**Aula JPA Maven – Mapeamento Objeto-Relacional**

**Contextualização**

**Quando se trabalha com os recursos do JDBC para resgate e manipulação de dados, passa-se por uma gincana para fazer o mapeamento objeto relacional manualmente. Para mudar esse paradigma, vem aí a especificação JPA (Java Persistence API) implementada pelo Hibernate, uma união de tecnologias que promete executar o processo mapeamento objeto-relacional automaticamente.**

**Programando o Sistema**

1. **Configure o pom.xml para que o sistema maven do projeto importe as dependências necessárias. São elas: o Hibernate Core, Hibernate EntityManager e SQLServer conector.**

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>7.0.0.Final</version>

<type>pom</type>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>5.6.15.Final</version>

</dependency>

<dependency>

<groupId>com.microsoft.sqlserver</groupId>

<artifactId>mssql-jdbc</artifactId>

<version>12.6.1.jre11</version>

</dependency>

</dependencies>

1. **Dentro das pastas src/main/resources, crie a pasta META-INF para guardar o arquivo persistence.xml com as configurações do JPA para conexão com o banco.**

<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence

http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd" version="2.1">

<persistence-unit name="exemplo-jpa" transaction-type="RESOURCE\_LOCAL">

<properties>

<property name="javax.persistence.jdbc.url"

value="jdbc:sqlserver://restdb.database.windows.net:1433;database=RestaurantDatabase;encrypt=true;trustServerCertificate=false;loginTimeout=30;" />

<property name="javax.persistence.jdbc.driver"

value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />

<property name="javax.persistence.jdbc.user" value="boss" />

<property name="javax.persistence.jdbc.password" value="restaurantSystem123" />

<property name="hibernate.hbm2ddl.auto" value="update" />

<property name="hibernate.dialect" value="org.hibernate.dialect.SQLServerDialect" />

</properties>

</persistence-unit>

</persistence>

1. **Crie a classe Pessoa que será o objeto associado ao corpo dos registros do mesmo no banco de dados.**

package dominio;

import java.io.Serializable;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

*@Entity*

public class Pessoa implements Serializable{

private static final long ***serialVersionUID*** = 1L;

*@Id*

*@GeneratedValue*(strategy=*GenerationType*.***IDENTITY***)

private Integer id;

private String nome;

private String email;

//-----------------Construtores--------------------

//-----------------Getters and Setters------------

}

1. **Crie um EntityManagerFactory para gerar uma instância do objeto EntityManager, necessário para interagir com o banco de dados.**

EntityManagerFactory emf = Persistence.*createEntityManagerFactory*("exemplo-jpa");

EntityManager em = emf.createEntityManager();

1. **Explore as diversas possibilidades de manipulação de dados que o EntityManager é capaz de oferecer, como o comando persist (inserir dados), remove (deletar dados) e refresh (atualizar dados).**

package aplicacao;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

import dominio.Pessoa;

public class Program {

public static void main(String[] args) {

EntityManagerFactory emf = Persistence.createEntityManagerFactory("exemplo-jpa");

EntityManager em = emf.createEntityManager();

//Inserindo objetos no banco de dados

Pessoa pessoa = new Pessoa(null, "Azul", "azul@gmail.com");

em.getTransaction().begin();

em.persist(pessoa);

em.getTransaction().commit();

Pessoa p = em.find(Pessoa.class, 2);

em.getTransaction().begin();

em.remove(p);

em.getTransaction().commit();

System.out.println("Pronto!");

em.close();

emf.close();

}

}