
FIAP

Domain Driven Design
using Java

AGENDA

1. Persistência de Dados com Spring Data JPA
2. Exercícios

Spring Data JPA

Prof. Evando Borges

Persistência de Dados com Spring Data JPA

- O que é Persistência de Dados?
  - Em aplicações Java, a persistência de dados refere-se à capacidade de armazenar, recuperar e manipular informações em um banco de dados.
- Por que usar Spring Data JPA?
  - Simplifica a comunicação entre a aplicação e o banco de dados.
  - Reduz o código necessário para operações CRUD (Create, Read, Update, Delete).
  - Gerencia automaticamente entidades e mapeamentos.

Arquitetura do Spring Data JPA

- Componentes principais:
  - **Entity** (@Entity) → Representa uma tabela no banco de dados.
  - **Repository** (JpaRepository) → Interface que gerencia operações no banco.
  - **Service** → Contém regras de negócio.
  - **Controller** → Exposição das APIs RESTful.
  - **Banco de Dados** → Armazena os dados e permite consultas.

Configuração do Spring Data JPA

- Dependência Maven no pom.xml:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
    <version>mais_recente</version>
</dependency>
<dependency>
    <groupId>org.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

Configuração do Spring Data JPA

- Configuração do banco de dados no application.properties:

```
spring.datasource.url=jdbc:h2:mem:meubanco
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.h2.console.enabled=true
```

Criando entidade

Prof. Evando Borges

Definição da Classe Entidade

```java
import jakarta.persistence.*;

@Entity
@Table(name = "produtos")
public class Produto {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;
    private Double preco;

    // Getters e Setters
}
```

- *@Entity* → Define que a classe será armazenada no banco.
- *@Table(name = "produtos")* → Define o nome da tabela correspondente.
- *@Id* **e** *@GeneratedValue* → Define o identificador único do objeto.

Criando repositório

Prof. Evando Borges

Definição da Interface Repository

```java
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProdutoRepository extends JpaRepository<Produto, Long> {
    List<Produto> findByNomeContaining(String nome);
}
```

- *JpaRepository<Produto, Long>* → Fornece métodos CRUD automaticamente.
- *findByNomeContaining* → Busca produtos pelo nome com filtros dinâmicos.

Criando o Serviço

Definição da Classe Service

```java
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class ProdutoService {

    private final ProdutoRepository produtoRepository;

    public ProdutoService(ProdutoRepository produtoRepository) {
        this.produtoRepository = produtoRepository;
    }

    public List<Produto> listarTodos() {
        return produtoRepository.findAll();
    }

    public Produto salvar(Produto produto) {
        return produtoRepository.save(produto);
    }

    public void deletar(Long id) {
        produtoRepository.deleteById(id);
    }
}
```

- *@Service* → Define a classe como um serviço de regras de negócio.
- Utiliza *produtoRepository* para acessar o banco de dados.

Criando o Controller

Definição da Classe Service

```java
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/produtos")
public class ProdutoController {

    private final ProdutoService produtoService;

    public ProdutoController(ProdutoService produtoService) {
        this.produtoService = produtoService;
    }

    @GetMapping
    public List<Produto> listarTodos() {
        return produtoService.listarTodos();
    }

    @PostMapping
    public Produto salvar(@RequestBody Produto produto) {
        return produtoService.salvar(produto);
    }

    @DeleteMapping("/{id}")
    public void deletar(@PathVariable Long id) {
        produtoService.deletar(id);
    }
}
```

Referências

- **Documentação Oficial:** https://spring.io/projects/spring-data-jpa.
- Livro: *Spring in Action* - Craig Walls.


FIAP