Java DAO

Java é uma linguagem de programação orientada a objetos, e faz parte do ecossistema de OOP (Object Oriented Programming). Isso significa que a principal função da linguagem é desenvolver programas e sistemas por meio de objetos interconectados, usando classes, métodos e atributos dos objetos para resolver o problema. No entanto, em um escopo mais geral, fora do educativo, muitas grandes empresas desenvolvem aplicações de mundo real em Java, usando diferentes técnicas de desenvolvimento.

Quando tratam de uma grande quantidade de dados, a maioria das empresas preferem utilizar uma base de dados ou um banco de dados para o armazenamento e tratamento deles, muitas vezes por meio de uma empresa terceira que garante a segurança dos dados. Um dos jeitos de fazer isso é por meio do DAO (Data Access Object), que garante a transparência dos dados durante a transferência e a segurança no armazenamento.

Diagram

Description automatically generatedO DAO é um objeto intermediário que pega as informações do Business Object (BO) utilizado pela empresa e transfere para a base de dados. Pode ser composto por mais de uma classe mas o propósito segue o mesmo, ser um adaptador entre a empresa e seus dados.

No exemplo que desenvolvemos, a empresa é o Mackenzie, em que guarda dados de alunos como nome, TIA, curso e número da chamada. A empresa pode criar um objeto do tipo student, e por meio da classe StudentDAOJDBC, que por sua vez faz a conexão com a base SQL pela classe ConnectionFactory, pode adicionar, editar, remover e consultar os dados presentes na base sem diretamente acessá-la. Este é apenas um exemplo simples do que um DAO pode fazer, exemplos maiores são compostos por mais classes Factory para gerar o objeto e a conexão, e possuem diversos processos a serem executados em uma devida base de dados.

No entanto, assim como qualquer outra tecnologia, há vantagens e desvantagens no uso do DAO com Java. Vamos discutir algumas delas abaixo:

**Vantagens do uso do DAO com Java:**

1. Separação de responsabilidades: O uso do padrão DAO permite a separação de responsabilidades entre a lógica de negócios e a lógica de acesso a dados. Isso torna o código mais modular e fácil de manter.
2. Maior segurança: O uso do DAO pode melhorar a segurança do aplicativo, pois é possível validar e verificar os dados que estão sendo enviados para o banco de dados antes de serem salvos.
3. Flexibilidade: O uso do DAO permite que diferentes bancos de dados sejam usados sem afetar a lógica de negócios do aplicativo. Isso torna o código mais flexível e escalável.
4. Testabilidade: O DAO torna o código mais testável, pois a camada de acesso a dados pode ser facilmente substituída por uma versão em memória ou mock durante os testes unitários.

**Desvantagens do uso do DAO com Java:**

1. Overhead de código: O uso do DAO pode levar a um overhead de código, pois é necessário escrever um código adicional para cada entidade de banco de dados.
2. Aumento de complexidade: O uso do DAO pode aumentar a complexidade do aplicativo, pois adiciona uma camada adicional de abstração que pode ser difícil de entender para desenvolvedores menos experientes.
3. Desempenho: O uso do DAO pode afetar o desempenho do aplicativo, pois adiciona uma camada adicional de abstração que pode levar a mais operações de I/O.

**Padrões de Projeto**

Os padrões de projeto em Java são soluções testadas e comprovadas para problemas recorrentes no desenvolvimento de software orientado a objetos. Eles são um conjunto de regras e diretrizes para projetar e implementar software que é fácil de entender, manter e modificar. Os padrões de projeto em Java são usados ​​para ajudar os desenvolvedores a resolver problemas comuns, como gerenciamento de memória, persistência de dados, segurança e escalabilidade.

Os padrões de projeto em Java são importantes porque ajudam os desenvolvedores a escrever código de alta qualidade que é fácil de manter e escalar. Ao usar padrões de projeto em Java, os desenvolvedores podem garantir que o código seja robusto, eficiente e escalável. Eles também podem ajudar a reduzir a complexidade do código, tornando-o mais fácil de entender e modificar. Além disso, os padrões de projeto em Java são amplamente utilizados na indústria de desenvolvimento de software e são um requisito comum para muitas posições de desenvolvimento de software.

Principais Exemplos

1. Padrão de projeto Singleton: este padrão de projeto é usado quando apenas uma instância de uma classe deve ser criada em todo o sistema. Ele garante que a classe tenha apenas uma instância e fornece um ponto de acesso global para essa instância.
2. Padrão de projeto Factory: este padrão de projeto é usado para criar objetos sem expor a lógica de criação ao cliente. Ele fornece uma interface para criar objetos de diferentes tipos, mas permite que as subclasses decidam quais objetos criar.
3. Padrão de projeto Observer: este padrão de projeto é usado quando um objeto precisa ser notificado quando outro objeto muda de estado. Ele permite que objetos dependentes sejam atualizados automaticamente quando o objeto observado é alterado.

Ao aprender sobre um padrão de projeto, é importante ter em mente alguns pontos-chave que ajudarão a compreender e aplicar corretamente o padrão. Sem ter uma boa noção do que você quer desenvolver, pode cair em uma aplicação falha ou completamente errada, assim os pontos abaixo são cruciais para implementar qualquer padrão.

1. Compreensão do problema: Antes de aplicar um padrão de projeto, é importante entender o problema que está sendo resolvido. O padrão deve ser aplicado para resolver um problema específico e não apenas porque é popular ou está na moda.
2. Conhecimento do padrão: É importante ter um bom conhecimento sobre o padrão de projeto, incluindo seus princípios, características e benefícios. É importante saber quando e como aplicá-lo corretamente.
3. Contexto de aplicação: O contexto de aplicação do padrão de projeto é fundamental para entender como ele pode ser aplicado. É importante entender como o padrão pode ser aplicado em diferentes cenários e como ele se relaciona com outras partes do sistema.
4. Implementação correta: É importante implementar o padrão corretamente, seguindo as melhores práticas e princípios de programação. Isso inclui a escolha adequada dos componentes do padrão e a organização do código.
5. Consideração da manutenibilidade: É importante considerar a manutenibilidade do código quando se aplica um padrão de projeto. O padrão deve tornar o código mais fácil de manter e modificar, e não mais difícil.
6. Avaliação dos resultados: É importante avaliar os resultados da aplicação do padrão de projeto. Isso inclui a medição da eficácia do padrão em resolver o problema, a avaliação da qualidade do código resultante e a medição do impacto no desempenho do sistema.

Fontes:

<https://www.roseindia.net/tutorial/java/jdbc/dataaccessobjectdesignpattern.html>

<https://www.oracle.com/java/technologies/dataaccessobject.html>

ChatGPT foi usado para escrever todas as partes do artigo após a primeira página