Hortolândia, 06 de novembro de 2013.

Aulas 57 e 58 – Acesso a Banco de Dados usando JDBC

1. Introdução

Na presente aula será mostrada uma das possíveis formas de **acesso a Banco de Dados** a partir da **Linguagem Java**.

Existem bancos de dados nativos para o Java, como por exemplo o Java DB. Além disso, Java consegue se comunicar com bancos de dados externos (SGBDs), tais como MySQL, PostgreeSQL, Oracle, dentre outros.

A forma com que o Java realiza a comunicação com o SGBD é através de uma biblioteca da famíla Java, denominada **JDBC** (*Java Database Connectivity*).

Particularmente, será criada nesta aula uma aplicação de exemplo em Java, que se comunicará com o SGBD MySQL.

2. Pré-Requisitos

Para a execução das tarefas descritas nesta aula, sugere-se o uso do NetBeans IDE versão 6.9+.

Os softwares que precisam estar previamente instalados na máquina são:

- Java Development Kit (JDK) versão 6+;
- MySQL Community Server;
- Conector JDBC (MySQL Connector).

3. Driver JDBC: "Conector J"

O **Connector/J** é um driver **JDBC** (*Java Database Connectivity*) que permite a aplicações escritas em Java conectarem-se a uma base de dados residente em um SGBD MySQL.

Ele é um tipo particular de driver JDBC, e serve especificamente para realizar a conexão entre o **Java** e o **MySQL**. Outros drivers JDBC são empregados para a conexão com outros SGBDs, como mostrado na **Figura 1**.

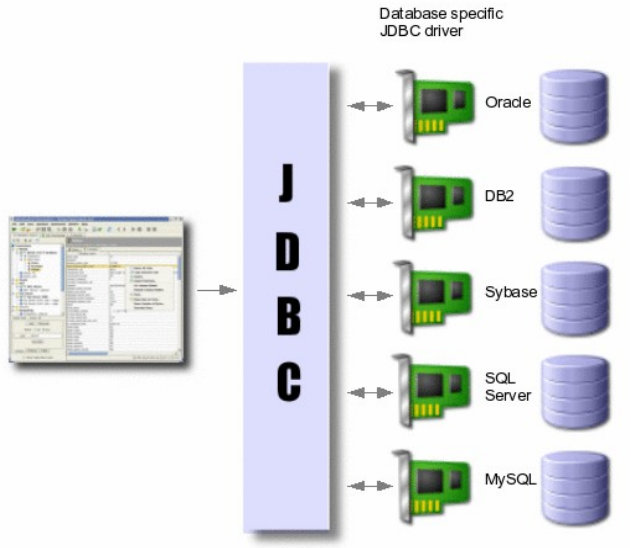

Figura 1: Drivers JDBC (Java Database Connectivity).

3.1. Instalando o Conector J

a) Para a instalação do **Conector J**, precisa-se inicialmente baixá-lo da página do MySQL:

http://www.mysql.com/products/connector/

Estando-se no ambiente Windows, pode-se escolher o pacote "**mysql-connector-java-5.1.17.zip**".

Já no ambiente Linux, a escolha pode ser "**mysql-connector-java-5.1.17.tar.gz**". Os conteúdos dos pacotes, entretanto, são os mesmos, pois o conector J nada mais é que uma coleção de classes em Java, agrupadas no formato de um arquivo jar (ou seja, é independente de plataforma).

b) Após baixar o arquivo zipado, deve-se extraí-lo em algum lugar de fácil localização. Em seguida, abrir o diretório onde o mesmo foi extraído e localizar o arquivo:

mysql-connector-java-5.1.17-bin.jar

c) Este arquivo precisa ser copiado para o diretório em que se encontra a instalação do Java, dentro do diretório "**lib**".

No Windows, o caminho para este diretório será semelhante a:

"**C:\Arquivos de Programas\ Java\ jdk1.6.0_20\lib**"

d) Em seguida, deve-se adicionar o caminho para o arquivo do conector J à variável de ambiente CLASSPATH do computador. Isto pode ser feito por meio das opções (no Windows):

Painel de Controle → Sistema → Opções Avançadas → Variáveis de Ambiente

Dentro das variáveis de ambiente, procurar pela variável de sistema "CLASSPATH". Se a mesma não existir, criá-la, e alterar o seu valor para conter a sequencia:

.;C:\Arquivos de Programas\ Java\ jdk1.6.0_20\lib\mysql-connector-java-5.1.17-bin.jar

Seguindo-se estes passos, programas em Java estarão aptos a se conectarem com o SGBD MySQL.

A Seção 5 a seguir descreve um exemplo de configuração para o acesso ao MySQL a partir de código Java.

4. Preparando a Base de Dados no MySQL

A seguir descrevem-se as etapas para a criação de uma nova base de dados no Banco MySQL.

I) Acessar algum cliente do mysql.

Por exemplo, utilizando o prompt de comandos, acessar o MySQL com o comando:

mysql -u root -p

Em seguida, fornecer a senha do usuário ***root***.

II) Criar uma nova base de dados, denominada "teste", com a instrução:

```
create database teste;
```

III) Selecionar a base de dados "teste" para utilização.

```
use teste;
```

IV) Dentro da base de dados "teste", criar uma nova tabela, chamada "cliente".

```
create table cliente(

codigo INT AUTO_INCREMENT PRIMARY KEY,

nome VARCHAR(40) NOT NULL,

endereco VARCHAR(70),

rg VARCHAR(20),

telefone VARCHAR(20),

email VARCHAR(40)

);
```

V) Verificar se a estrutura da tabela ficou correta.

```
desc cliente;
```

VI) Preencher a tabela "**cliente**" com algumas entradas, correspondentes a diferentes clientes.

```
insert into cliente values(null, 'Lucas Alexandre', 'Avenida
Brasil, 1200', '3878675678', '(19) 3398-7667',
lucas.alexandre@gmail.com);

insert into cliente values(null, 'Adriana de Almeida', 'Rua
Indiana, 730', '9329338220', '(19) 3332-3987',
adriana.almeida@gmail.com);

insert into cliente values(null, 'Ademir Carlos da Silva',
'Avenida Lima, 250', '7390875678', '(19) 3398-9293',
ademir.carlos@gmail.com);
```

VII) Verificar se as entradas foram inseridas adequadamente no Banco de Dados.

```
select * from cliente;
```

4.1. Classe Java de Acesso ao Banco de Dados

Uma vez tendo sido criadas a base de dados "**teste**" e a tabela "**cliente**", pode-se proceder à próxima etapa, que é a de acessar esta base de dados a partir de código escrito em Java.

A Listagem 1 apresenta uma possível forma de se conectar à Base de Dados do MySQL, através da classe "**AcessoBaseDados.java**".

```
/*********************************************************************
Arquivo: AcessoBaseDados.java
Descricao: Classe simples de acesso a uma base de dados do banco mysql.
Consulta dos dados de uma tabela chamada "cliente".
*********************************************************************/
import java.sql.*;
public class AcessoBaseDados{
 private static String pedidoSQL;
 private static String codigo;
 private static String nome;
 private static String endereco;
 public static void main(String[] args){
        System.out.println("AcessoBaseDados> Testando acesso...");
        Connection conn = null;
        pedidoSQL = "SELECT codigo,nome,endereco FROM cliente;";
        try{
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection("jdbc:mysql://localhost/teste?
user=root&password=root");
        System.out.println("Conexao bem sucedida.");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(pedidoSQL);
        while(rs.next()){
               codigo = rs.getString("codigo");
               nome = rs.getString("nome");
               sobrenome = rs.getString("endereco");
               System.out.println("Codigo: " + codigo);
               System.out.println("Nome: " + nome);
               System.out.println("Endereco: " + endereco);
               }
               System.out.println("Consulta realizada com sucesso.");
        }
        catch(ClassNotFoundException cnfe){
               System.out.println("Excessao de classe nao encontrada.");
        }
        catch(SQLException sqle){
               System.out.println("Excessao de conexao ao banco.");
        }
        finally{
               System.out.println("Encerrando a conexao com o banco...");
```

```
      try{
                              conn.close();
                              System.out.println("Conexao com o banco
encerrada.");
                        }
                        catch(SQLException sqle){
                              System.out.println("Excessao de conexao ao banco.");
                        }
                  }
            }
      }
```

Listagem 1: Conexão com o banco de dados MySQL.

5. Referências Bibliográficas

BORATTI, Isaias Camilo. **Programação Orientada a Objetos usando Delphi.** Quarta Edição. Editora Visual Books. Florianópolis, 2007.

DEITEL, H.M., DEITEL, P.J. **Java – Como programar**. Terceira edição. Porto Alegre: Bookman Editora, 2001.

HORSTMAN, C. ***Conceitos de Computação com o Essencial de Java***. 3ª Edição. Porto Alegre: Bookman, 2003.

ORACLE. ***API da classe ArrayList***. URL: http://docs.oracle.com/javase/6/docs/api/java/util/ArrayList.html. Última Consulta: 06/11/2013.

SANTOS, Rafael. **Introdução à Programação orientada a objetos usando Java**. 8ª Reimpressão. Rio de Janeiro: Campus - Elsevier, 2003.