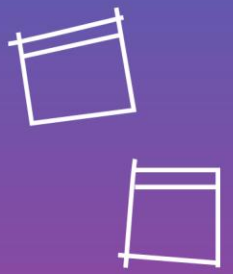
DBC
DIGITAL BUSINESS COMPANY®

VEM SER
DBC



Spring Data

Aula 3 – Consultas JPQL e SQL Nativas

Sumário

- Queries JPQL
- Exercício
- Queries Nativas
- Exercício
- Compor Query em DTO
- Homework

Consulta @Query

```
public interface UserRepository extends JpaRepository<User, Long> {

  @Query("select u from User u where u.emailAddress = ?1")
  User findByEmailAddress(String emailAddress);
}
```

- **Seu atributo de *valor* contém o JPQL (Java Persistence Query Language) a ser executado.**
- **Esse JPQL é similar ao SQL com algumas particularidades**
- **O repositório é responsável pela persistência é o melhor lugar para armazenar essas definições.**

Exercício #1

- Construir as seguintes consultas abaixo com @Query para consultar:
  - endereços por país
  - endereços por id da pessoa
  - contatos por tipo de contato
  - pessoas por data de nascimento entre duas datas
  - pessoas que possuem endereço

- **Apenas para facilitar, nesse exercício utilizar direto no controller o repository...**

Consulta @Query nativas

```
public interface UserRepository extends JpaRepository<User, Long> {

  @Query(value = "SELECT * FROM USERS WHERE EMAIL_ADDRESS = ?1", nativeQuery = true)
  User findByEmailAddress(String emailAddress);
}
```

- **Seu atributo de *valor* contém o SQL (Structured Query Language) a ser executado.**
- **O repositório é responsável pela persistência é o melhor lugar para armazenar essas definições.**

Exercício #2

- Construir as seguintes consultas abaixo com @Query nativa para consultar:
  - endereços por cidade ou país
  - endereços sem complemento
  - contatos por id da pessoa
  - pessoas que não possuem endereço

- **Apenas para facilitar, nesse exercício utilizar direto no controller o repository...**

Compor Query em DTO

```
@Query("     select  new com.dbc.vemser.dto.response.AlunoCompletoDTO(" +
        "                e.idAluno," +
        "                u.idUsuario, " +
        "                u.nome," +
        "                u.email," +
        "                e.stack, " +
        "                e.semestre, " +
        "                e.git, " +
        "                e.curso, " +
        "                e.sistemaOperacional, " +
        "                e.processador, " +
        "                e.memoriaRam, " +
        "                e.cidadeEstado, " +
        "                e.idEdicao," +
        "                edc.descricao) " +
        "         from AlunoEntity e " +
        "         join e.usuario u " +
        "    left join e.edicao edc" +
        "        where (:nome is null or upper(u.nome) like upper(concat('%',:nome,'%')))" +
        "          and (:stack is null or e.stack = :stack)" +
        "          and (:idEdicao is null or e.idEdicao = :idEdicao)")
Page<AlunoCompletoDTO> findAllWithUser(@Param("nome") String nome,
                                       @Param("stack") Stack stack,
                                       @Param("idEdicao") Integer idEdicao,
                                       Pageable pageable);
```

Homework

- Trazer a pessoa com todos os dados (endereços, contatos e pets).
  - Disponibilizar um endpoint no PessoaController “/pessoa-completo”
  - o endpoint 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.

- Faça um relatório personalizado trazendo os seguintes campos:
  - Pessoa: id, nome e email;
  - Contato: numero;
  - Endereco: cep, cidade, estado e pais.
  - Pet: nome
  - OBS: trazer pessoas mesmo sem contatos, endereços e pets.... ☺