UNIP
UNIVERSIDADE PAULISTA

Aplicações de Linguagem de Programação Orientada a Objetos

MÓDULO 5

Professor: Me. Pablo I. Gandulfo
Data: 01/08/2020
Versão: 1.1

Professor: Me. Pablo I. Gandulfo
Data: 01/08/2020
Versão: 1.1

JDBC – java.sql - Conexão com Banco de Dados

- JDBC
- Tipos de Drivers
- Exemplo de Aplicação JDBC Simples
- Banco de Dados MySQL
- Banco de Dados HyperSQL (Java puro)
- Exercícios

JDBC

- Para que seja possível conversar com um Banco de Dados a partir do JAVA, foi desenvolvida uma API nativa de acesso chamada JDBC - Java Database Connectivity

- A API JDBC é uma interface padronizada de acesso a dados de diversos bancos de dados relacionais

- O JDBC permite:
  - Conectar com um banco de dados
  - Executar comandos SQL
  - Alterar e consultar metadados e dados

Driver JDBC

- Para conversar com um (ou mais) Bancos de Dados, é necessário utilizar um *driver*

- *Driver* JDBC é uma coleção de classes implementando as classes e interfaces da API JDBC, contendo, em especial, uma classe que implementa a interface java.sql.Driver

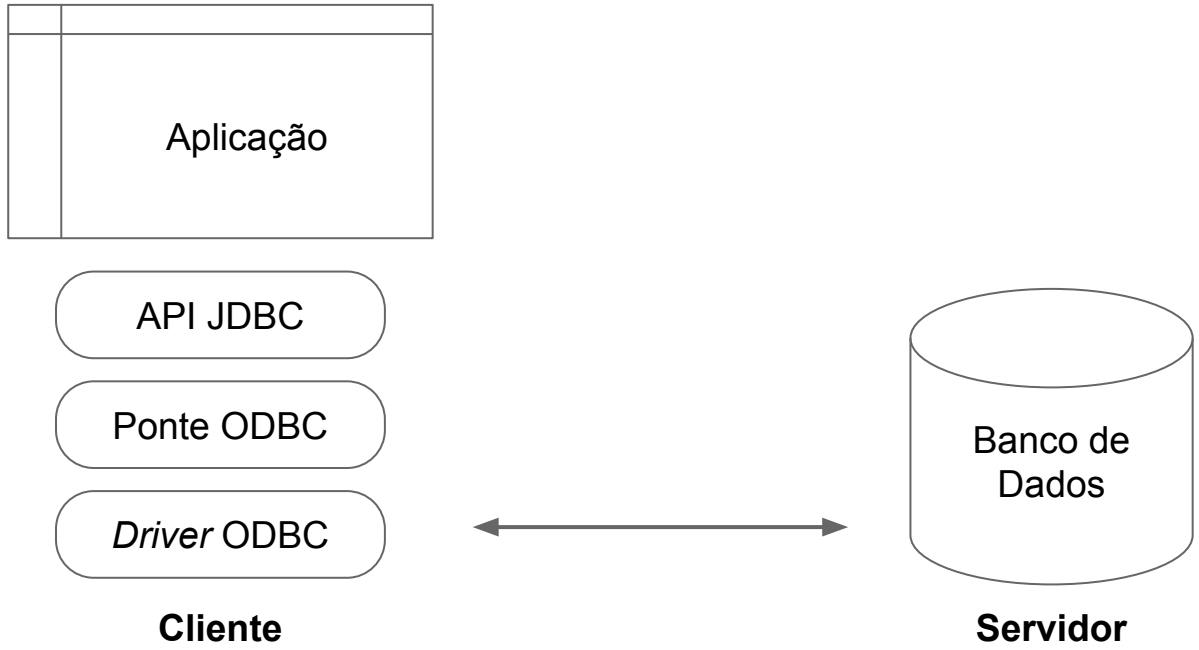
Tipos de *Drivers*

- Existem basicamente 4 tipos de *drivers* JDBC:
  1. Ponte JDBC-ODBC
  2. Baseado em API Nativa
  3. JDBC-Net
  4. Protocolo Nativo

Driver JDBC - Ponte JDBC-ODBC

- Usa tecnologia existente
- Fornece acesso ao Banco de Dados através de um *driver* ODBC
- Requer que o *driver* ODBC esteja instalado no cliente
- É mais apropriado quando a aplicação rodará em poucos locais (exigindo, portanto, menos instalações nos clientes)
- Possui vários pontos em potencial suscetíveis a falhas

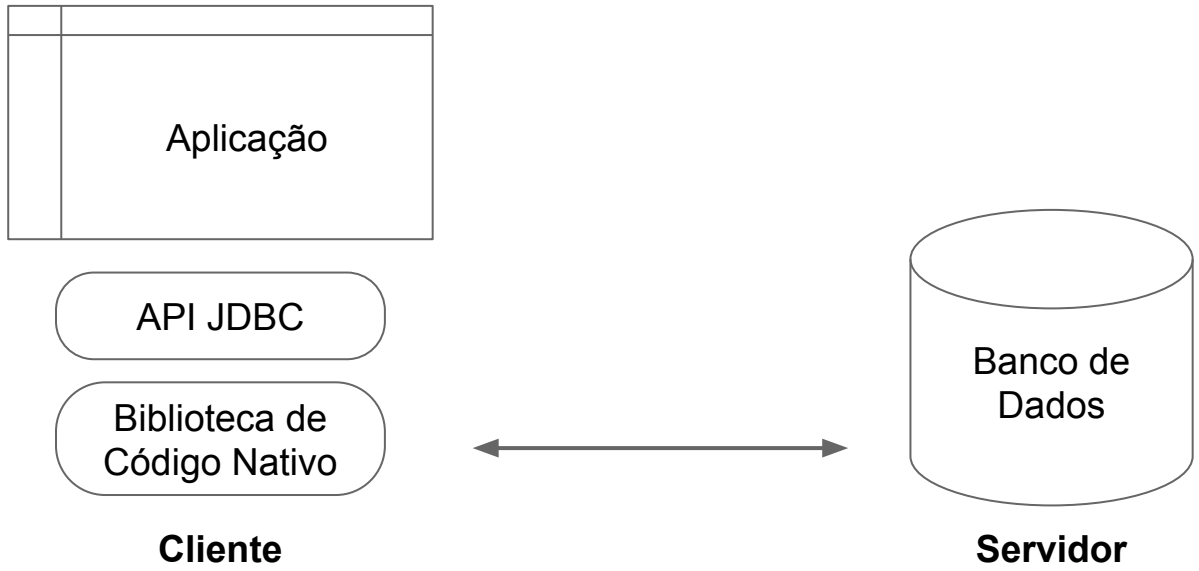
Driver JDBC - Ponte JDBC-ODBC (cont.)



Driver JDBC – Baseado em API Nativa

- Construído a partir de uma biblioteca cliente nativa do Banco de Dados
- O *driver* traduz chamadas JDBC em chamadas da API da biblioteca cliente nativa
- A biblioteca é distribuída pelo fabricante do Sistema de Gerenciamento de Banco de Dados (SGBD)
- Não cumpre o objetivo da API JDBC quanto à portabilidade entre plataformas

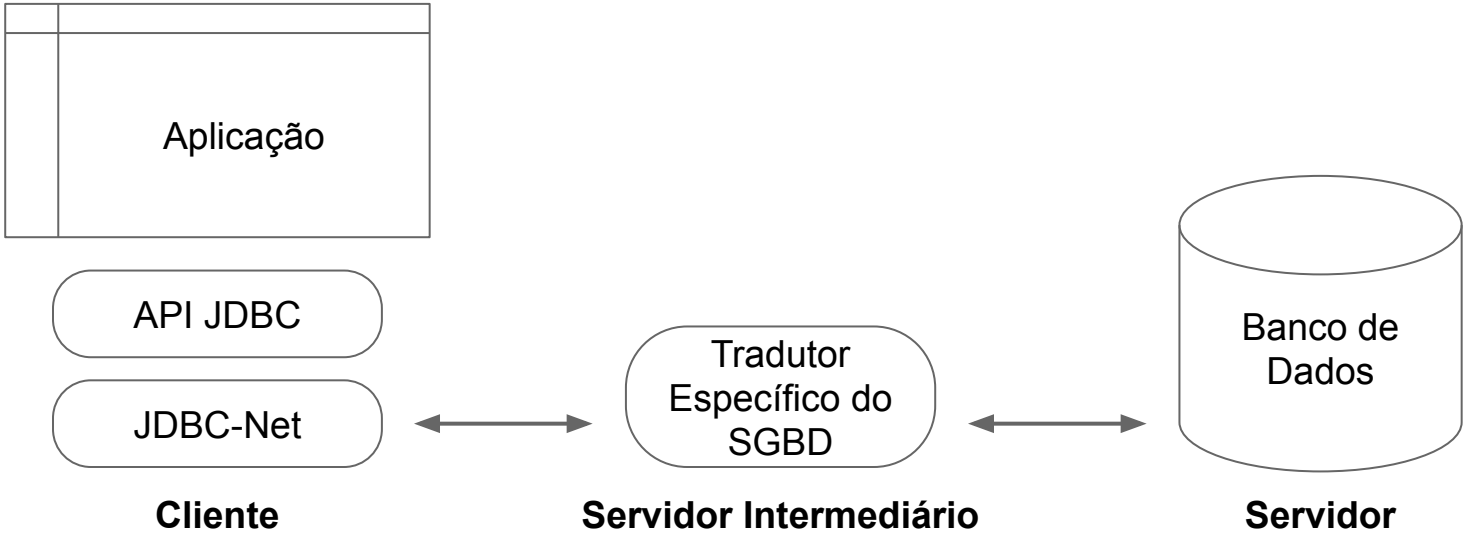
Driver JDBC - Baseado em API Nativa (cont.)



Driver JDBC - JDBC-Net

- Comunica com um servidor intermediário situado entre o cliente e o Banco de Dados
- Usa um protocolo de rede específico do intermediário
- Nenhum código nativo precisa ser instalado nas máquinas clientes do Banco de Dados
- Apresenta a maior flexibilidade

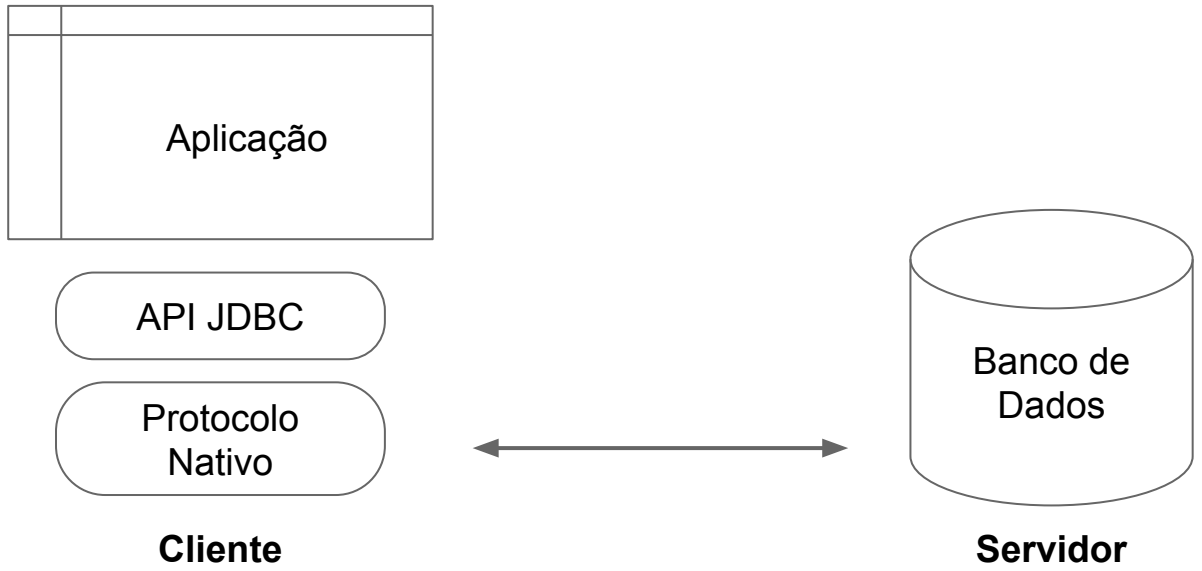
Driver JDBC – JDBC-Net (cont.)



Driver JDBC – Protocolo Nativo

- Converte chamadas JDBC diretamente no protocolo de rede utilizado pelo SGBD
- Permite chamadas diretas do cliente ao Banco de Dados
- Apresenta considerável desempenho, mas sem flexibilidade para uma possível troca de SGBD

Driver JDBC – Protocolo Nativo (cont.)



Classes e Interfaces Relacionadas

- DriverManager: gerencia *drivers* de Banco de Dados e permite criar novas conexões ao Banco de Dados
- Driver: representa uma implementação JDBC específica
- Conexão: é uma sessão com o Banco de Dados específico
- Statement, PreparedStatement, CallableStatement: executam comandos SQL numa conexão
- ResultSet: provê acesso aos dados de uma tabela

Identificação de um Banco de Dados Usando uma URL

- Um localizador uniforme de recursos (URL) JDBC:
  - Identifica um Banco de Dados para que o *driver* correto possa reconhecer e estabelecer uma conexão
  - É determinado pelo fabricante do SGBD
  - Permite um nível de indireção
- Sintaxe completa da URL para acesso local:
  - Jdbc:[protocolo]:[nome]
    - Exemplo: jdbc:odbc:demodb

Identificação de um Banco de Dados Usando uma URL (cont.)

- Sintaxe completa da URL para acesso remoto:
  - jdbc:[protocolo]://[servidor]:[porta]/[nome]
    - Exemplo: jdbc:mysql://dbserver:3306/test

- Campos da URL:
  - **protocolo**: o nome do *driver* ou o tipo do mecanismo de conectividade com o Banco de Dados (ex: odbc)
  - **nome**: identifica a base de dados
  - **servidor** e **porta**: endereço na rede do Banco de Dados (local ou remoto)

Características e Técnicas

- Criar uma aplicação JDBC básica envolve os seguintes passos:
  - Passo 1: Registrar um *driver* JDBC
  - Passo 2: Estabelecer uma conexão ao Banco de Dados
  - Passo 3: Criar um *statement*
  - Passo 4: Executar o comando SQL
  - Passo 5: Processar os resultados
  - Passo 6: Fechar objetos JDBC

Exemplo de Aplicação JDBC Simples

```
CREATE SCHEMA test; -- Específico do MYSQL
USE test;                    -- Específico do MYSQL

DROP TABLE funcionario;

CREATE TABLE funcionario (
  matricula      INTEGER NOT NULL,
  nome           VARCHAR(80) NOT NULL,
  cargo          VARCHAR(45) NOT NULL,
  PRIMARY KEY (matricula)
);

INSERT INTO funcionario (matricula, nome, cargo)
VALUES (1, 'Carlos', 'Gerente de Operações');

INSERT INTO funcionario (matricula, nome, cargo)
VALUES (2, 'Raissa', 'Gerente de Telemarketing');
```

Exemplo de Aplicação JDBC Simples - Banco de Dados MySQL

```
import java.sql.Connection; import java.sql.DriverManager;
import java.sql.PreparedStatement; import java.sql.ResultSet;
import java.sql.SQLException;

public static void main(String[] args) throws ClassNotFoundException, SQLException {
    Class.forName("com.mysql.cj.jdbc.Driver");       // Passo 1 para o MySQL 8
    //Class.forName("com.mysql.jdbc.Driver");        // Passo 1 para o MySQL 5
    Connection conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?useSSL=false&serverTimezone=UTC", "root", "admin");          // Passo 2
    PreparedStatement stmt = conn.prepareStatement("SELECT * FROM funcionario");   // Passo 3
    ResultSet rs = stmt.executeQuery();        // Passo 4
    while (rs.next()) {   // Passo 5
        System.out.println(rs.getInt("matricula") + " - "
            + rs.getString("nome") + " - " + rs.getString("cargo"));
    }
    rs.close();   // Passo 6 ...
    stmt.close(); conn.close();
}
```

Saída

1 - Carlos - Gerente de Operações
2 - Raissa - Gerente de Telemarketing

Exemplo de Aplicação JDBC Simples - Banco de Dados HyperSQL (HSQL)

```
import java.sql.Connection; import java.sql.DriverManager;
import java.sql.PreparedStatement; import java.sql.ResultSet;
import java.sql.SQLException;

public static void main(String[] args) throws ClassNotFoundException, SQLException {
    Class.forName("org.hsqldb.jdbcDriver");// Passo 1
    Connection conn = DriverManager.getConnection("jdbc:hsqldb:file:test", "sa", "");    // Passo 2
    PreparedStatement stmt = conn.prepareStatement("SELECT * FROM funcionario");  // Passo 3
    ResultSet rs = stmt.executeQuery();        // Passo 4
    while (rs.next()) {   // Passo 5
        System.out.println(rs.getInt("matricula") + " - "
            + rs.getString("nome") + " - " + rs.getString("cargo"));
    }
    rs.close();   // Passo 6 ...
    stmt.close();
    conn.close();
}
```

Saída

1 - Carlos - Gerente de Operações
2 - Raissa - Gerente de Telemarketing

Exercício 1

- Desenvolva um programa (com interface gráfica AWT / Swing ou não) com as seguintes características:
  - Conecte num BD qualquer
  - Execute um comando SQL INSERT para incluir um registro na tabela
  - Execute um comando SQL SELECT para ler os dados do registro incluído
  - Imprima mensagens informando o usuário de cada passo