

# VEM SER

## **Módulo 3.2 - Spring Data**

### Aula 01 - Fundamentos

# Conteúdo da aula

- **ORM**
- **JPA**
- **Como funciona**
- **Spring Data**
- **Spring Data JPA**
- **JpaRepository**

# ORM

Mapeamento objeto-relacional (*Object-Relational Mapper*) é uma técnica que conecta POO com bancos de dados relacionais, fazendo a **ponte** entre classes/objetos e tabelas/registros.

POO	Modelo relacional
Classes	Tabelas
Objetos	Registros
Atributo	Coluna
Associação	Chave estrangeira



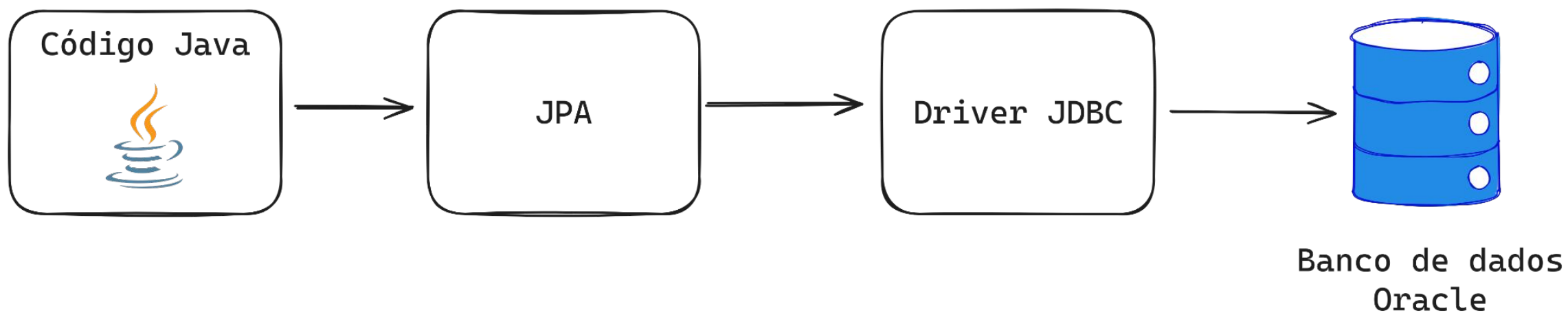
# JPA

**Java Persistence API** é uma solução para mapeamento objeto-relacional, transformando objetos Java em entidades para que o banco de dados consiga armazenar os registros.

# Vantagens do JPA

- **Abstração:** o JPA abstrai os detalhes mais específicos da implementação, como SQL e APIs específicas, permitindo que o desenvolvedor **foque na lógica da aplicação**;
- **Portabilidade:** o JPA fornece suporte a grande maioria de bancos de dados, como Oracle, MySQL, PostgreSQL, etc;
- **Anotações:** utilize as anotações para definir entidades, mapear e fornecer as informações necessárias para a persistência dos dados.

# Como funciona



# Spring Data

O **Spring Data** é um “projeto guarda-chuva” que fornece um modelo para implementar diversas formas de armazenamento utilizando o *framework* Spring.

Os principais modelos são:

- Spring Data JPA;
- Spring Data JDBC;
- Spring Data MongoDB;
- Spring Data Redis;
- Spring Data DynamoDB;
- etc.

Link: <https://spring.io/projects/spring-data>

# Spring Data JPA

O **Spring Data JPA** é uma das principais formas de trabalhar com o Spring e bancos de dados relacionais, baseando-se na JPA (Java Persistence API).

É uma forma fácil de implementar a camada entre aplicação e armazenamento, através dos **repositórios**.

Não precisamos nos preocupar com DAOs e já temos algumas funcionalidades implementadas como ordenação das consultas e paginação dos registros.

Documentação: <https://docs.spring.io/spring-data/jpa/docs/2.5.6/reference/html>



# Passo a passo da configuração

1. Adicionar a dependência do Spring JPA e do *driver* do banco de dados no ***pom.xml***;
2. Adicionar os parâmetros de conexão com o banco de dados (via `application.properties` / VM args);
3. Configurar entidades;
4. Configurar os *repositories*.

# 1. 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>
```

## 2. Parâmetros de conexão

*# 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
```




**Prática!**

## 3. Entidades

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

Nome da tabela



# Atributos

```
@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;
```

## 4. Repositórios

A interface que herdaremos se chama **JpaRepository** e ela tem os métodos CRUD básicos (criar, ler, atualizar e excluir).

@Repository

```
public interface PessoaRepository extends JpaRepository<PessoaEntity, Integer>
```



# Métodos principais do JpaRepository

- **delete;**
- **deleteById;**
- **findAll;**
- **findById;**
- **save.**

# Exercício #1

- Criar uma pasta no seu repositório chamada **modulo-03-2-springdata**, copiar o seu **pessoa-api** do módulo anterior para dentro dessa pasta;
- Executar o script **script-aula-01.sql** no seu banco de dados;
- Configurar o Spring JPA no seu projeto pessoa-api;
- Configurar o PessoaEntity com as anotações corretas;
- Configurar o PessoaRepository com o JPARepository;
- Ajustar o projeto para funcionar com os métodos do JPARepository.

# Task #1 (obrigatório)

- Configurar o ContatoEntity com as anotações corretas;
- Configurar o ContatoRepository com o JpaRepository;
- Configurar o EnderecoEntity com as anotações corretas (conforme a tabela ENDERECO\_PESSOA) (inserir somente endereço sem a pessoa por enquanto, ignorar id da pessoa);
- Configurar o EnderecoRepository com o JpaRepository;
- Ajustar o projeto para funcionar com os métodos do JpaRepository.

## Task #2 Grupo (opcional)

- Configurar o Spring Data no projeto do seu grupo;
- Começar o mapeamento das entidades no projeto;

# Referências

[https://docs.oracle.com/cloud/help/pt\\_BR/analytics-cloud/ACSDS/GUID-FB2AEC3B-2178-48DF-8B9F-76ED2D6B5194.htm#ACSDS-GUID-FB2AEC3B-2178-48DF-8B9F-76ED2D6B5194](https://docs.oracle.com/cloud/help/pt_BR/analytics-cloud/ACSDS/GUID-FB2AEC3B-2178-48DF-8B9F-76ED2D6B5194.htm#ACSDS-GUID-FB2AEC3B-2178-48DF-8B9F-76ED2D6B5194)

<https://spring.io/projects/spring-data>

<https://docs.spring.io/spring-data/jpa/docs/2.5.6/reference/html>



Let's *Tech Up Together*