPessoas.getString(1) + "    " + rsPessoas.getString(2));
            }
        } catch (SQLException e) {
            System.out.print("Ocorreu um Erro JDBC");
        } catch (ClassNotFoundException e) {
            System.out.print("A classe informada não foi encontrada");
        } catch (Exception e) {
            System.out.print("Ocorreu uma exceção!");
        }
    }

}
```

Carregando Driver

Criando uma Connection

Criando um Statement

Manipulação do ResultSet

ResultSet de SQL