Em alguns projetos em Java, dependendo da tecnologia escolhida, é comum encontrarmos classes que seguem o padrão **DAO**, utilizado para isolar o acesso aos dados. Entretanto, neste curso utilizaremos um outro padrão, conhecido como **Repository**.

Mas aí podem surgir algumas dúvidas: qual a diferença entre as duas abordagens e o porquê dessa escolha?

**Padrão DAO**

O padrão de projeto DAO, conhecido também por **Data Access Object**, é utilizado para persistência de dados, onde seu principal objetivo é separar regras de negócio de regras de acesso a banco de dados. Nas classes que seguem esse padrão, isolamos todos os códigos que lidam com conexões, comandos SQLs e funções diretas ao banco de dados, para que assim tais códigos não se espalhem por outros pontos da aplicação, algo que dificultaria a manutenção do código e também a troca das tecnologias e do mecanismo de persistência.

**Implementação**

Vamos supor que temos uma tabela de produtos em nosso banco de dados. A implementação do padrão DAO seria o seguinte:

Primeiro, seria necessário criar uma classe básica de domínio Produto:

**public** **class** **Produto** {

**private** Long id;

**private** String nome;

**private** BigDecimal preco;

**private** String descricao;

// construtores, getters e setters

}

COPIAR CÓDIGO

Em seguida, precisaríamos criar a classe ProdutoDao, que fornece operações de persistência para a classe de domínio Produto:

**public** **class** **ProdutoDao** {

**private** **final** EntityManager entityManager;

**public** **ProdutoDao**(EntityManager entityManager) {

this.entityManager = entityManager;

}

**public** **void** **create**(Produto produto) {

entityManager.persist(produto);

}

**public** Produto **read**(Long id) {

**return** entityManager.find(Produto.class, id);

}

**public** **void** **update**(Produto produto) {

entityManger.merge(produto);

}

**public** **void** **remove**(Produto produto) {

entityManger.remove(produto);

}

}

COPIAR CÓDIGO

No exemplo anterior foi utilizado a JPA como tecnologia de persistência dos dados da aplicação.

**Padrão Repository**

De acordo com o famoso livro *Domain-Driven Design*, de Eric Evans:

O repositório é um mecanismo para encapsular armazenamento, recuperação e comportamento de pesquisa, que emula uma coleção de objetos.

Simplificando, um repositório também lida com dados e oculta consultas semelhantes ao DAO. No entanto, ele fica em um nível mais alto, mais próximo da lógica de negócios de uma aplicação. Um repositório está vinculado à regra de negócio da aplicação e está associado ao agregado dos seus objetos de negócio, retornando-os quando preciso.

Só que devemos ficar atentos, pois assim como no padrão DAO, regras de negócio que estão envolvidas com processamento de informações não devem estar presentes nos repositórios. Os repositórios não devem ter a responsabilidade de tomar decisões, aplicar algoritmos de transformação de dados ou prover serviços diretamente a outras camadas ou módulos da aplicação. Mapear entidades de domínio e prover as funcionalidades da aplicação são responsabilidades muito distintas.

Um repositório fica entre as regras de negócio e a camada de persistência:

1. Ele provê uma interface para as regras de negócio onde os objetos são acessados como em uma coleção;
2. Ele usa a camada de persistência para gravar e recuperar os dados necessários para persistir e recuperar os objetos de negócio.

**Por que o padrão repository ao invés do DAO utilizando Spring?**

O padrão de repositório incentiva um design orientado a domínio, fornecendo uma compreensão mais fácil do domínio e da estrutura de dados. Além disso, utilizando o repository do Spring não temos que nos preocupar em utilizar diretamente a API da JPA, bastando apenas criar os métodos que o Spring cria a implementação em tempo de execução, deixando o código muito mais simples, menor e legível.