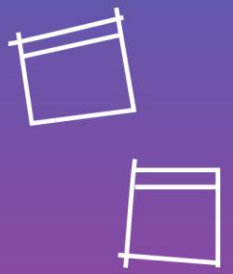


VEM SER DBC

Spring Data

Aula 2 - Relacionamentos

Sumário

- Chaves compostas e classes embutidas
- Métodos de Consulta
- Relacionamentos

@EmbbededId

- Anotação para chaves compostas
- A classe PK sempre tem que implementar Serializable

```java
@Entity(name = "PROFESSOR")
public class ProfessorEntity {

    @EmbeddedId
    private ProfessorPK professorPK;
```

```java
@Embeddable
public class ProfessorPK implements Serializable {
```

DBC
DIGITAL BUSINESS COMPANY®

show me your code;

VEM SER
DBC

Exercício #1

- Executar o Script “script_aula2.sql”
- Codificar a classe ProfessorEntity, ProfessorPK e também um ProfessorRepository
- Codificar um ProfessorController com um list e um create, testar os métodos

Query Methods

- Permite criar consultas por assinatura de métodos

```
1  public interface Produtos extends JpaRepository<Produto, Long> {
2
3    Produto findByNome(String nome);
4
5  }
```

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods

https://github.com/maiconn/vem-ser-2021/blob/main/projetos/3%20-%20spring%20boot/2%20-%20spring%20data/pessoaapi/Exemplos.java

DBC
DIGITAL BUSINESS COMPANY®

show me your code;

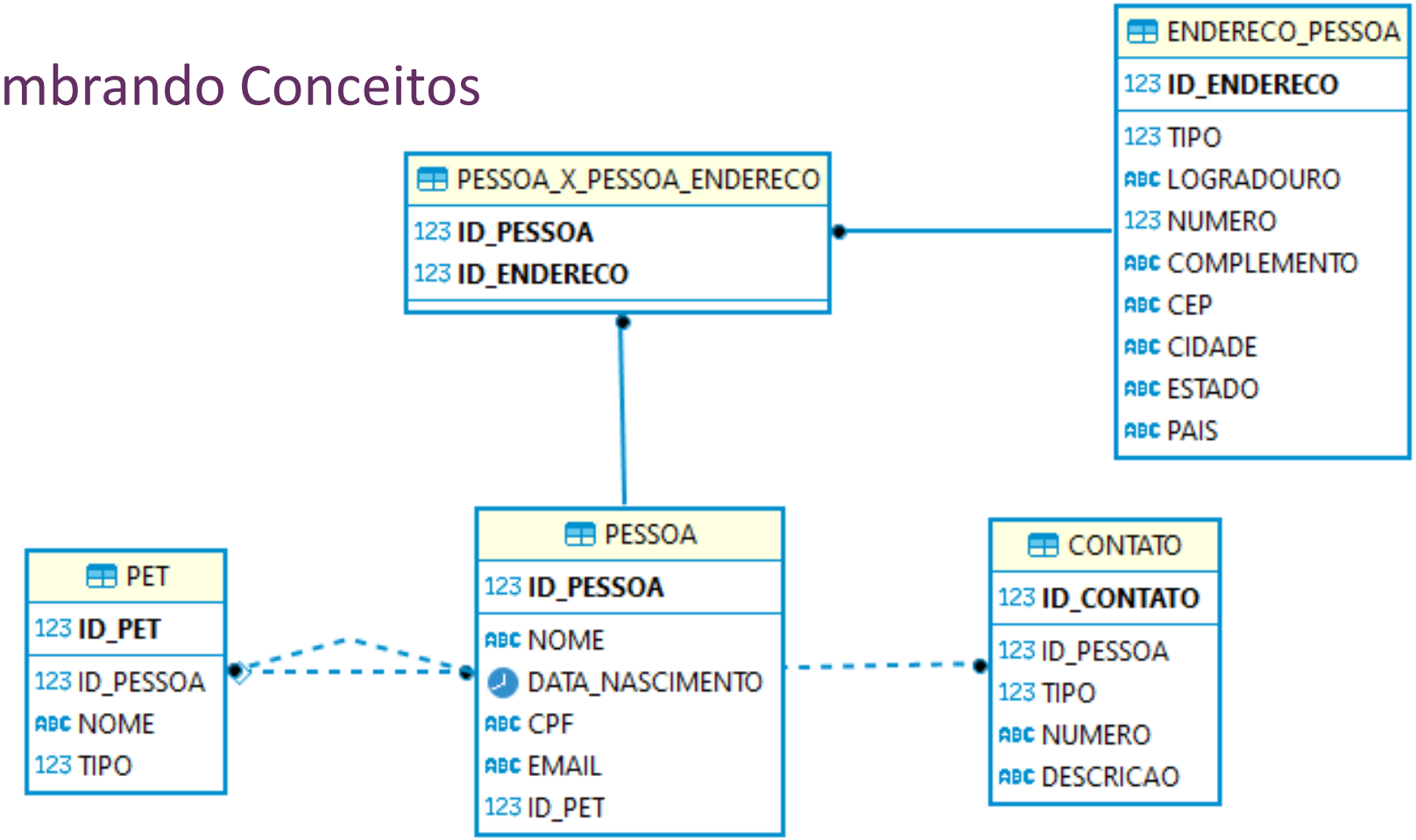
VEM SER
DBC

Exercício #2

- Criar métodos no repository para buscar pessoas
  - por nome (contains ignore case)
  - por cpf
  - por data de nascimento que está entre data inicial e final que o usuário irá informar
- Criar métodos na PessoaController para testar os métodos do repository

Relacionamentos

- Relembrando Conceitos



DBC
DIGITAL BUSINESS COMPANY®

show me your code;

VEM SER
DBC

Relacionamentos

```java
@ManyToOne
@JoinColumn(name = "id_pessoa", referencedColumnName = "id_pessoa")
private PessoaEntity pessoaEntity;
```

```java
@OneToMany(mappedBy = "pessoaEntity", cascade = CascadeType.ALL, orphanRemoval = true)
private Set<ContatoEntity> contatos;
```

```java
@OneToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "id_pet", referencedColumnName = "id_pet")
private PetEntity pet;
```

```java
@ManyToMany
@JoinTable(name = "Pessoa_X_Pessoa_Endereco",
        joinColumns = @JoinColumn(name="id_pessoa"),
        inverseJoinColumns = @JoinColumn(name="id_endereco")
)
private Set<EnderecoEntity> enderecos;
```

```java
@ManyToMany(mappedBy = "enderecos")
private Set<PessoaEntity> pessoas;
```

Exercício / Homework

- Criar todos os relacionamentos vistos em aula e utilizando os relacionamentos faça:
  - Criar Controller com todas as operações para Pet
  - Criar endpoints extras na PessoaController para:
    - /listar-com-enderecos: listar pessoas com todos os seus enderecos
    - /listar-com-contatos: listar pessoas com todos os seus contatos
    - /listar-com-pets: listar pessoas com os seus pets

  - Todos os endpoints acima devem receber o id da pessoa por query param como opcional, se não for informado listar todos, se for informado, listar somente a pessoa pelo id.

  - Reestabelecer as operações de endereços por id da pessoa no EnderecoController!

  - Importante: Implementar consultas somente na Service para Repository e utilizar o Controller para expor os endpoints

  - **NUNCA**, **JAMAIS**, devemos utilizar Repository na Controller ☺