Universidade Estácio de Sá

Polo Estácio EAD - Batatais / SP

Curso: Desenvolvimento Full Stack
Disciplina: BackEnd sem banco não tem
Missão Prática - Nível 3
Turma: RPG0016
Semestre Letivo: 2024.3
Integrante: Daiana Maira de Oliveira Lascala

Título

Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.

Objetivo 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.

Descrição dos Códigos:

Neste projeto, foram implementadas as seguintes classes

- **Classe Pessoa**: Classe base com campos comuns e método para exibir os dados.
- **Classe PessoaFisica**: Herdou de Pessoa e adicionou o campo CPF, incluindo construtores e método sobrescrito para exibir os dados.
- **Classe PessoaJuridica**: Herdou de Pessoa e adicionou o campo CNPJ, também com construtores e método sobrescrito para exibir os dados.
- **Classe ConectorBD**: Responsável pela conexão com o banco de dados.
- **Classe SequenceManager**: Gerencia a sequência de IDs para as inserções.
- **Classes PessoaFisicaDAO e PessoaJuridicaDAO**: Implementaram os métodos para CRUD (Create, Read, Update, Delete) no banco de dados.

Os resultados da execução dos códigos

Inclusão de Pessoa Física: Sucesso

Alteração de Pessoa Física: Sucesso

Lista de Pessoas Físicas:

Pessoa Física: Nome: João Silva, Telefone: 88888-8888

Exclusão de Pessoa Física: Sucesso

Inclusão de Pessoa Jurídica: Sucesso

Alteração de Pessoa Jurídica: Sucesso

Lista de Pessoas Jurídicas:

Pessoa Jurídica: Nome: Empresa XYZ, Telefone: 96666-6666

Exclusão de Pessoa Jurídica: Sucesso

Análise e Conclusão:

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

   Os componentes de middleware, como o JDBC (Java Database Connectivity), são essenciais para a comunicação entre aplicações Java e bancos de dados. O JDBC oferece uma API padrão que facilita o acesso a dados, permitindo que desenvolvedores realizem operações como consultas, inserções e atualizações de forma consistente e eficiente. Sem o JDBC, seria necessário desenvolver um código específico para cada tipo de banco de dados, o que aumentaria a complexidade e reduziria a portabilidade da aplicação.

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

   O Statement e o PreparedStatement são usados para executar instruções SQL em um banco de dados, mas têm diferenças significativas:

   Statement: Utilizado para executar instruções SQL simples. Cada vez que uma nova instrução é executada, a consulta deve ser recompilada, o que pode impactar a performance em operações repetitivas.

   PreparedStatement: Permite que você preencha os parâmetros da consulta e compile a instrução apenas uma vez. Isso não só melhora a performance, mas também oferece maior segurança contra ataques de SQL injection, pois os parâmetros são tratados de maneira segura.

3. **Como o padrão DAO melhora a manutenibilidade do software?**

O padrão DAO (Data Access Object) promove a separação das preocupações no código, isolando a lógica de acesso a dados da lógica de negócios. Isso traz vários benefícios para a manutenibilidade do software:

- **Encapsulamento**: A lógica de acesso a dados é centralizada, facilitando a manutenção e a implementação de alterações.
- **Testabilidade**: A separação permite que os desenvolvedores realizem testes unitários nas classes de negócios sem depender de operações diretas no banco de dados.

- **Flexibilidade**: Mudanças no banco de dados ou na forma como os dados são acessados podem ser feitas com mínimo impacto na lógica de negócios.

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

Em um modelo de banco de dados estritamente relacional, a herança pode ser um desafio, pois o SQL não possui suporte nativo para a herança de classes como em linguagens de programação orientadas a objetos. As abordagens comuns para representar herança em um banco de dados relacional incluem:

- **Tabela por Hierarquia**: Criar uma tabela para a classe base e tabelas separadas para cada subclasse, onde cada subclasse armazena apenas seus atributos adicionais.
- **Tabela por Classe**: Criar uma tabela para cada classe (base e subclasses) e incluir todos os atributos. Isso pode resultar em colunas nulas para atributos não aplicáveis, mas mantém todas as informações em tabelas separadas.
- **Tabela Única**: Criar uma única tabela que armazena todos os atributos das classes, incluindo um tipo de coluna para diferenciar entre os tipos de registros. Isso simplifica as operações, mas pode causar problemas de integridade de dados.

Repositório Git:

https://github.com/DaianaLascala/Missao-3---BackEnd-sem-banco-n-o-tem