POLO OLARIA - RIO DE JANEIRO - RJ

DESENVOLVIMENTO FULL STACK

Nivel 3: Back-End Sem Banco Não Tem | Turma 9001

Erik Bastos de Moraes

Missão Prática | Nível 3 | Mundo 3

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Objetivos da prática

1- Implementar persistência com base no middleware JDBC.

2- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.

3- Implementar o mapeamento objeto-relacional em sistemas Java.

4- Criar sistemas cadastrais com persistência em banco relacional.

5- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

Pessoa.java

```
package cadastrobd.model;

public class Pessoa {
    int id;
    String nome;
    String logradouro;
    String cidade;
    String estado;
    String telefone;
    String email;

    public Pessoa() {}

    public Pessoa(int id, String nome, String logradouro, String cidade, String estado,
String telefone, String email) {
        this.id = id;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    public void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
```

```
        System.out.println("Logradouro: " + logradouro);
        System.out.println("Cidade: " + cidade);
        System.out.println("Estado: " + estado);
        System.out.println("Telefone: " + telefone);
        System.out.println("Email: " + email);
    }
}
```

PessoaFisica.java

```
package cadastrobd.model;


public class PessoaFisica extends Pessoa {
    String cpf;


    public PessoaFisica() {}


    public PessoaFisica(int id, String nome, String logradouro, String cidade, String
estado, String telefone, String email, String cpf) {
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cpf = cpf;
    }


    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }
}
```

PessoaJuridica.java

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    String cnpj;

    public PessoaJuridica() {}

    public PessoaJuridica(int id, String nome, String logradouro, String cidade, String
estado, String telefone, String email, String cnpj) {
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cnpj = cnpj;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}
```

ConectorBD.java

```
package cadastrobd.model.util;


import java.sql.*;


public class ConectorBD {
    private static final String URL =
"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true";
    private static final String USER = "loja";
    private static final String PASSWORD = "loja";


    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }


    public static PreparedStatement getPrepared(String sql) throws SQLException {
        return getConnection().prepareStatement(sql);
    }


    public static ResultSet getSelect(String sql) throws SQLException {
        return getPrepared(sql).executeQuery();
    }


    public static void close(Connection conn) {
        try {
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
```

```
        } catch (SQLException e) {
        }
    }

    public static void close(PreparedStatement stmt) {
        try {
            if (stmt != null && !stmt.isClosed()) {
                stmt.close();
            }
        } catch (SQLException e) {
        }
    }

    public static void close(ResultSet rs) {
        try {
            if (rs != null && !rs.isClosed()) {
                rs.close();
            }
        } catch (SQLException e) {
        }
    }
}
```

SequenceManager.java

```
package cadastrobd.model.util;

import java.sql.*;

public class SequenceManager {

    public static int getValue(String sequenceName) {
        int value = 0;
        String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS nextVal";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql);
             ResultSet rs = stmt.executeQuery()) {

            if (rs.next()) {
                value = rs.getInt("nextVal");
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return value;
    }
}
```

PessoaFisicaDAO.java

```
package cadastrobd.model;


import cadastrobd.model.util.ConectorBD;

import cadastrobd.model.util.SequenceManager;

import java.sql.*;

import java.util.ArrayList;

import java.util.List;


public class PessoaFisicaDAO {


    public PessoaFisica getPessoa(int id) {

        String sql = "SELECT * FROM Pessoa INNER JOIN PessoaFisica ON Pessoa.id =
PessoaFisica.id WHERE Pessoa.id = ?";

        PessoaFisica pessoa = null;


        try (Connection conn = ConectorBD.getConnection();

             PreparedStatement stmt = conn.prepareStatement(sql)) {


            stmt.setInt(1, id);

            ResultSet rs = stmt.executeQuery();


            if (rs.next()) {

                pessoa = new PessoaFisica(
```

```
                rs.getInt("id"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cpf")
            );
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return pessoa;
}

public List<PessoaFisica> getPessoas() {
    List<PessoaFisica> pessoas = new ArrayList<>();
    String sql = "SELECT * FROM Pessoa INNER JOIN PessoaFisica ON Pessoa.id =
PessoaFisica.id";

    try (Connection conn = ConectorBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery()) {
```

```
            while (rs.next()) {
                PessoaFisica pessoa = new PessoaFisica(
                    rs.getInt("id"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
                    rs.getString("telefone"),
                    rs.getString("email"),
                    rs.getString("cpf")
                );
                pessoas.add(pessoa);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return pessoas;
    }

    public void incluir(PessoaFisica pessoa) {
        String sqlPessoa = "INSERT INTO Pessoa (id, nome, logradouro, cidade, estado,
telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        String sqlPessoaFisica = "INSERT INTO PessoaFisica (id, cpf) VALUES (?, ?)";
```

```
        int id = SequenceManager.getValue("PessoaSeq");
        pessoa.id = id;

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa);
             PreparedStatement stmtPessoaFisica =
conn.prepareStatement(sqlPessoaFisica)) {

            stmtPessoa.setInt(1, pessoa.id);
            stmtPessoa.setString(2, pessoa.nome);
            stmtPessoa.setString(3, pessoa.logradouro);
            stmtPessoa.setString(4, pessoa.cidade);
            stmtPessoa.setString(5, pessoa.estado);
            stmtPessoa.setString(6, pessoa.telefone);
            stmtPessoa.setString(7, pessoa.email);
            stmtPessoa.executeUpdate();

            stmtPessoaFisica.setInt(1, pessoa.id);
            stmtPessoaFisica.setString(2, pessoa.cpf);
            stmtPessoaFisica.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    public void alterar(PessoaFisica pessoa) {

        String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?,
estado = ?, telefone = ?, email = ? WHERE id = ?";

        String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE id = ?";


        try (Connection conn = ConectorBD.getConnection();

             PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa);

             PreparedStatement stmtPessoaFisica =
conn.prepareStatement(sqlPessoaFisica)) {


            stmtPessoa.setString(1, pessoa.nome);

            stmtPessoa.setString(2, pessoa.logradouro);

            stmtPessoa.setString(3, pessoa.cidade);

            stmtPessoa.setString(4, pessoa.estado);

            stmtPessoa.setString(5, pessoa.telefone);

            stmtPessoa.setString(6, pessoa.email);

            stmtPessoa.setInt(7, pessoa.id);

            stmtPessoa.executeUpdate();


            stmtPessoaFisica.setString(1, pessoa.cpf);

            stmtPessoaFisica.setInt(2, pessoa.id);

            stmtPessoaFisica.executeUpdate();


        } catch (SQLException e) {

            e.printStackTrace();
```

```
        }
    }

    public void excluir(int id) {
        String sqlPessoaFisica = "DELETE FROM PessoaFisica WHERE id = ?";
        String sqlPessoa = "DELETE FROM Pessoa WHERE id = ?";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmtPessoaFisica =
conn.prepareStatement(sqlPessoaFisica);
             PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {

            stmtPessoaFisica.setInt(1, id);
            stmtPessoaFisica.executeUpdate();

            stmtPessoa.setInt(1, id);
            stmtPessoa.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

PessoaJuridicaDAO.java

```
package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {

    public PessoaJuridica getPessoa(int id) {
        String sql = "SELECT * FROM Pessoa INNER JOIN PessoaJuridica ON Pessoa.id =
PessoaJuridica.id WHERE Pessoa.id = ?";
        PessoaJuridica pessoa = null;

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setInt(1, id);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                pessoa = new PessoaJuridica(
                    rs.getInt("id"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
```

```
                rs.getString("estado"),
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return pessoa;
}

public List<PessoaJuridica> getPessoas() {
    List<PessoaJuridica> pessoas = new ArrayList<>();
    String sql = "SELECT * FROM Pessoa INNER JOIN PessoaJuridica ON Pessoa.id =
PessoaJuridica.id";

    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql);
         ResultSet rs = stmt.executeQuery()) {

        while (rs.next()) {
            PessoaJuridica pessoa = new PessoaJuridica(
                rs.getInt("id"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
```

```
                rs.getString("telefone"),
                rs.getString("email"),
                rs.getString("cnpj")
            );
            pessoas.add(pessoa);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return pessoas;
}

public void incluir(PessoaJuridica pessoa) {
    String sqlPessoa = "INSERT INTO Pessoa (id, nome, logradouro, cidade, estado,
telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
    String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (id, cnpj) VALUES (?, ?)";

    int id = SequenceManager.getValue("PessoaSeq");
    pessoa.id = id;

    try (Connection conn = ConectorBD.getConnection();
         PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa);
         PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {

        stmtPessoa.setInt(1, pessoa.id);
        stmtPessoa.setString(2, pessoa.nome);
        stmtPessoa.setString(3, pessoa.logradouro);
```

```
            stmtPessoa.setString(4, pessoa.cidade);
            stmtPessoa.setString(5, pessoa.estado);
            stmtPessoa.setString(6, pessoa.telefone);
            stmtPessoa.setString(7, pessoa.email);
            stmtPessoa.executeUpdate();

            stmtPessoaJuridica.setInt(1, pessoa.id);
            stmtPessoaJuridica.setString(2, pessoa.cnpj);
            stmtPessoaJuridica.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void alterar(PessoaJuridica pessoa) {
        String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?,
estado = ?, telefone = ?, email = ? WHERE id = ?";
        String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE id = ?";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa);
             PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {

            stmtPessoa.setString(1, pessoa.nome);
            stmtPessoa.setString(2, pessoa.logradouro);
            stmtPessoa.setString(3, pessoa.cidade);
            stmtPessoa.setString(4, pessoa.estado);
            stmtPessoa.setString(5, pessoa.telefone);
```

```
            stmtPessoa.setString(6, pessoa.email);
            stmtPessoa.setInt(7, pessoa.id);
            stmtPessoa.executeUpdate();

            stmtPessoaJuridica.setString(1, pessoa.cnpj);
            stmtPessoaJuridica.setInt(2, pessoa.id);
            stmtPessoaJuridica.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void excluir(int id) {
        String sqlPessoaJuridica = "DELETE FROM PessoaJuridica WHERE id = ?";
        String sqlPessoa = "DELETE FROM Pessoa WHERE id = ?";

        try (Connection conn = ConectorBD.getConnection();
             PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica);
             PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {

            stmtPessoaJuridica.setInt(1, id);
            stmtPessoaJuridica.executeUpdate();

            stmtPessoa.setInt(1, id);
            stmtPessoa.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
```

```
    }
  }
}
```

CadastroBDTeste.java

```
package cadastrobd.model;

import java.util.List;

public class CadastroBDTeste {
    public static void main(String[] args) {
        PessoaFisicaDAO pfDao = new PessoaFisicaDAO();
        PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();

        System.out.println("=== Pessoa Física ===");

        PessoaFisica pf = new PessoaFisica(0, "Ruan Pablo", "Rua 20, Norte", "São Paulo",
"SP", "1212-1212", "ruanpablo@gmail.com", "11111111111");
        pfDao.incluir(pf);

        pf.nome = "Ruan Pablo Alterado";
        pf.cidade = "Campinas";
        pfDao.alterar(pf);

        List<PessoaFisica> listaPF = pfDao.getPessoas();
        for (PessoaFisica p : listaPF) {
            p.exibir();
            System.out.println();
        }

        pfDao.excluir(pf.id);

        System.out.println("\n=== Pessoa Jurídica ===");
```

```
        PessoaJuridica pj = new PessoaJuridica(0, "JCorp", "Destrito 10, Centro", "Rio de Janeiro", "RJ", "2222-2222", "jcorp@corp.com", "11111111111111");
        pjDao.incluir(pj);

        pj.nome = "JCorp Ltda.";
        pj.telefone = "2222-3333";
        pjDao.alterar(pj);



        List<PessoaJuridica> listaPJ = pjDao.getPessoas();
        for (PessoaJuridica p : listaPJ) {
            p.exibir();
            System.out.println();
        }

        pjDao.excluir(pj.id);

    }
}
```

Resultado

=== Pessoa Física ===

ID: 7

Nome: Ruan Pablo Alterado

Logradouro: Rua 20, Norte

Cidade: Campinas

Estado: SP

Telefone: 1212-1212

Email: ruanpablo@gmail.com

CPF: 11111111111

=== Pessoa Jurídica ===

ID: 8

Nome: JCorp Ltda.

Logradouro: Destrito 10, Centro

Cidade: Rio de Janeiro

Estado: RJ

Telefone: 2222-3333

Email: jcorp@corp.com

CNPJ: 11111111111111

Análise e Conclusão

A. Qual a importância dos componentes de middleware, como o JDBC?

R: Ele facilita a interação do java com banco de dados.

B. Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?

R: A forma com a qual as consultas SQL são feitas e executadas, no Statement o banco de dados precisa compilar a consulta a cada execusão, já no PreparedStatement, a consulta SQL é previamente compilada.

C. Como o padrão DAO melhora a manutenibilidade do software?

R: Ele simplifica o código deixando muito mais flexivel, organizado e de fácil manutenção.

D. Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

R: Em bancos de dados relacionais, a herança é feita através de três estratégias principais: Tabela Única, uma tabela com todos os atributos, usando uma coluna discriminadora e permitindo valores nulos, Tabelas Concretas. uma tabela para cada classe concreta, duplicando atributos da superclasse, e Tabelas por Classe, uma tabela para cada classe, com tabelas de subclasses referenciando a superclasse via chave estrangeira.

---

https://github.com/ErikBM2661/CadastroBD.git