



Spring Data

Aula 1 – Configurações e Repository

Conteúdo do Sub-módulo

- Configuração Spring Data JPA
- Relacionamento, Listas, Chaves Compostas
- Joins, HQL, Querys Nativas
- Paginação de Registros e Migração de dados Para Postgres
- Trabalho Final

Sumário

- O que é Spring Data
- Spring Data JPA
- Passos para configuração
- Dependências e Configurações
- Principais anotações
- Principais métodos JpaRepository
- Exercício / Homework

Spring Data

- O Spring Data tem por objetivo facilitar nosso trabalho com persistência de dados de uma forma geral. Ele possui vários outros projetos:

<https://blog.algaworks.com/spring-data-jpa/>

Spring Data

- O Spring Data tem por objetivo facilitar nosso trabalho com persistência de dados de uma forma geral. Ele possui vários outros projetos:
 - Spring Data Commons
 - Spring Data Gemfire
 - Spring Data KeyValue
 - Spring Data LDAP
 - Spring Data MongoDB
 - Spring Data REST
 - Spring Data Redis
 - Spring Data for Apache Cassandra
 - Spring Data JPA

<https://blog.algaworks.com/spring-data-jpa/>

Spring Data

- O Spring Data tem por objetivo facilitar nosso trabalho com persistência de dados de uma forma geral. Ele possui vários outros projetos:
 - Spring Data Commons
 - Spring Data Gemfire
 - Spring Data KeyValue
 - Spring Data LDAP
 - Spring Data MongoDB
 - Spring Data REST
 - Spring Data Redis
 - Spring Data for Apache Cassandra
 - **Spring Data JPA**

<https://blog.algaworks.com/spring-data-jpa/>

O que é Spring Data JPA?

- O Spring Data JPA é um framework que nasceu para facilitar a criação dos nossos repositórios

<https://blog.algaworks.com/spring-data-jpa/>

O que é Spring Data JPA?

- O Spring Data JPA é um framework que nasceu para facilitar a criação dos nossos repositórios
- Ele faz isso nos liberando de ter que implementar as interfaces referentes aos nossos repositórios (ou DAOs)

<https://blog.algaworks.com/spring-data-jpa/>

O que é Spring Data JPA?

- O Spring Data JPA é um framework que nasceu para facilitar a criação dos nossos repositórios
- Ele faz isso nos liberando de ter que implementar as interfaces referentes aos nossos repositórios (ou DAOs)
- Já deixa pré-implementado algumas funcionalidades como, por exemplo, de ordenação das consultas e de paginação de registros

<https://blog.algaworks.com/spring-data-jpa/>

Passos para configuração

- Colocar a dependência do spring jpa e do driver do banco de dados
- Configurar parâmetros do banco de dados
- Configurar entidades
- Configurar repositórios

Wualá

Dependências

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

```
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc8</artifactId>
  <scope>runtime</scope>
</dependency>
```

Configurações

```
# Oracle settings
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=oracle
spring.datasource.driverClassName=oracle.jdbc.driver.OracleDriver
spring.jpa.database-platform=org.hibernate.dialect.Oracle10gDialect
spring.jpa.properties.hibernate.default_schema=VEM_SER
```

Configurações Extras

```
# create and drop tables and sequences, loads import.sql
spring.jpa.hibernate.ddl-auto=create-drop
# none, validate, update, create-drop

spring.jpa.show-sql=true
log4j.logger.org.hibernate.type=trace
spring.jpa.properties.hibernate.format_sql=true
```

show me your code;

Anotações de Entidades

```
@Entity(name = "PESSOA")
public class PessoaEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "PESSOA_SEQ")
    @SequenceGenerator(name = "PESSOA_SEQ", sequenceName = "seq_pessoa2", allocationSize = 1)
    @Column(name = "id_pessoa")
    private Integer idPessoa;

    @Column(name = "nome")
    private String nome;
}
```

JpaRepository

- Ela tem todos os métodos que a gente precisa para fazer um CRUD (criar, ler, atualizar, deletar).

```
@Repository
public interface PessoaRepository extends JpaRepository<PessoaEntity, Integer> {
```


Principais Métodos do JPA Repository

```

Inherited members (Ctrl+F12) Anonymous Classes (Ctrl+I) Lambdas (Ctrl+L)
JpaRepository
count(): long → CrudRepository
count(Example<S>): long → QueryByExampleExecutor
delete(T): void → CrudRepository
deleteAll(): void → CrudRepository
deleteAll(Iterable<? extends T>): void → CrudRepository
deleteAllById(Iterable<? extends ID>): void → CrudRepository
deleteAllByIdInBatch(Iterable<ID>): void
deleteAllInBatch(): void
deleteAllInBatch(Iterable<T>): void
deleteById(ID): void → CrudRepository
deleteInBatch(Iterable<T>): void
exists(Example<S>): boolean → QueryByExampleExecutor
existsById(ID): boolean → CrudRepository
findAll(): List<T> → CrudRepository
findAll(Example<S>): Iterable<S> → QueryByExampleExecutor
findAll(Example<S>, Pageable): Page<S> → QueryByExampleExecutor
findAll(Example<S>, Sort): Iterable<S> → QueryByExampleExecutor
findAll(Example<S>): List<S> → QueryByExampleExecutor
findAll(Example<S>, Sort): List<S> → QueryByExampleExecutor
findAll(Pageable): Page<T> → PagingAndSortingRepository
findAll(Sort): List<T> → PagingAndSortingRepository
findAllById(Iterable<ID>): Iterable<T> → CrudRepository
findAllById(Iterable<ID>): List<T> → CrudRepository
findById(ID): Optional<T> → CrudRepository
findOne(Example<S>): Optional<S> → QueryByExampleExecutor
flush(): void
getById(ID): T
getOne(ID): T
save(S): S → CrudRepository
saveAll(Iterable<S>): Iterable<S> → CrudRepository
saveAll(Iterable<S>): List<S> → CrudRepository
saveAllAndFlush(Iterable<S>): List<S>
saveAndFlush(S): S

```

Exercício #1

- Criar uma pasta no seu repositório “modulo 3.2”, copiar o seu pessoaapi do módulo anterior para dentro dessa pasta
- Executar o script “script_aula1.sql” no seu banco de dados
- Configurar o Spring JPA no seu projeto pessoaapi
- Configurar o PessoaEntity com as anotações corretas
- Configurar o PessoaRepository com o JpaRepository
- Ajustar o projeto para funcionar com os métodos do JpaRepository

Exercício #2

- Configurar o ContatoEntity com as anotações corretas
- Configurar o ContatoRepository com o JpaRepository
- Ajustar o projeto para funcionar com os métodos do JpaRepository

Homework

- Na pasta “modulo 3.2” no repositório do git, faça:
- Configurar o EnderecoEntity com as anotações corretas (conforme a tabela ENDERECO_PESSOA)
- Configurar o EnderecoRepository com o JPARepository
- Ajustar o projeto para funcionar com os métodos do JPARepository