Campus: POLO SAGUAÇU - JOINVILLE - SC

Curso: DESENVOLVIMENTO FULL STACK

Disciplina: Back-end Sem Banco Não Tem

Turma: 9001

Semestre: 1° Semestre (2024)

Aluno: Jederson Borges de Oliveira

Link: https://github.com/JedersonBorges/CadastroBD.git

BackEnd sem banco não tem

1° Procedimento

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**
6. **Server na persistência de dados.**

Códigos utilizados

Classe Pessoa

```
package cadastrobd.model;

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

    public Pessoa() {}

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

```
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    public int getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(int idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void exibir() {
        System.out.println("ID: " + idPessoa);
        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);
    }
}


    public int getId() {
        return id;
    }
```

```
    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

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

Classe PessoaFisica

```
package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private long cpf;

    public PessoaFisica() {}

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

    public long getCpf() {
        return cpf;
    }

    public void setCpf(long cpf) {
        this.cpf = cpf;
    }

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

Classe PessoaJuridica:

```
 package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    private long cnpj;

    public PessoaJuridica() {}
```

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

    public long getCnpj() {
        return cnpj;
    }

    public void setCnpj(long cnpj) {
        this.cnpj = cnpj;
    }

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

Classe ConectorBD:

```
 package cadastrobd.model.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConectorBD {
    private static final String URL =
"jdbc:sqlserver://JEDERSON16:1434;databaseName=loja;encrypt=true;trust
ServerCertificate=true";
    private static final String USER = "sa";
    private static final String PASSWORD = "loja";

    public static Connection getConnection() throws SQLException {
        try {
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
            throw e;
        }
    }
}
```

Classe SequenceManager:

```
 package cadastrobd.model.util;

import java.sql.Connection;
import java.sql.ResultSet;
```

```
import java.sql.Statement;

public class SequenceManager {
    public static int nextId(String tableName, String columnName) {
        int nextId = 1;
        try (Connection conn = ConectorBD.getConnection();
             Statement stmt = conn.createStatement()) {
            String query = "SELECT MAX(" + columnName + ") FROM " +
tableName;
            ResultSet rs = stmt.executeQuery(query);
            if (rs.next()) {
                nextId = rs.getInt(1) + 1;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return nextId;
    }
}
```

Classe PessoaFisicaDAO:

```
 package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {

    public void incluir(PessoaFisica pessoaFisica) {
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "INSERT INTO Pessoas (idPessoa, nome,
logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?,
?)";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                pessoaFisica.setIdPessoa(SequenceManager.nextId("Pesso
as", "idPessoa"));
                stmt.setInt(1, pessoaFisica.getIdPessoa());
                stmt.setString(2, pessoaFisica.getNome());
                stmt.setString(3, pessoaFisica.getLogradouro());
                stmt.setString(4, pessoaFisica.getCidade());
                stmt.setString(5, pessoaFisica.getEstado());
                stmt.setString(6, pessoaFisica.getTelefone());
                stmt.setString(7, pessoaFisica.getEmail());
                stmt.executeUpdate();
            }

            sql = "INSERT INTO PessoaFisica (cpf, idPessoa) VALUES (?,
?)";
```

```
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                stmt.setLong(1, pessoaFisica.getCpf());
                stmt.setInt(2, pessoaFisica.getIdPessoa());
                stmt.executeUpdate();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void atualizar(PessoaFisica pessoaFisica) {
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "UPDATE Pessoas SET nome=?, logradouro=?,
cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                stmt.setString(1, pessoaFisica.getNome());
                stmt.setString(2, pessoaFisica.getLogradouro());
                stmt.setString(3, pessoaFisica.getCidade());
                stmt.setString(4, pessoaFisica.getEstado());
                stmt.setString(5, pessoaFisica.getTelefone());
                stmt.setString(6, pessoaFisica.getEmail());
                stmt.setInt(7, pessoaFisica.getIdPessoa());
                stmt.executeUpdate();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public List<PessoaFisica> consultarTodos() {
        List<PessoaFisica> pessoasFisicas = new ArrayList<>();
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "SELECT * FROM Pessoas p JOIN PessoaFisica pf
ON p.idPessoa = pf.idPessoa";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                try (ResultSet rs = stmt.executeQuery()) {
                    while (rs.next()) {
                        PessoaFisica pessoaFisica = new PessoaFisica(
                                rs.getLong("cpf"),
                                rs.getInt("idPessoa"),
                                rs.getString("nome"),
                                rs.getString("logradouro"),
                                rs.getString("cidade"),
                                rs.getString("estado"),
                                rs.getString("telefone"),
                                rs.getString("email")
                        );
                        pessoasFisicas.add(pessoaFisica);
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return pessoasFisicas;
    }

    public void excluir(PessoaFisica pessoaFisica) {
```

```
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "DELETE FROM PessoaFisica WHERE idPessoa=?";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                stmt.setInt(1, pessoaFisica.getIdPessoa());
                stmt.executeUpdate();
            }

            sql = "DELETE FROM Pessoas WHERE idPessoa=?";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                stmt.setInt(1, pessoaFisica.getIdPessoa());
                stmt.executeUpdate();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Classe PessoaJuridicaDAO:

```
 package cadastrobd.model;

import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {

    public void incluir(PessoaJuridica pessoaJuridica) {
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "INSERT INTO Pessoas (idPessoa, nome,
logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?,
?)";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                pessoaJuridica.setIdPessoa(SequenceManager.nextId("Pes
soas", "idPessoa"));
                stmt.setInt(1, pessoaJuridica.getIdPessoa());
                stmt.setString(2, pessoaJuridica.getNome());
                stmt.setString(3, pessoaJuridica.getLogradouro());
                stmt.setString(4, pessoaJuridica.getCidade());
                stmt.setString(5, pessoaJuridica.getEstado());
                stmt.setString(6, pessoaJuridica.getTelefone());
                stmt.setString(7, pessoaJuridica.getEmail());
                stmt.executeUpdate();
            }

            sql = "INSERT INTO PessoaJuridica (cnpj, idPessoa) VALUES
(?, ?)";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
```

```
                stmt.setLong(1, pessoaJuridica.getCnpj());
                stmt.setInt(2, pessoaJuridica.getIdPessoa());
                stmt.executeUpdate();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void atualizar(PessoaJuridica pessoaJuridica) {
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "UPDATE Pessoas SET nome=?, logradouro=?,
cidade=?, estado=?, telefone=?, email=? WHERE idPessoa=?";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                stmt.setString(1, pessoaJuridica.getNome());
                stmt.setString(2, pessoaJuridica.getLogradouro());
                stmt.setString(3, pessoaJuridica.getCidade());
                stmt.setString(4, pessoaJuridica.getEstado());
                stmt.setString(5, pessoaJuridica.getTelefone());
                stmt.setString(6, pessoaJuridica.getEmail());
                stmt.setInt(7, pessoaJuridica.getIdPessoa());
                stmt.executeUpdate();
            }
        } catch (Exception
        e) {
            e.printStackTrace();
        }
    }

    public List<PessoaJuridica> consultarTodos() {
        List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "SELECT * FROM Pessoas p JOIN PessoaJuridica
pj ON p.idPessoa = pj.idPessoa";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                try (ResultSet rs = stmt.executeQuery()) {
                    while (rs.next()) {
                        PessoaJuridica pessoaJuridica = new
PessoaJuridica(
                                rs.getLong("cnpj"),
                                rs.getInt("idPessoa"),
                                rs.getString("nome"),
                                rs.getString("logradouro"),
                                rs.getString("cidade"),
                                rs.getString("estado"),
                                rs.getString("telefone"),
                                rs.getString("email")
                        );
                        pessoasJuridicas.add(pessoaJuridica);
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return pessoasJuridicas;
    }

    public void excluir(PessoaJuridica pessoaJuridica) {
```

```
        try (Connection conexao = ConectorBD.getConnection()) {
            String sql = "DELETE FROM PessoaJuridica WHERE
idPessoa=?";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                stmt.setInt(1, pessoaJuridica.getIdPessoa());
                stmt.executeUpdate();
            }

            sql = "DELETE FROM Pessoas WHERE idPessoa=?";
            try (PreparedStatement stmt =
conexao.prepareStatement(sql)) {
                stmt.setInt(1, pessoaJuridica.getIdPessoa());
                stmt.executeUpdate();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Classe SequenceManager:

```
package cadastrobd;

import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import java.util.List;

public class CadastroBDTeste {

    public static void main(String[] args) {
        // Instanciar e persistir Pessoa Fisica
        PessoaFisica pessoaFisica = new PessoaFisica(12345678901L, 16,
"Maria", "Rua XV de Novembro", "Joinville", "SC", "4799999090",
"mariacarolina@estacio.com.br");
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        pessoaFisicaDAO.incluir(pessoaFisica);

        // Alterar dados da Pessoa Fisica
        pessoaFisica.setNome("Maria Carolina");
        pessoaFisicaDAO.atualizar(pessoaFisica);

        // Consultar todas as Pessoas Fisicas
        List<PessoaFisica> pessoasFisicas =
pessoaFisicaDAO.consultarTodos();
        System.out.println("Pessoas Fisicas:");
        for (PessoaFisica pf : pessoasFisicas) {
            pf.exibir();
        }

        // Excluir a Pessoa Fisica criada anteriormente
        pessoaFisicaDAO.excluir(pessoaFisica);

        // Instanciar e persistir Pessoa Juridica
```

```
        PessoaJuridica pessoaJuridica = new
PessoaJuridica(12345678000199L, 23, "Aoop", "Avenida Beira Rio",
"Cidade Barueri", "SP", "21999999959", "aoop@empresa.com");
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
        pessoaJuridicaDAO.incluir(pessoaJuridica);

        // Alterar dados da Pessoa Juridica
        pessoaJuridica.setNome("Aoop Cloud Solutions");
        pessoaJuridicaDAO.atualizar(pessoaJuridica);

        // Consultar todas as Pessoas Juridicas
        List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.consultarTodos();
        System.out.println("Pessoas Juridicas:");
        for (PessoaJuridica pj : pessoasJuridicas) {
            pj.exibir();
        }

        // Excluir a Pessoa Juridica criada anteriormente
        pessoaJuridicaDAO.excluir(pessoaJuridica);
    }
}
```

Resultado no console (Pessoas Fisícas)

```
[java] Pessoas Fisicas:
[java] ID: 7
[java] Nome: Joao
[java] Logradouro: Rua 12, casa 3, Quitanda
[java] Cidade: Riacho do Sul
[java] Estado: PA
[java] Telefone: 1111-1111
[java] Email: joao@riacho.com
[java] CPF: 11111111111
[java] ID: 1
[java] Nome: Maria Carolina
[java] Logradouro: Rua XV de Novembro
[java] Cidade: Joinville
[java] Estado: SC
[java] Telefone: 4799999090
[java] Email: mariacarolina@estacio.com.br
[java] CPF: 12345678901
```

Resultado no console (Pessoa Jurídicas)

```
[java] Pessoas Juridicas:
[java] ID: 1
[java] Nome: Aoop Cloud Solutions
[java] Logradouro: Avenida Beira Rio
[java] Cidade: Cidade Barueri
[java] Estado: SP
[java] Telefone: 21999999959
[java] Email: aoop@empresa.com
[java] CNPJ: 12345678000199
[java] ID: 15
[java] Nome: JJC
[java] Logradouro: Rua 11, Centro
[java] Cidade: Riacho do Norte
[java] Estado: PA
[java] Telefone: 1212-1212
[java] Email: jcc@riacho.com
[java] CNPJ: 22222222222222
```

Análise e Conclusão

Qual a importância dos componentes de middleware, como JDBC?

Os componentes de middleware são importantes porque fazem a ponte entre a aplicação e o banco de dados, permitindo que a gente faça consultas e manipule dados de forma padronizada.

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

Statement é usado para executar instruções SQL simples, PreparedStatement é usado para instruções SQL parametrizadas, oferecendo maior segurança e melhor desempenho

Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO promove um design modular e facilita a manutenção;

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

Em um modelo estritamente relacional, a herança não é suportada diretamente em linguagens orientadas a objetos, para refletir a herança no banco de dados, normalmente utilizamos tabelas separadas para cada

classe derivada, com uma relação de chave estrangeira para cada tabela da classe base, ou uma única tabela que inclui todos os campos das classes base e derivadas, usando uma coluna para diferenciar os tipos.