Estácio

Missão Prática | Nível 3 | Mundo 3

Luís Ricardo Nogueira / 2023 0616 8625

RPG0016 - BackEnd sem banco não tem

Objetivo da Prática

Implementar persistência com base no middleware JDBC.

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

Implementar o mapeamento objeto-relacional em sistemas Java.

Criar sistemas cadastrais com persistência em banco relacional.

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

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Pessoa.java:

```
package cadastrobd.model;

public class Pessoa {
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private 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);
    }

    // getters e setters
}
```

PessoaFisica.java:

```
package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private 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);
    }

    // getters e setters
}
```

PessoaFisicaDAO.java:

```
package cadastrobd.model;

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

public class PessoaFisicaDAO {
    private ConectorBD conectorBD;
    private SequenceManager sequenceManager;

    public PessoaFisicaDAO() {
        conectorBD = new ConectorBD();
        sequenceManager = new SequenceManager();
    }

    public PessoaFisica getPessoa(int id) {
        PessoaFisica pessoaFisica = null;
        Connection connection = conectorBD.getConnection();
        try {
            PreparedStatement ps = connection.prepareStatement("SELECT * FROM Pessoa
WHERE id = ?");
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
```

```
            if (rs.next()) {
                pessoaFisica = 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();
        } finally {
            conectorBD.close(connection);
        }
        return pessoaFisica;
    }

    public List<PessoaFisica> getPessoas() {
        List<PessoaFisica> pessoasFisicas = new ArrayList<>();
        Connection connection = conectorBD.getConnection();
        try {
            PreparedStatement ps = connection.prepareStatement("SELECT * FROM Pessoa");
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                PessoaFisica pessoaFisica = 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"));
                pessoasFisicas.add(pessoaFisica);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conectorBD.close(connection);
        }
        return pessoasFisicas;
    }

    public void incluir(PessoaFisica pessoaFisica) {
        Connection connection = conectorBD.getConnection();
```

```
        try {
            int id = sequenceManager.getValue("Pessoa_seq");
            pessoaFisica.setId(id);
            PreparedStatement ps = connection.prepareStatement("INSERT INTO Pessoa (id, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)");
            ps.setInt(1, id);
            ps.setString(2, pessoaFisica.getNome());
            ps.setString(3, pessoaFisica.getLogradouro());
            ps.setString(4, pessoaFisica.getCidade());
            ps.setString(5, pessoaFisica.getEstado());
            ps.setString(6, pessoaFisica.getTelefone());
            ps.setString(7, pessoaFisica.getEmail());
            ps.executeUpdate();
            ps = connection.prepareStatement("INSERT INTO PessoaFisica (id, cpf) VALUES (?, ?)");
            ps.setInt(1, id);
            ps.setString(2, pessoaFisica.getCpf());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conectorBD.close(connection);
        }
    }

    public void alterar(PessoaFisica pessoaFisica) {
        Connection connection = conectorBD.getConnection();
        try {
            PreparedStatement ps = connection.prepareStatement("UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?");
            ps.setString(1, pessoaFisica.getNome());
            ps.setString(2, pessoaFisica.getLogradouro());
            ps.setString(3, pessoaFisica.getCidade());
            ps.setString(4, pessoaFisica.getEstado());
            ps.setString(5, pessoaFisica.getTelefone());
            ps.setString(6, pessoaFisica.getEmail());
```

```
            ps.setInt(7, pessoaFisica.getId());
            ps.executeUpdate();
            ps = connection.prepareStatement("UPDATE PessoaFisica SET cpf = ? WHERE
id = ?");
            ps.setString(1, pessoaFisica.getCpf());
            ps.setInt(2, pessoaFisica.getId());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conectorBD.close(connection);
        }
    }

    public void excluir(int id) {
        Connection connection = conectorBD.getConnection();
        try {
            PreparedStatement ps = connection.prepareStatement("DELETE FROM
PessoaFisica WHERE id = ?");
            ps.setInt(1, id);
            ps.executeUpdate();
            ps = connection.prepareStatement("DELETE FROM Pessoa WHERE id = ?");
            ps.setInt(1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conectorBD.close(connection);
        }
    }
}
```

PessoaJuridica.java:

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
```

```
    private 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);
    }

    // getters e setters
}
```

PessoaJuridicaDAO.java:

```
package cadastrobd.model;

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

public class PessoaJuridicaDAO {
    private ConectorBD conectorBD;
    private SequenceManager sequenceManager;

    public PessoaJuridicaDAO() {
        conectorBD = new ConectorBD();
        sequenceManager = new SequenceManager();
```

```
    }

    public PessoaJuridica getPessoa(int id) {
        PessoaJuridica pessoaJuridica = null;
        Connection connection = conectorBD.getConnection();
        try {
            PreparedStatement ps = connection.prepareStatement("SELECT * FROM Pessoa
WHERE id = ?");
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                pessoaJuridica = 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();
        } finally {
            conectorBD.close(connection);
        }
        return pessoaJuridica;
    }

    public List<PessoaJuridica> getPessoas() {
        List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
        Connection connection = conectorBD.getConnection();
        try {
            PreparedStatement ps = connection.prepareStatement("SELECT * FROM
Pessoa");
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                PessoaJuridica pessoaJuridica = 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"));
                pessoasJuridicas.add(pessoaJuridica);
            }
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        } finally {
            conectorBD.close(connection);
        }
        return pessoasJuridicas;
    }

    public void incluir(PessoaJuridica pessoaJuridica) {
        Connection connection = conectorBD.getConnection();
        try {
            int id = sequenceManager.getValue("Pessoa_seq");
            pessoaJuridica.setId(id);
            PreparedStatement ps = connection.prepareStatement("INSERT INTO Pessoa
(id, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)");
            ps.setInt(1, id);
            ps.setString(2, pessoaJuridica.getNome());
            ps.setString(3, pessoaJuridica.getLogradouro());
            ps.setString(4, pessoaJuridica.getCidade());
            ps.setString(5, pessoaJuridica.getEstado());
            ps.setString(6, pessoaJuridica.getTelefone());
            ps.setString(7, pessoaJuridica.getEmail());
            ps.executeUpdate();
            ps = connection.prepareStatement("INSERT INTO PessoaJuridica (id, cnpj)
VALUES (?, ?)");
            ps.setInt(1, id);
            ps.setString(2, pessoaJuridica.getCnpj());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conectorBD.close(connection);
        }
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        Connection connection = conectorBD.getConnection();
```

```
        try {
            PreparedStatement ps = connection.prepareStatement("UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?");
            ps.setString(1, pessoaJuridica.getNome());
            ps.setString(2, pessoaJuridica.getLogradouro());
            ps.setString(3, pessoaJuridica.getCidade());
            ps.setString(4, pessoaJuridica.getEstado());
            ps.setString(5, pessoaJuridica.getTelefone());
            ps.setString(6, pessoaJuridica.getEmail());
            ps.setInt(7, pessoaJuridica.getId());
            ps.executeUpdate();
            ps = connection.prepareStatement("UPDATE PessoaJuridica SET cnpj = ? WHERE id = ?");
            ps.setString(1, pessoaJuridica.getCnpj());
            ps.setInt(2, pessoaJuridica.getId());
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conectorBD.close(connection);
        }
    }

    public void excluir(int id) {
        Connection connection = conectorBD.getConnection();
        try {
            PreparedStatement ps = connection.prepareStatement("DELETE FROM PessoaJuridica WHERE id = ?");
            ps.setInt(1, id);
            ps.executeUpdate();
            ps = connection.prepareStatement("DELETE FROM Pessoa WHERE id = ?");
            ps.setInt(1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
```

```
            conectorBD.close(connection);
        }
    }
}
```

CadastroDBTeste.java:

```
package cadastrobd.model;

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

        // Instanciar uma pessoa física e persistir no banco de dados
        PessoaFisica pessoaFisica = new PessoaFisica("João da Silva", "Rua dos Bobos", "São Paulo", "SP", "11999999999", "joao@email.com", "12345678909");
        pessoaFisicaDAO.incluir(pessoaFisica);

        // Alterar os dados da pessoa física no banco
        pessoaFisica.setNome("João da Silva Junior");
        pessoaFisicaDAO.alterar(pessoaFisica);

        // Consultar todas as pessoas físicas do banco de dados e listar no console
        List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
        for (PessoaFisica pf : pessoasFisicas) {
            System.out.println(pf.toString());
        }

        // Excluir a pessoa física criada anteriormente no banco
        pessoaFisicaDAO.excluir(pessoaFisica.getId());

        // Instanciar uma pessoa jurídica e persistir no banco de dados
        PessoaJuridica pessoaJuridica = new PessoaJuridica("Empresa XYZ Ltda", "Rua dos Empreendedores", "São Paulo", "SP", "11999999999", "empresa@email.com", "12345678901234");
```

```
        pessoaJuridicaDAO.incluir(pessoaJuridica);

        // Alterar os dados da pessoa jurídica no banco
        pessoaJuridica.setNome("Empresa XYZ Ltda ME");
        pessoaJuridicaDAO.alterar(pessoaJuridica);

        // Consultar todas as pessoas jurídicas do banco de dados e listar no console
        List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
        for (PessoaJuridica pj : pessoasJuridicas) {
            System.out.println(pj.toString());
        }

        // Excluir a pessoa jurídica criada anteriormente no banco
        pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
    }
}
```

Perguntas:

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

R: Os componentes de middleware, como o JDBC (Java Database Connectivity), desempenham um papel em uma aplicação, pois atuam como uma camada intermediária entre a aplicação e os recursos externos, como bancos de dados, sistemas de arquivos, redes, etc.

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

R: Diferenças entre Statement e PreparedStatement

Compilação de SQL: PreparedStatement compila o SQL uma vez, enquanto Statement compila a cada execução.

Parâmetros de consulta: PreparedStatement suporta parâmetros de consulta, enquanto Statement não.

Tipos de dados: PreparedStatement suporta tipos de dados Java nativos, enquanto Statement requer conversão para strings.

Flexibilidade: Statement é mais flexível, mas menos seguro e eficiente do que PreparedStatement.

Resumo: PreparedStatement é mais seguro e eficiente para consultas SQL repetidas ou com parâmetros, enquanto Statement é mais flexível, mas menos seguro e eficiente.

c) Como o padrão DAO melhora a manutenibilidade do software?

R: O padrão DAO melhora a manutenibilidade do software ao separar a lógica de negócios da lógica de acesso a dados, tornando o código mais modular, flexível e fácil de manter.

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

R: A herança não é suportada nativamente em modelos estritamente relacionais. Em vez disso, são utilizadas estratégias para simular a herança:

Tabela por Classe: Cada classe herdeira tem sua própria tabela.

Tabela por Hierarquia: Uma tabela para a classe pai e tabelas adicionais para as classes filhas.

Tabela com Coluna de Tipo: Uma tabela única com uma coluna adicional para indicar o tipo de objeto (classe pai ou filha).

Essas estratégias permitem representar a herança em um modelo relacional, mas não há uma forma direta de implementar herança como em linguagens de programação orientadas a objetos.

2º Procedimento | Alimentando a Base

```
public static void main(String[] args) {
   Scanner scanner = new Scanner(System.in);

   while (true) {
```

```
System.out.println("Opções:");
System.out.println("1 - Incluir");
System.out.println("2 - Alterar");
System.out.println("3 - Excluir");
System.out.println("4 - Exibir pelo ID");
System.out.println("5 - Exibir todos");
System.out.println("0 - Sair");

int opcao = scanner.nextInt();

switch (opcao) {
    case 1:
        incluir(scanner);
        break;
    case 2:
        alterar(scanner);
        break;
    case 3:
        excluir(scanner);
        break;
    case 4:
        obter(scanner);
        break;
    case 5:
        obterTodos(scanner);
        break;
```

```
            case 0:
                System.out.println("Saindo...");
                return;
            default:
                System.out.println("Opção inválida");
        }
    }
}

private static void incluir(Scanner scanner) {
    System.out.println("Incluir:");
    System.out.println("1 - Pessoa Física");
    System.out.println("2 - Pessoa Jurídica");

    int tipo = scanner.nextInt();

    PessoaDAO dao;
    if (tipo == 1) {
        dao = new PessoaFisicaDAO();
    } else {
        dao = new PessoaJuridicaDAO();
    }

    System.out.println("Digite os dados:");
    // ler dados do teclado e criar objeto Pessoa
    Pessoa pessoa = new Pessoa(/* dados */);
```

```
    dao.incluir(pessoa);
}

private static void alterar(Scanner scanner) {
    System.out.println("Alterar:");
    System.out.println("1 - Pessoa Física");
    System.out.println("2 - Pessoa Jurídica");

    int tipo = scanner.nextInt();

    PessoaDAO dao;
    if (tipo == 1) {
        dao = new PessoaFisicaDAO();
    } else {
        dao = new PessoaJuridicaDAO();
    }

    System.out.println("Digite o ID:");
    int id = scanner.nextInt();

    Pessoa pessoa = dao.obter(id);
    if (pessoa != null) {
        System.out.println("Dados atuais:");
        // imprimir dados atuais
        System.out.println("Digite os novos dados:");
        // ler dados do teclado e atualizar objeto Pessoa
```

```
            dao.alterar(pessoa);
        } else {
            System.out.println("Pessoa não encontrada");
        }
    }

    private static void excluir(Scanner scanner) {
        System.out.println("Excluir:");
        System.out.println("1 - Pessoa Física");
        System.out.println("2 - Pessoa Jurídica");

        int tipo = scanner.nextInt();

        PessoaDAO dao;
        if (tipo == 1) {
            dao = new PessoaFisicaDAO();
        } else {
            dao = new PessoaJuridicaDAO();
        }

        System.out.println("Digite o ID:");
        int id = scanner.nextInt();

        dao.excluir(id);
    }
```

```
private static void obter(Scanner scanner) {
    System.out.println("Obter:");
    System.out.println("1 - Pessoa Física");
    System.out.println("2 - Pessoa Jurídica");

    int tipo = scanner.nextInt();

    PessoaDAO dao;
    if (tipo == 1) {
        dao = new PessoaFisicaDAO();
    } else {
        dao = new PessoaJuridicaDAO();
    }

    System.out.println("Digite o ID:");
    int id = scanner.nextInt();

    Pessoa pessoa = dao.obter(id);
    if (pessoa != null) {
        System.out.println("Dados:");
        // imprimir dados
    } else {
        System.out.println("Pessoa não encontrada");
    }
}
```

```
private static void obterTodos(Scanner scanner) {
    System.out.println("Obter todos:");
    System.out.println("1 - Pessoa Física");
    System.out.println("2 - Pessoa Jurídica");

    int tipo = scanner.nextInt();

    PessoaDAO dao;
    if (tipo == 1) {
        dao = new PessoaFisicaDAO();
    } else {
        dao = new PessoaJuridicaDAO();
    }

    List<Pessoa> pessoas = dao.obterTodos();
    for (Pessoa pessoa : pessoas) {
        System.out.println("Dados:");
        // imprimir dados
    }
}
```

a) Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

R: Em resumo, a persistência em arquivo é mais simples e adequada para aplicações pequenas e simples, enquanto a persistência em banco de dados é mais robusta e escalável, adequada para aplicações mais complexas e que requerem alta disponibilidade e segurança.

b) Como o uso de operador *lambda* simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?

R: Antes do Java 8, imprimir os valores contidos nas entidades era um processo mais verboso e requereria mais código. Com a introdução do operador lambda e da API de Streams, a impressão de valores se tornou mais concisa e eficiente.

c) Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como *static*?

R: Em resumo, os métodos acionados diretamente pelo método main precisam ser marcados como static porque o método main é chamado antes de qualquer objeto ser criado, e os métodos não estáticos precisam de um objeto para serem chamados.