JAVAEE, SPRING E JPA

Prof. Júlio Machado

julio.machado@pucrs.br

PADRÕES DE PROJETO

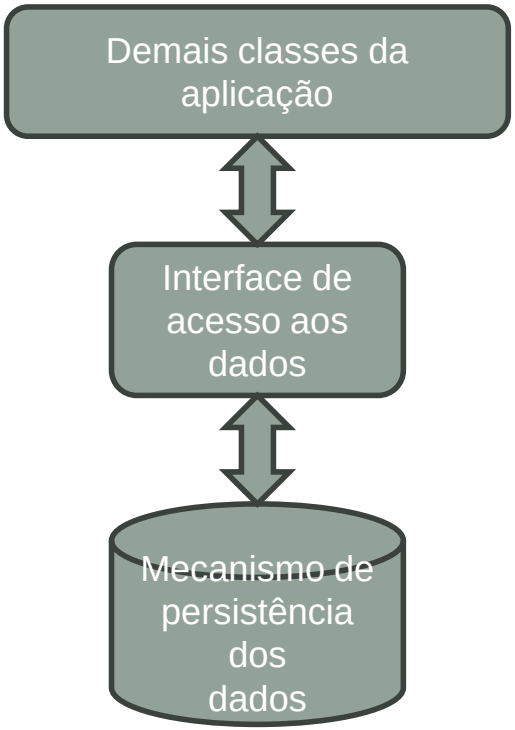
Introdução

Repositórios

- A grande maioria das aplicações corporativas necessita armazenar dados em meios não voláteis.
- Dependendo da necessidade esses dados podem ser armazenados em arquivos, bancos de dados relacionais, bancos de dados não relacionais, serviços de nuvem, etc.
- É usual também que, ao longo do tempo, as necessidades se alterem, e seja necessário trocar o fornecedor da tecnologia (provedor do SGBD) ou a própria tecnologia de armazenamento (passar de arquivos para um BD ou de BD para um serviço de nuvem).
- A questão que fica, então, é: como garantir a independência da aplicação em relação a tecnologia de armazenamento dos dados?

Isolando o acesso aos dados

- A melhor maneira de isolar uma aplicação das tecnologias de acesso aos dados é criar uma interface de acesso aos dados que seja independente da tecnologia utilizada.
  - Todo o restante da aplicação deverá se comunicar com os dados através dessa interface.
- A interface deve dispor de comandos de alto nível tais como:
  - Cadastrar um item
  - Remover um item
  - Alterar um item
  - Recuperar um item por seu identificador único
  - Recuperar todos os itens
  - Recuperar a lista de itens por atributo
  - etc.
- Dessa forma se a tecnologia de acesso aos dados mudar, basta criar uma nova classe que atenda a essa interface sem a necessidade de outra alterações na aplicação.



Padrão Repository

- O padrão “Repository” procura exatamente solucionar o problema de isolar uma aplicação do acesso aos dados.
- Um repositório executa as tarefas de um intermediário entre lógica da aplicação e a persistência dos dados, funcionando de maneira semelhante a um conjunto de objetos na memória.
- Os objetos de clientes criam consultas de forma declarativa e enviam-nas para os repositórios buscando respostas.
- Conceitualmente, um repositório encapsula um conjunto de objetos armazenados em um banco de dados – ou outro tipo de mecanismo de persistência – e as operações que podem ser executadas sobre eles.

Padrão Repository: contexto

- Um sistema necessita trabalhar com coleções de entidades (objetos de domínio) que estão armazenadas em algum tipo de mecanismo de persistência.
- Deseja-se manter o acoplamento baixo, mantendo os objetos de domínio ignorantes em relação aos mecanismos de persistência utilizados.

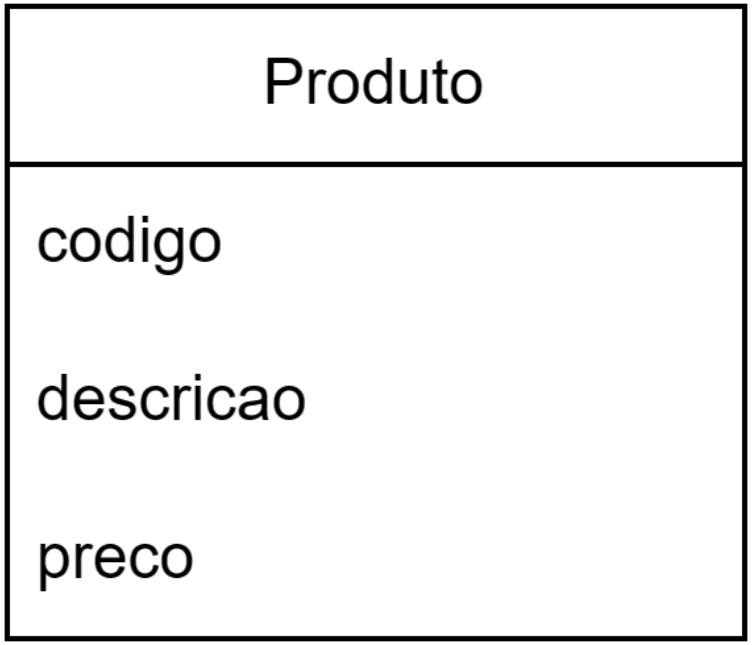
Padrão Repository: solução

- Implementar uma classe que abstrai as operações básicas de acesso a dados (adicionar, atualizar, remover, consultar) com uma interface semelhante a de uma coleção em memória (lista, dicionário, etc.).
- Deve-se usar um “Repository” para cada entidade ou objeto de domínio.

Estudo de caso: um repositório de produtos

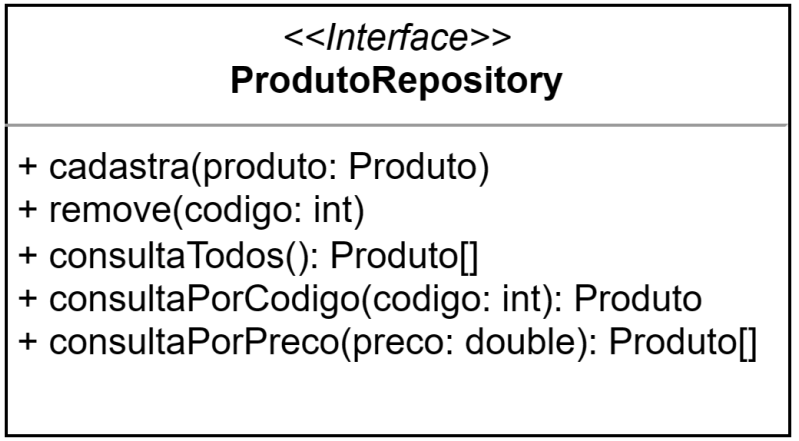
Passo 1: definindo a entidade

- O padrão “Repository” trabalha com o conceito de entidade. Cada repositório é responsável por manter as instâncias de um tipo de entidade.
- Neste caso queremos persistir os produtos vendidos por uma loja.
- Então iremos criar uma classe que modela este tipo de entidade.



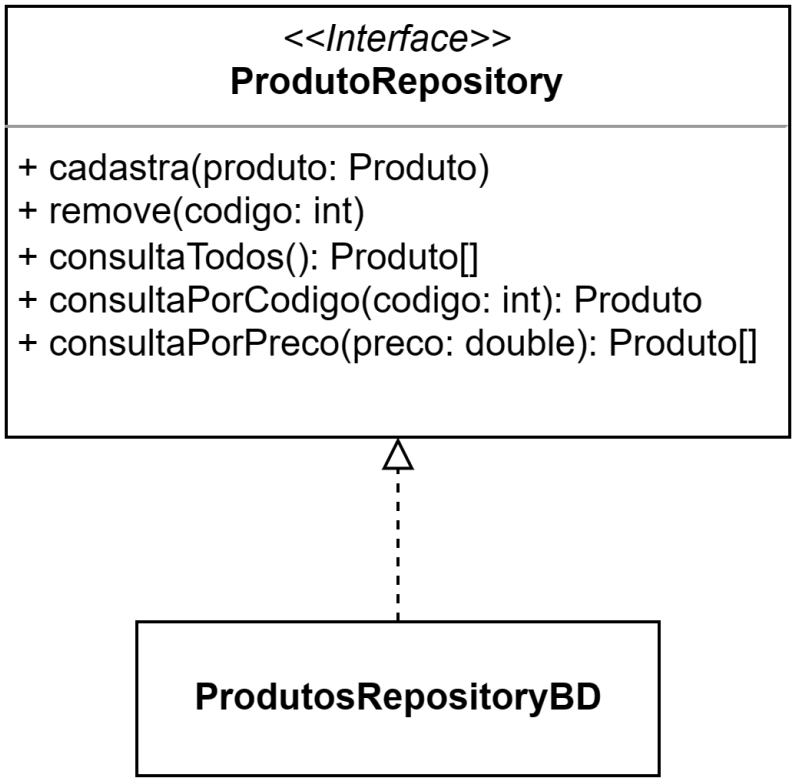
Passo 2: definindo a interface do Repository

- Um repositório implementa um conjunto de operações de acesso a dados definido em uma interface. Então, é necessário definir a interface do "Repository", isto é, a interface padrão de acesso as instancias da entidade alvo.



Passo 3: implementando a interface

- A interface deve ser implementada através de uma classe concreta.
- A tecnologia utilizada fica encapsulada dentro do repositório.



SPRING DATA

Introdução

Spring Data

- É um projeto Spring que proporciona um framework para a implementação de acesso diferentes fontes de dados
- Suporta:
  - Provê a implementação do Padrão *Repository* através da interface *Repository<T,ID>*
  - Geração dinâmica de código para manipulação de dados a partir das assinaturas de métodos
  - Implementação de classes de domínio de negócio a partir de propriedades básicas
  - Integração avançada com Spring MVC
  - Etc.

Spring Data

- Fornece diferentes implementações de código-base para todas as operações de CRUD a partir de interfaces tais como *CrudRepository<T,ID>*, *ListCrudRepository<T,ID>*, *PagingAndSortingRepository<T,ID>*, etc.
  - *T* é o objeto de negócio (entidade) a ser gerenciada
  - *ID* é o tipo do identificador do objeto de negócio

Spring Data

```
public interface CrudRepository<T, ID
extends Serializable> extends Repository<T,
ID> {
    <S extends T> S save(S entity);
    T findOne(ID primaryKey);
    Iterable<T> findAll();
    Long count();
    void delete(T entity);
    boolean exists(ID primaryKey);
    ...
}
```

Spring Data

- Framework permite estender as interfaces de repositórios para criação automática de código
- Exemplo:

```
interface UserRepository extends
CrudRepository<User, Long> {
  long countByLastname(String lastname);
  long deleteByLastname(String lastname);
  List<User> removeByLastname(String lastname);
}
```

Spring Data

- Uma classe cliente recebe acesso à implementação do repositório através da injeção de dependências
- Exemplo:

```
class SomeClient {
  private final PersonRepository repository;
  SomeClient(PersonRepository repository) {
    this.repository = repository;
  }
  void doSomething() {
    List<Person> persons =
repository.findByLastname("Matthews");
  }
}
```

Spring Data JPA

- Projeto Spring Data JPA adiciona elementos de automatização do Spring Data sobre acesso a dados via API do JPA (Jakarta Persistence API)
  - https://jakarta.ee/specifications/persistence/
  - Novas interfaces dependentes da tecnologia, tal como *JpaRepository*
- O framework Spring Data JPA cria automaticamente as consultas utilizando a API JPA em função da interface escolhida para o repositório e a assinatura dos métodos definidos pelo programador ou consultas explícitas definidas pelo programador

Spring Data JPA

- A anotação *@EnableJpaRepositories* sobre a classe de configuração do projeto habilita o uso do framework
- Exemplo:

```
@Configuration
@EnableJpaRepositories
class ApplicationConfiguration {
   ...
}
```

Spring Data JPA

- Samples (oficiais):
  - https://github.com/spring-projects/spring-data-examples/tree/master/jpa
  - https://github.com/spring-projects/spring-data-book/tree/master/jpa