Utilizando Mysql com Java

Leonardo Cabral da Rocha Soares

30 de maio de 2018

Conteúdo

Introdução

Porque utilizar banco de dados?

A maioria dos sistemas precisam manter as informações com as quais trabalham para permitir consultas futuras, geração de relatórios ou possíveis alterações nas informações.
Embora seja possível salvar estas informações como arquivos de texto, o uso de banco de dados possibilita a terceirização de diversos problemas que teriam de ser tratados como integridade e redundância.

Pré-requisitos

O que você precisa saber

- Criar banco de dados;
- Criar tabelas e consultas;
- Conhecer a linguagem SQL.

Pré-requisitos

O que você precisa saber

- Criar banco de dados;
- Criar tabelas e consultas;
- Conhecer a linguagem SQL.

Pré-requisitos

O que você precisa saber

- Criar banco de dados;
- Criar tabelas e consultas;
- Conhecer a linguagem SQL.

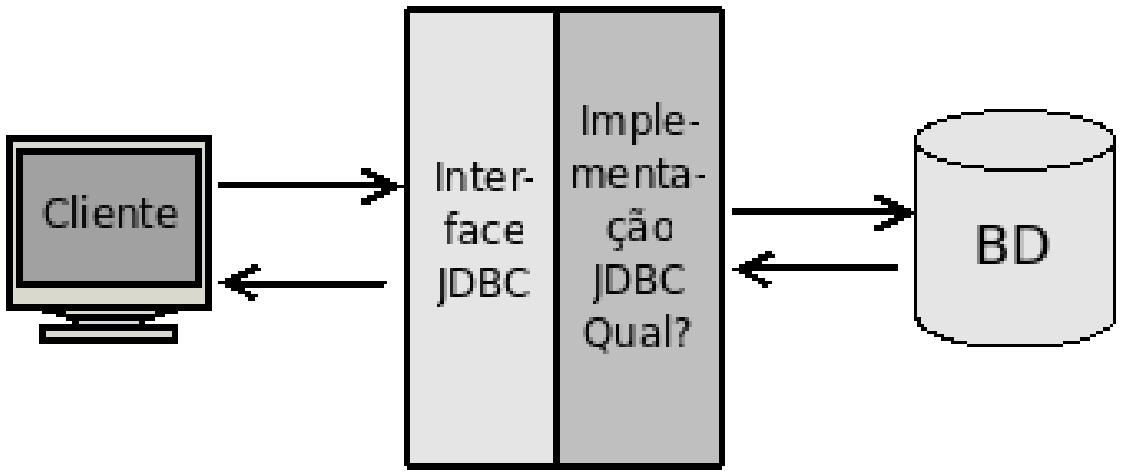
Desenvolvimento

Como é feito a conexão?

Para evitar que tenhamos que escrever um código de comunicação específico para cada banco de dados, o Java disponibiliza um conjunto de interfaces que devem ser implementadas. Essas interfaces ficam dentro do pacote **java.sql** e nos referiremos a elas como **JDBC** (Java Database Connectivity).

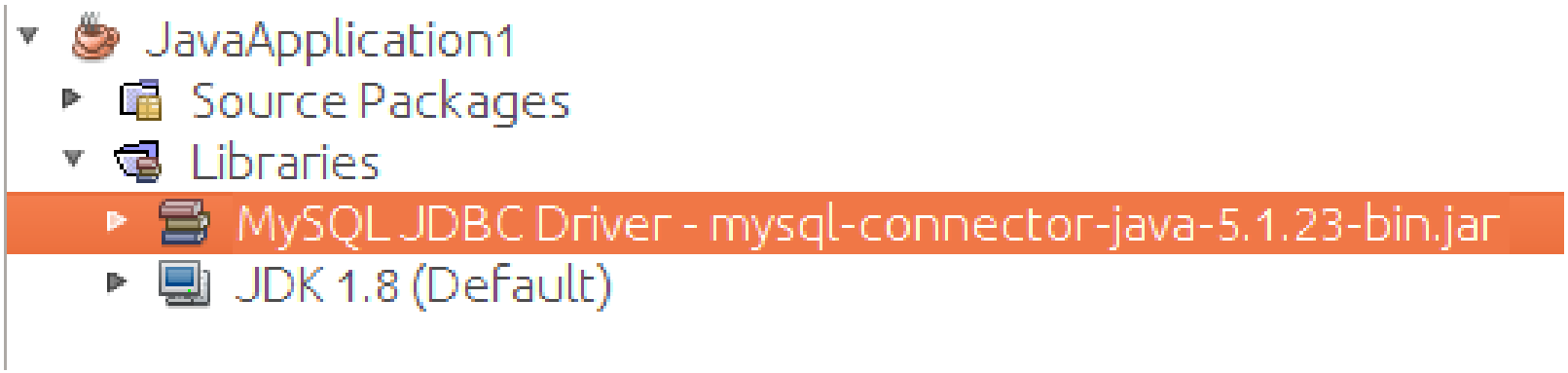
Driver de conexão

Um conjunto de classes concretas que implementa a JDBC é chamado de **driver**. Todos os principais bancos de dados possuem Drivers JDBC.



Adicione o driver ao projeto

Como utilizaremos o **MySQL** você deve adicionar ao seu projeto a *library* **MySQL JDBC Driver**, disponível no NetBeans. Se for utilizar outro banco de dados, faça o download do conector a adicione-o ao seu projeto.



Desenvolvimento

DriverManager

A classe *DriverManager* é responsável por conversar com todos os drivers disponíveis para utilização. Entre seus métodos, está o método estático *getConnection(String)*. Este método recebe como parâmetro uma string de conexão JDBC que indica qual banco de dados iremos utilizar.

String de conexão MySQL

jdbc:mysql://ip_servidor/nome_banco

Desenvolvimento

Código exemplo

```
public class JDBCExemplo {
    public static void main(String[] args) throws SQLException {
        Connection conexao = DriverManager.getConnection(
                    "jdbc:mysql://localhost/fj21");
        System.out.println("Conectado!");
        conexao.close();
    }
}
```

Desenvolvimento

Principais interfaces JDBC

1. DriverManager;
2. Connection;
3. Statement;
4. PreparedStatement;
5. ResultSet.

Desenvolvimento

Principais interfaces JDBC

1. DriverManager;
2. Connection;
3. Statement;
4. PreparedStatement;
5. ResultSet.

Desenvolvimento

Principais interfaces JDBC

1. DriverManager;
2. Connection;
3. Statement;
4. PreparedStatement;
5. ResultSet.

Desenvolvimento

Principais interfaces JDBC

1. DriverManager;
2. Connection;
3. Statement;
4. PreparedStatement;
5. ResultSet.

Banco de dados

Configurações do servidor

Endereço: 192.168.50.100
Usuário: root
Senha: senhadb
Banco: sysformat

Tabela: formandos

```
mysql> desc formandos;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| nome  | varchar(200) | YES  |     | NULL    |                |
| email | varchar(200) | YES  |     | NULL    |                |
| idade | int(11)      | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
```

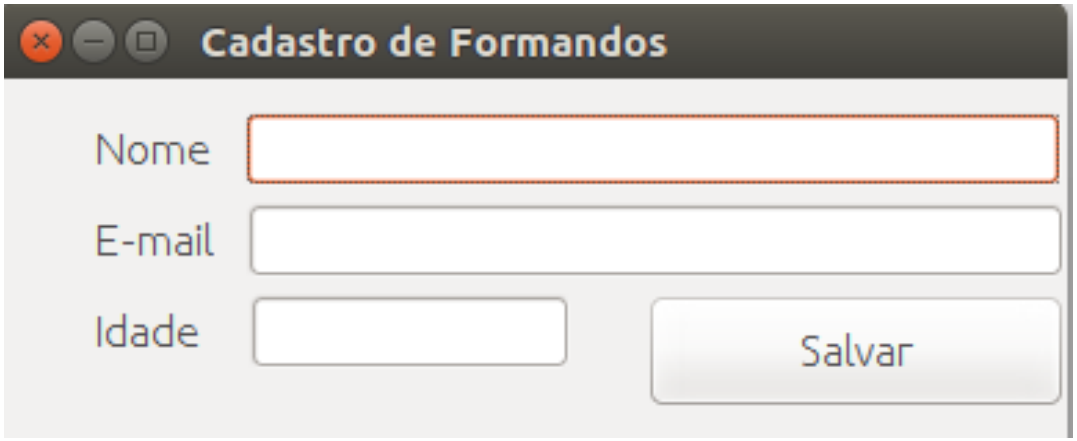
Classe Conexao

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class Conexao {
    public Connection getConexao() throws SQLException{
        return DriverManager.getConnection(
                "jdbc:mysql://192.168.50.100/sysformat",
                                    "root","senhadb");
    }
}
```

Classe Formando

```
public class Formando {
    public void salvar(Connection con, String nome,
            String email, int idade) throws SQLException{
        String SQL = "insert into formandos "
                + "(nome, email, idade)values (?,?,?)";
        PreparedStatement pst = con.prepareStatement(SQL);
        pst.setString(1, nome);
        pst.setString(2, email);
        pst.setInt(3, idade);
        pst.execute();
    }
}
```

Interface do usuário



Evento ActionPerformed

```
private void jbSalvarActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        Connection con = new Conexao().getConexao();
        Formando formando = new Formando();
        String nome = jtNome.getText();
        String email = jtEmail.getText();
        int idade = Integer.parseInt(jtIdade.getText());
        formando.salvar(con, nome, email, idade);
        JOptionPane.showMessageDialog(this,
                                      "Formando cadastrado com sucesso.");
        jtNome.setText(null);
        jtEmail.setText(null);
        jtIdade.setText(null);
        jtNome.requestFocus();
    } catch(SQLException sq){
        JOptionPane.showMessageDialog(this,
                                      "Ocorreu um erro: "+sq.getMessage());
    }
}
```