UNIVERSIDADE PITÁGORAS UNOPAR

Engenharia de Software
Sistemas Operacionais

AUTOR: ALLAS MAYCON DO VALLE
RA: 3649457105

Programação Orientada a Objetos II

Conectando Java e MySQL com JDBC

Cidade de Serrana, SP
2025

AUTOR: ALLAS MAYCON DO VALLE
RA: 3649457105

Programação Orientada a Objetos II

Conectando Java e MySQL com JDBC

Neste projeto marca o início da nossa jornada em programação de banco de dados usando Java. O objetivo principal é fazer com que o código se comunique com um banco de dados MySQL usando a tecnologia padrão do Java, chamada JDBC(Java Database Connectivity).
Vou criar um pequeno programa para demonstrar esta conexão. Preciso garantir que o driver MySQL Connector/J esteja no nosso projeto. Depois, configurar a URL, o usuário e a senha. O código será focado em quatro passos essenciais do JDBC: **1)** carregar o driver (implícito na versão moderna), **2)** estabelecer a conexão, **3)** executar uma consulta **SELECT** para buscar dados de uma tabela de exemplo e **4)** exibir os resultados no console. O sucesso do projeto será comprovado ao listar corretamente os dados dos usuários da tabela.
Orientador:
Tutor à Distância: Frederico Aparecido Faedo Pinto.
Prof. Renan Cleverson Laureano Flor da Rosa.

Cidade de Serrana, SP
2025

SUMÁRIO

1 INTRODUÇÃO

Este quarto projeto é um marco na nossa evolução como desenvolvedores, pois nos tira do mundo puramente Java e nos conecta a um elemento externo e crucial: o **Banco de Dados**. O foco aqui é estabelecer a comunicação entre nossa aplicação Java e um servidor **MySQL**, usando a tecnologia padrão conhecida como **JDBC** (Java Database Connectivity).

O JDBC é essencialmente uma ponte. Ele é uma API (Interface de Programação de Aplicações) que permite que comandos SQL sejam enviados a um banco de dados e que os resultados voltem para o nosso código Java, de forma padronizada.

Para começar, tivemos que configurar o ambiente. Isso incluiu dois passos fora do código Java:

- Instalação do Driver: Precisamos do **MySQL Connector/J**, que é o "tradutor" específico para o MySQL. Ele precisa estar na pasta do projeto para que o Java saiba "falar" com o banco.

- Preparação do Banco: Criamos um banco de dados e uma tabela simples (Usuarios) com alguns registros (ID, Nome, Email) para termos dados reais para buscar.

No código, o processo se resume em gerenciar o fluxo da conexão: primeiro, definir a *URL*, o *usuário* e a *senha*. Em seguida, usar a classe Connection para ligar, a classe Statement para enviar a consulta SELECT e, finalmente, a classe ResultSet para receber e navegar pelos dados retornados.

A meta do projeto é provar que a conexão funciona e exibir no console os dados dos usuários exatamente como estão na tabela, validando o poder do Java de interagir com sistemas de dados externos.

2 DESENVOLVIMENTO

Perfeito! Vamos detalhar a Seção de Desenvolvimento do Projeto 4, que envolve tanto a preparação do banco de dados quanto o código Java que realiza a conexão e a consulta.

2.1 Seção de Desenvolvimento: Implementando a Conexão JDBC

O desenvolvimento deste projeto é dividido em três etapas: **Configuração**, **SQL** e **Código Java**.

2.2 Configuração e Dependência (Requisito 1)

Antes de rodar o código, garantimos que o driver necessário está acessível:

- Driver JDBC: O arquivo mysql-connector-java-x.x.x.jar (MySQL Connector/J) deve ser adicionado como **dependência** (biblioteca externa) no projeto Java.

2.3 Preparação do Banco de Dados MySQL (Requisito 2)

Criamos o banco de dados e a tabela de exemplo (a qual chamaremos de Usuarios) com os dados que serão consultados:

```
SQL
-- Criação do Banco de Dados (se necessário)
CREATE DATABASE IF NOT EXISTS meu_projeto_db;

-- Seleciona o Banco de Dados
USE meu_projeto_db;

-- Criação da Tabela de Exemplo
CREATE TABLE IF NOT EXISTS Usuarios (
    ID INT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL
);

-- Inserção dos Dados (Requisitos do Resultado)
INSERT INTO Usuarios (ID, Nome, Email) VALUES
(1, 'João Silva', 'joao@example.com'),
(2, 'Maria Oliveira', 'maria@example.com'),
(3, 'Carlos Souza', 'carlos@example.com');
```

2.4 Código Java: Classe ConexaoMySQL (Requisitos 3, 4, 5, 6 e 7)

A classe a seguir contém o código completo para estabelecer a conexão, executar o SELECT e exibir os resultados no console, atendendo a todo o checklist JDBC.

Java

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

public class ConexaoMySQL {

    private static final String URL = "jdbc:mysql://localhost:3306/meu_projeto_db";
    private static final String USUARIO = "root"; // Altere conforme seu usuário
    private static final String SENHA = "sua_senha"; // Altere conforme sua senha

    public static void main(String[] args) {

        Connection conexao = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            System.out.println("1. Tentando conectar ao banco de dados MySQL...");
            conexao = DriverManager.getConnection(URL, USUARIO, SENHA);
            System.out.println("✅ Conexão bem-sucedida com o banco de dados MySQL!");

            stmt = conexao.createStatement();
            String sql = "SELECT ID, Nome, Email FROM Usuarios";
            System.out.println("\n2. Executando a consulta: " + sql);
            rs = stmt.executeQuery(sql);

            System.out.println("\n--- Resultados da Consulta ---");

            while (rs.next()) {
                int id = rs.getInt("ID");
                String nome = rs.getString("Nome");
                String email = rs.getString("Email");
```

```
            System.out.printf("ID: %d, Nome: %s, Email: %s%n", id, nome, email);
        }

    } catch (SQLException e) {
        System.err.println("❌ Ocorreu um erro de SQL: " + e.getMessage());
        e.printStackTrace();

    } finally {
        try {
            if (rs != null) rs.close();
            if (stmt != null) stmt.close();
            if (conexao != null) conexao.close();
            System.out.println("\n3. Conexão e recursos fechados com sucesso.");
        } catch (SQLException e) {
            System.err.println("Erro ao fechar recursos: " + e.getMessage());
        }
    }
  }
}
```

3 CONCLUSÃO

O Projeto 4 foi a culminância prática do nosso estudo de Java, levando a aplicação além de meras estruturas de dados internas para interagir com o ambiente externo, especificamente um **Banco de Dados MySQL**. A tecnologia central utilizada, o **JDBC** (Java Database Connectivity), provou ser a ferramenta essencial para essa integração.

O sucesso deste projeto é validado pela **conexão bem-sucedida** e pela **exibição correta dos dados** no console, conforme o esperado. O processo seguiu rigorosamente o checklist, destacando a importância da organização:

- Configuração: O driver **MySQL Connector/J** atuou como o tradutor necessário, permitindo que o Java "entenda" o MySQL.

- Protocolo: O código Java demonstrou o fluxo padrão de qualquer comunicação com banco de dados: estabelecer a Connection, preparar a Statement e processar o ResultSet.

- Segurança e Boas Práticas: A utilização do bloco try-catch-finally foi crucial. O bloco finally garante que os recursos (Connection, Statement, ResultSet) sejam **fechados** em todas as circunstâncias, prevenindo vazamentos de memória e erros futuros no servidor de banco de dados.

A habilidade de conectar uma aplicação Java a um banco de dados relacional é um dos conhecimentos **mais importantes** no mercado de trabalho. Ela é a base para a criação de sistemas complexos, como aplicações web e sistemas de gestão.

Com este projeto, dominamos a ponte entre a Orientação a Objetos em Java e a persistência de dados em um SGBD (Sistema Gerenciador de Banco de Dados) robusto como o MySQL. Este ciclo de aprendizado, que começou nos fundamentos de Java e passou por threads e padrões de projeto, agora se completa com a integração de dados, nos preparando para projetos mais ambiciosos.

REFERÊNCIAS

TURINI, Rodrigo. Livro - *Explorando APIs e bibliotecas Java: JDBC, IO, Threads, JavaFX e mais*. São Paulo: Casa do Código, 2015. (Este livro é uma referência moderna e prática que cobre a API JDBC em profundidade, desde a configuração inicial até a execução de consultas e o fechamento de recursos, sendo diretamente aplicável ao Projeto 4.)

DEITEL, Harvey M.; DEITEL, Paul J. Livro - *Java: Como Programar*. 10. ed. São Paulo: Pearson Education do Brasil, 2016. (Esta referência é essencial e frequentemente cobre em seus capítulos avançados a API JDBC, detalhando as classes Connection, Statement e ResultSet, pilares da nossa implementação.)

FISCHER, S. Livro - *JDBC API Tutorial and Reference*. 3. ed. Nova York: Pearson, 2003. (Embora seja uma edição mais antiga, este é um guia abrangente e detalhado da API JDBC, ideal para entender a arquitetura dos drivers, o fluxo de comunicação entre Java e SGBDs e as boas práticas de manipulação de transações, fundamentais para o sucesso do Projeto 4.)

APÊNDICE: Repositório do Código Fonte

As atividades possuem material exclusivo do portfólio que se encontra no GitHub para download.

https://github.com/allas-amk/portfolio_POO_projeto_IIII