Curso de Jogos Digitais

Disciplina de Tecnologias Web

Aula 16

Implementação do Back-end

Professor: André Flores dos Santos

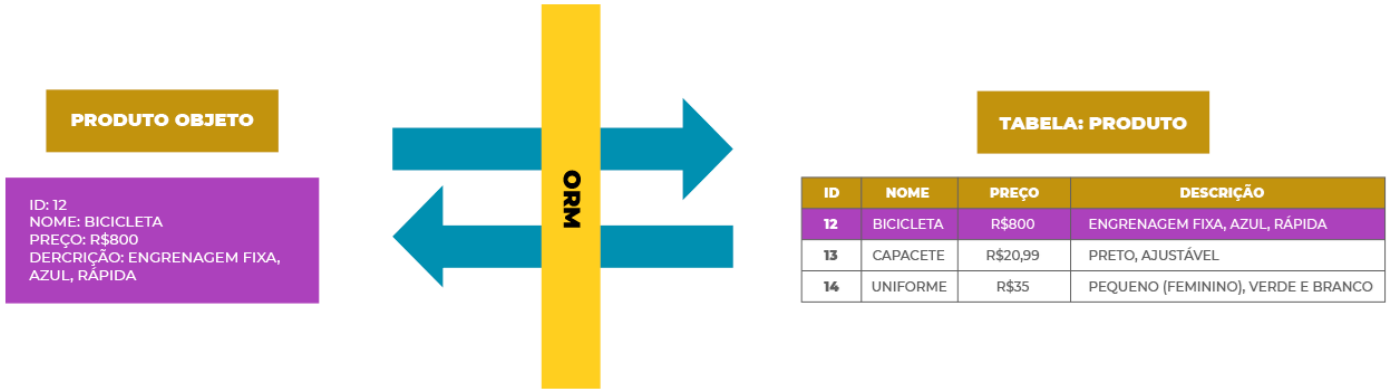
Santa Maria – RS 2025

Introdução ao ORM (Object Relational Mapper) ou (Mapeamento de Objeto Relacional)

ORM (Object Relational Mapper) é uma técnica de mapeamento objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam. Essa ferramenta vem crescendo bastante nos últimos anos.

Este crescimento tem se dado principalmente pelo fato de muitos desenvolvedores não se sentirem a vontade em escrever código SQL e pela produtividade que esta técnica nos proporciona. Existem ótimos ORM´s como Hibernate, NHibernate, Entity Framework e etc.

Introdução ao ORM (Object Relational Mapper) ou (Mapeamento de Objeto Relacional)



Introdução ao ORM (Object Relational Mapper) ou (Mapeamento de Objeto Relacional)

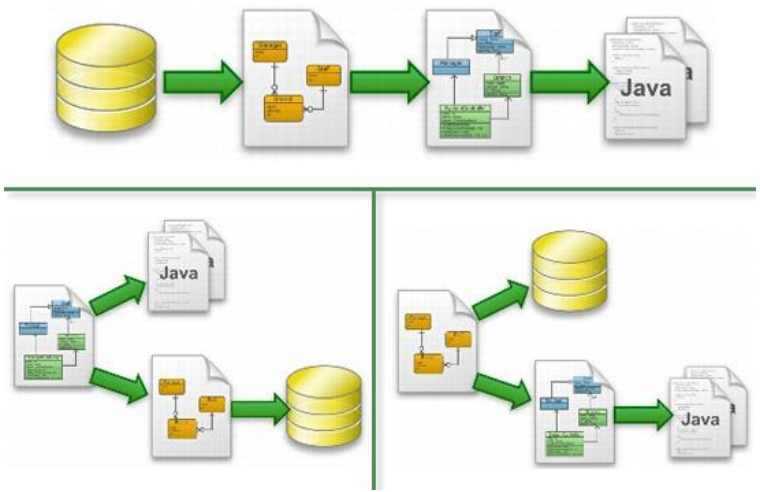
Existem dois mundos: o relacional e o orientado a objetos.

No mundo relacional prevalecem princípios matemáticos com a finalidade de armazenar e gerenciar corretamente os dados, de forma segura e se trabalha com a linguagem SQL que é utilizada para dizer o banco de dados “O QUE?” fazer e não como fazer.

Já no mundo orientado a objetos trabalhamos com classes e métodos, ou seja, trabalhamos fundamentados na engenharia de software e seus princípios que nos dizem “COMO” fazer. O ORM é justamente, a ponte entre estes dois mundos, ou seja, é ele quem vai permitir que você armazene os seus objetos no banco de dados.

Para isto precisamos fazer um mapeamento dos seus objetos para as tabelas do banco de dados.

Introdução ao ORM (Object Relational Mapper) ou (Mapeamento de Objeto Relacional)



Introdução ao ORM (Object Relational Mapper) ou (Mapeamento de Objeto Relacional)

A imagem anterior nos traz uma ideia de como o ORM trabalha. Ele faz o mapeamento de uma classe para o banco de dados, e cada ORM tem suas particularidades para gerar os comandos SQL referente a manipulação do objeto junto ao banco de dados.

Utilizando um ORM, também se ganha produtividade, pois deixa-se de escrever os comandos SQL para deixar que o próprio ORM, faça isto por você.

O que é ORM

É uma técnica de desenvolvimento utilizada para reduzir o atrito da OO utilizada com bancos de dados relacionais. Sua “mágica” é trabalhar com mapeamentos. Vamos entendê-lo através de um exemplo:

Você tem uma classe Cliente e uma tabela no seu banco de dados tb_Cliente. Para você ligar ambos você usa uma técnica de mapeamento padrão do ORM chamada Data Mapper.

Que nada mais é que o mapeamento da sua classe com seus respectivos atributos e como ele vai ser inserido dentro do banco de dados. Você usa mecanismos como XML, JSON ou *Anotations* para fazer o mapeamento entre objeto e tabela, entretanto usaremos *Anotations*.

No JPA para cada atributo de uma classe que corresponde um atributo da tabela você usa uma anotação sendo bem semelhante às anotações*: @Override* e *@Overload*.

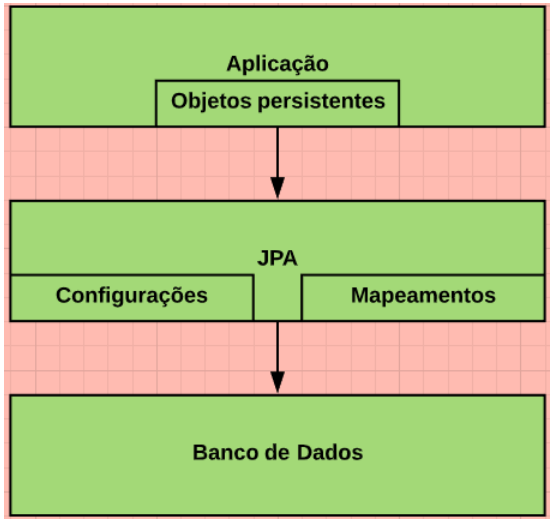
O que é ORM

Neste exemplo acima a tabela tb_cliente foi referenciada com a classe Cliente e para criar essa referência são utilizadas as anotações. O Hibernate vai gerar os códigos SQL e a partir disso o JPA irá utilizar as querys e criará a relação com a definição da tabela.

JPA (Java Persitence API)

JPA ou Java Persistence API é uma camada que descreve uma interface comum para frameworks ORM, sendo a especificação padrão da plataforma Java EE (Enterprise Edition) para mapeamento objeto-relacional e persistência de dados. Para trabalhar com JPA é preciso incluir no projeto uma implementação, como o Hibernate.
Exemplo: Você tem seus objetos persistentes, tal como os de Cliente e quer persistir seus atributos no banco, a JPA, fará o intermédio.

Os mapeamentos se dão através de *@anotations* que colocamos nas classes. Já as configurações são feitas em um arquivo chamado persistence.xml e nele colocamos coisas básicas como a url do banco, nome e senha do usuário do SGBD (Sistema Gerenciador de Banco de Dados), dentre outras.



Exemplo do persistence.xml

```xml
<?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://xml
    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:mysql://localhost/aulajpa?useSSL=false&amp;serverTimezone=UTC" />
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="" />
            <property name="hibernate.hbm2ddl.auto" value="update" />
            <!-- https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/dialect/package-summary.html -->
            <!--MariaDB103Dialect-->
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL8Dialect" />

        </properties>
    </persistence-unit>

</persistence>
```

HIBERNATE

É um framework ORM, sendo a implementação física do que se usará para persistir, remover, atualizar ou buscar dados do banco. Ele que vai seguir as especificações do JPA no seu projeto, inclusive é responsável por todo trabalho de baixa serialização de dados como, por exemplo, a criação de uma query a ser consumida pelo JPA.

Então, basicamente o Hibernate, seguindo as especificações do JPA, irá fazer a conversão de objetos em entidades baseadas em SQL, ou seja, ele faz todo o trabalho de baixo nível para converter dados entre paradigmas ER e OO.

HIBERNATE x JPA

O JPA é apenas uma interface que define mapeamentos das classes e algumas configurações e o Hibernate apenas a classe que vai implementar essa interface.

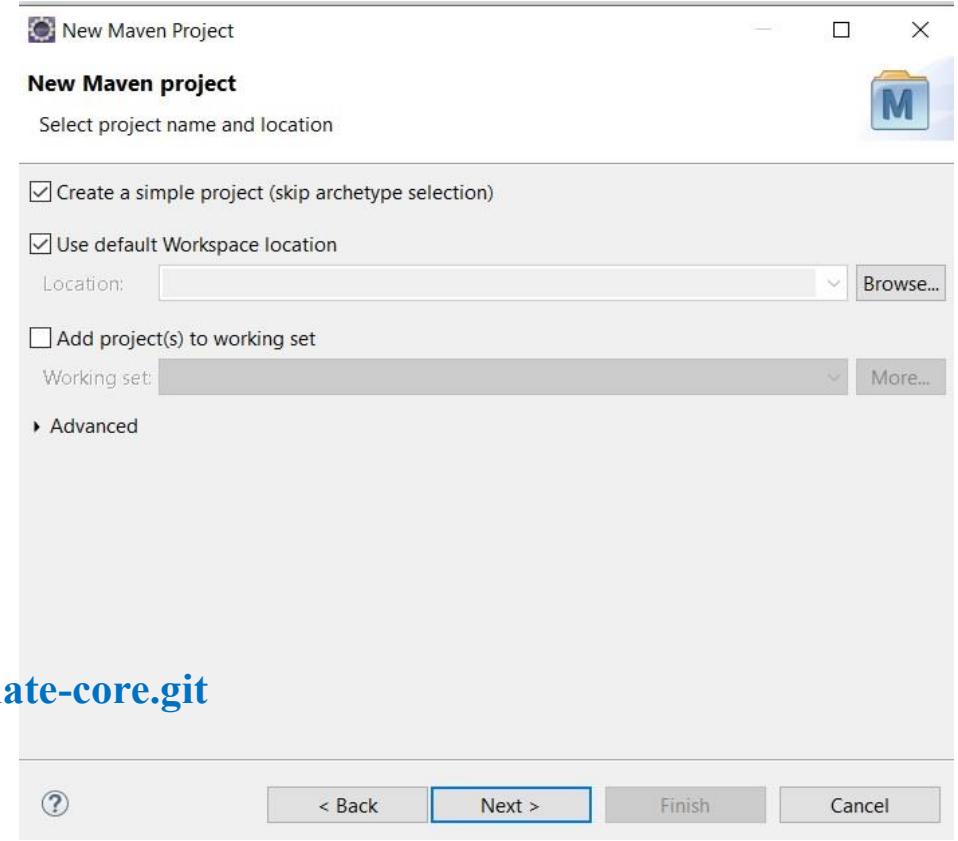
Sabendo disso, você já consegue entender que o JPA por ser uma interface nunca atua sozinho, enquanto o Hibernate por ser independente pode atuar com outras ferramentas do ecossistema do Java.

Prática

Criar um novo projeto maven.

File->New->Other->Maven Project

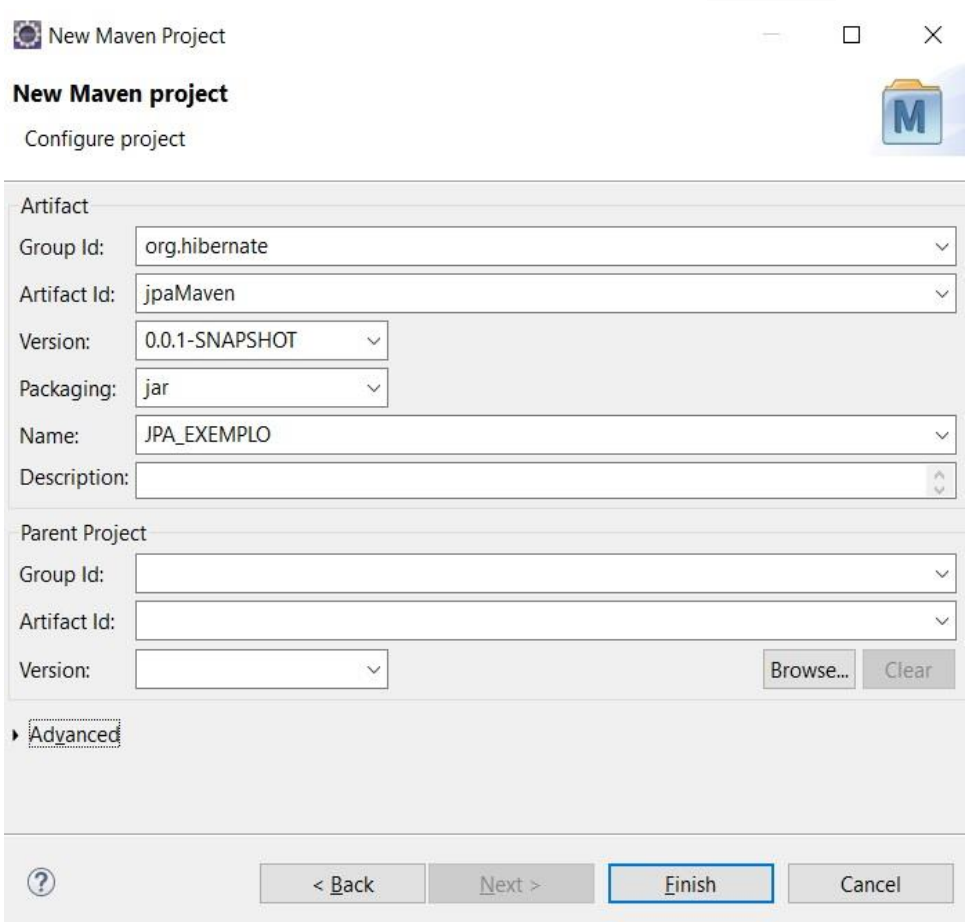
Marcar “Create a simple project



Link do projeto:
https://github.com/andreflores2009/hibernate-core.git

Prática

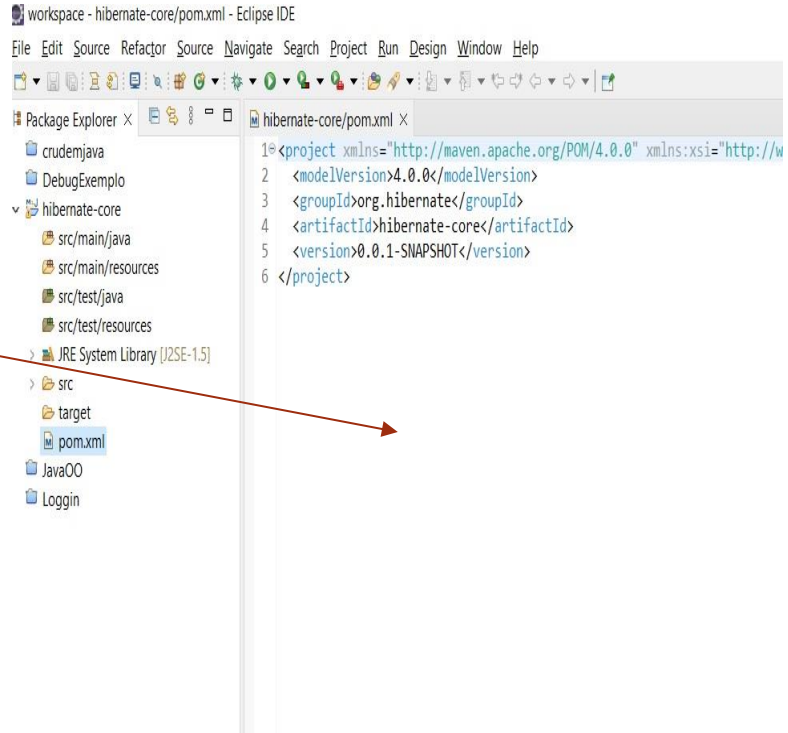
Group Id: (nome do pacote) =
org.hibernate
Artifact Id:
jpaMaven



Editando o arquivo POM.xml

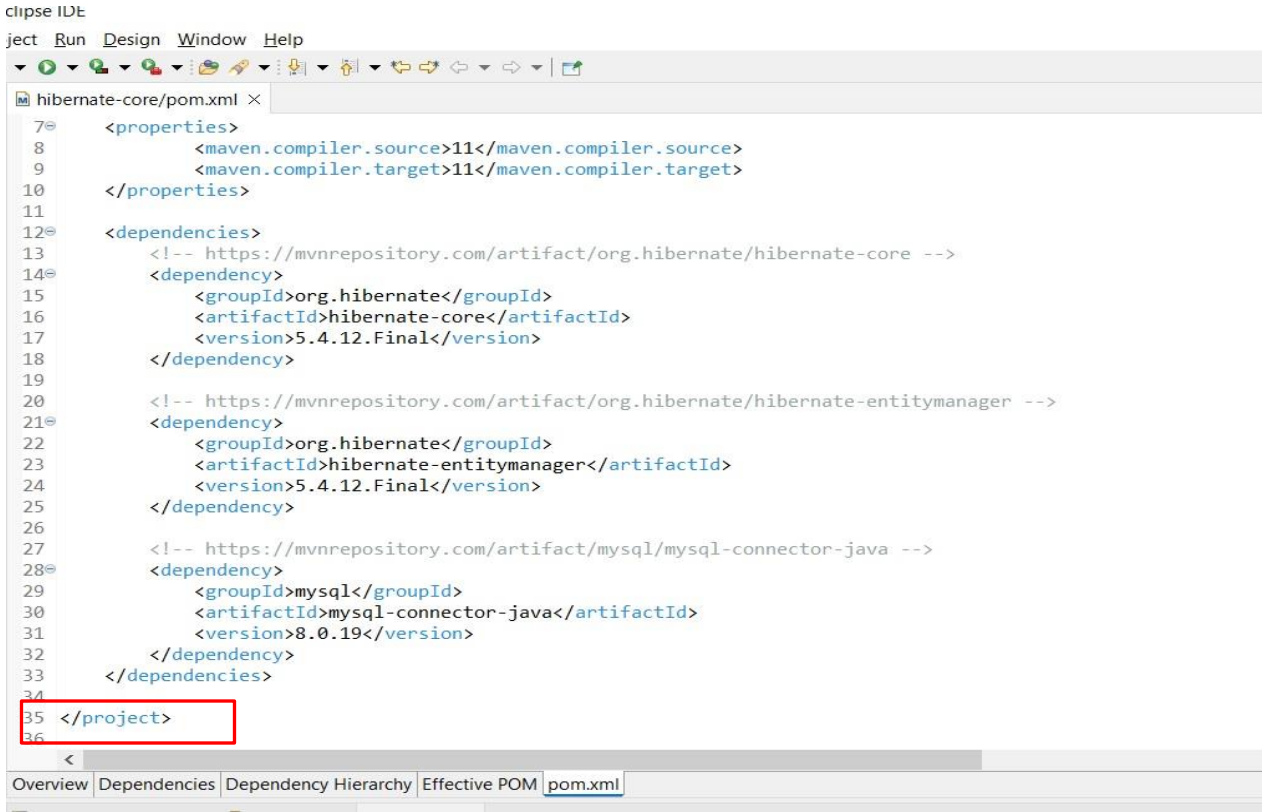
Atualizar o Maven para JAVA versão 11
Adicionar dependências do projeto

```
<properties>
		<maven.compiler.source>11</maven.compiler.source>
		<maven.compiler.target>11</maven.compiler.target>
</properties>

<dependencies>
	<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>5.4.12.Final</version>
	</dependency>

	<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-entitymanager</artifactId>
		<version>5.4.12.Final</version>
	</dependency>

	<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.19</version>
	</dependency>
</dependencies>
```



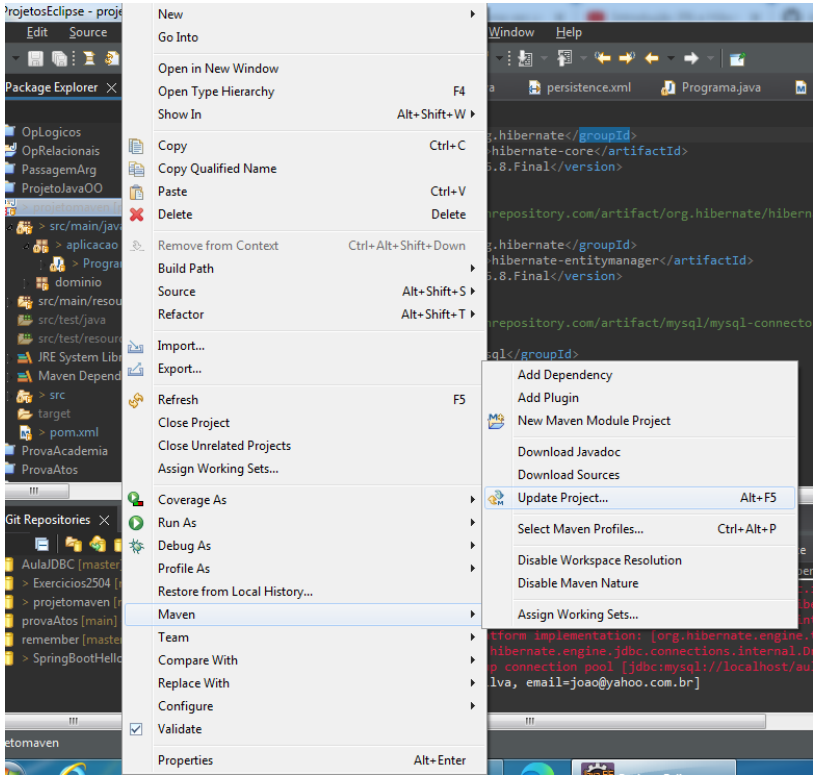
Editando o arquivo POM.xml

Atualizar o Maven para JAVA versão 11
Adicionar dependências do projeto



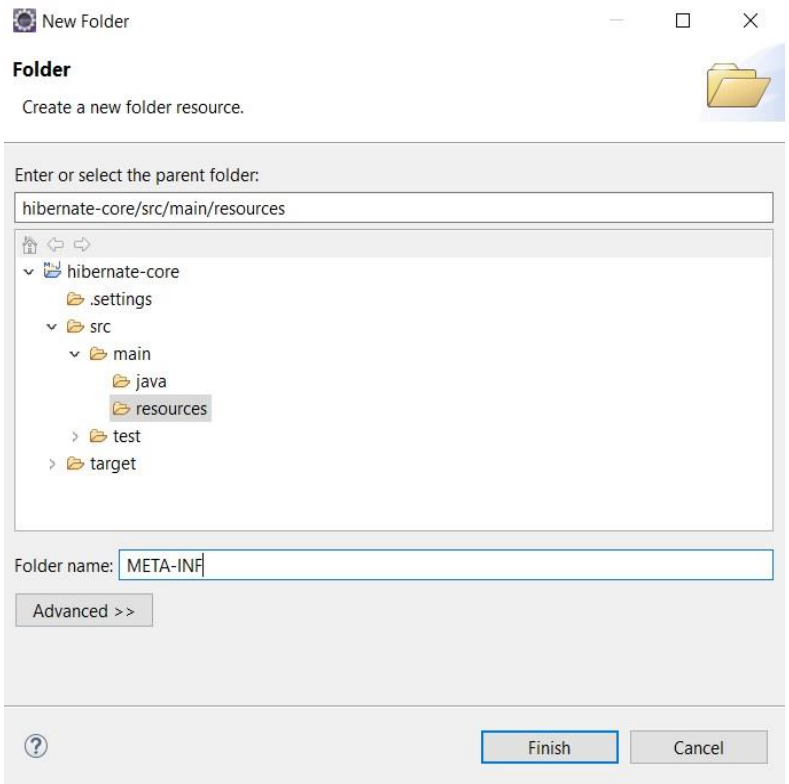
Editando o arquivo POM.xml

Após, clique com o botão direito do mouse sobre o seu Projeto:
Maven-> Update Project (Ou Alt + F5)



Configurando o JPA - persistence.xml

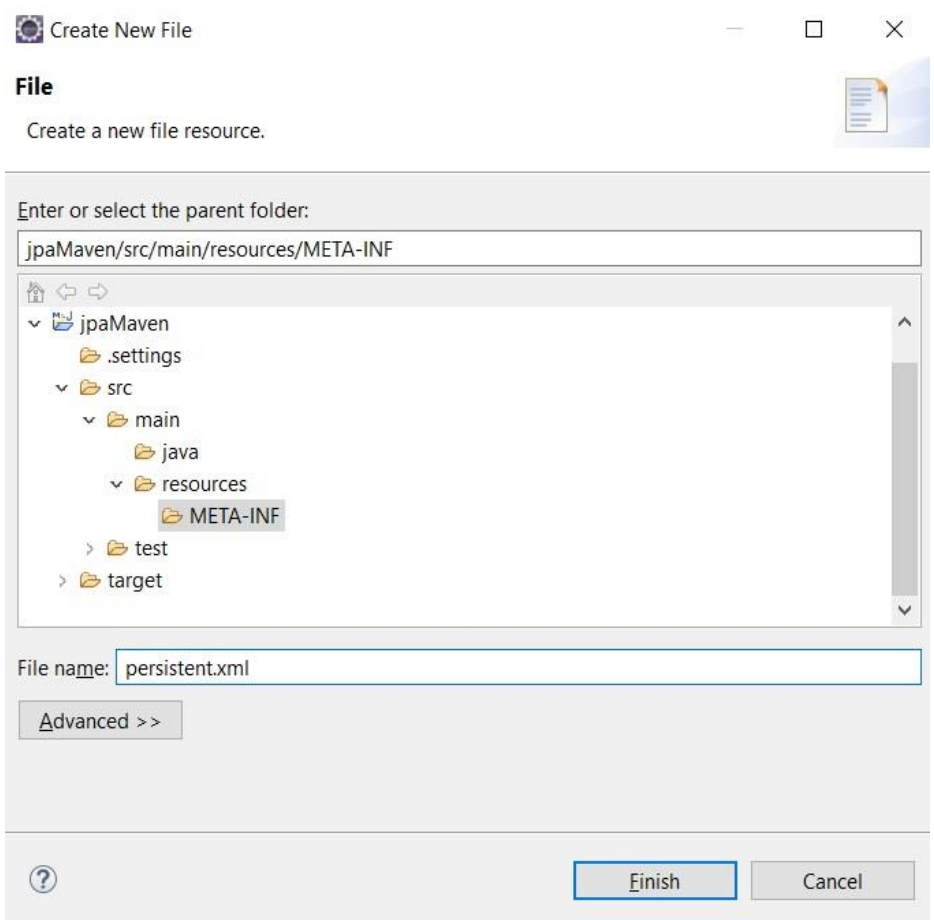
Crie uma pasta "META-INF" a partir da pasta "resources"
Dentro da pasta META-INF crie um arquivo "persistence.xml"



Configurando o JPA - persistence.xml

Crie uma pasta "META-INF" a partir da pasta "resources"

Dentro da pasta META-INF crie um arquivo "persistence.xml"



Configurando o JPA - persistence.xml

Crie uma pasta "META-INF" a partir da pasta "resources"
Dentro da pasta META-INF crie um arquivo "persistence.xml"
Conteúdo do arquivo persistence.xml:

```
<?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:mysql://localhost/exemplo-jpa?useSSL=false&amp;serverTimezone=UTC" />

          <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
          <property name="javax.persistence.jdbc.user" value="root" />
          <property name="javax.persistence.jdbc.password" value="" />

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

          <!-- https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/dialect/package-summary.html -->
          <property name="hibernate.dialect"    value="org.hibernate.dialect.MySQL8Dialect" />
     </properties>
     </persistence-unit>
  </persistence>
```

Configurando o JPA - persistence.xml

*persistence.xml

```xml
<?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:mysql://localhost/exemplo-jpa?useSSL=false&amp;serverTimezone=UTC" />

        <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
        <property name="javax.persistence.jdbc.user" value="root" />
        <property name="javax.persistence.jdbc.password" value="" />

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

        <!-- https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/dialect/package-summary.html -->
        <property name="hibernate.dialect"  value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
    </persistence-unit>
</persistence>
```

Configurando os Mapeamentos

```
@Entity
public class Pessoa implements Serializable {
   private static final long serialVersionUID = 1L;

   @Id
   @GeneratedValue(strategy=GenerationType.IDENTITY)
   private Integer id;
   (...)
```

Configurando os Mapeamentos

persistence.xml | Pessoa.java | Principal.java

```java
import javax.persistence.OneToOne;

//define a entidade que será a tabela no banco de dados
@Entity
public class Pessoa implements Serializable {
    /*
    Serializable = Ela dá capacidade da classe produzir um formato em que os dados do objeto sejam usados de forma externa ao
    código,
    em geral ele é persistido em alguma forma de armazenamento temporário ou permanente ou é transmitido para outro recurso*/


    private static final long serialVersionUID = 1L;
    //@id identificação de chave primária
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    /*Anotação @GeneratedValue, a qual deve ser declarada quando a geração do
     * valor da chave-primária é de responsabilidade do banco de dados.
     */
    Integer id;
    String nome;
    String email;
    String cargo;

    /*
     * O super() serve para chamar o construtor da superclasse. Ele sempre é chamado, mesmo quando não está explícito no código,
     *  quando for explicitado deve ser o primeiro item dentro do construtor. */
```

Criar as classes

Criar uma Classe Pessoa() e uma classe Principal() dentro da pasta ‘src/main/java’

Criando o Schema ou dB no Mysql

Nome: exemplo-jpa ou o mesmo nome que forem definidos no persistence.xml devem ser a tabela que existe no banco de dados

*persistence.xml

```xml
<?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:mysql://localhost/exemplo-jpa?useSSL=false&amp;serverTime
        <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Dri
        <property name="javax.persistence.jdbc.user" value="root" />
        <property name="javax.persistence.jdbc.password" value="" />

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

        <!-- https://docs.jboss.org/hibernate/orm/5.4/javadocs/org/hibernate/dialect/package-summary.html -->
        <property name="hibernate.dialect"  value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
    </persistence-unit>
</persistence>
```

Dicas para testar o Projeto

Executar a Classe definida como ‘main’ e testar operações direto no banco de dados.

Faremos juntos.

Referências

HOGAN, Brian P. HTML 5 e CSS3: Desenvolva hoje com o padrão de amanhã . Rio de Janeiro (RJ): Ciência Moderna Ltda., 2012 282 p. ISBN 978-85-399-0260-6

Fábio Flatschart. HTML 5 - Embarque Imediato, 2011. (Biblioteca Digital)

Deitel, Paul J.; Deitel, Harvey M.. Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores, 2008. (Biblioteca Digital)

SILVA, Maurício Samy. HTML 5: a linguagem de marcação que revolucionou a web. 2. ed. São Paulo, SP: Novatec, 2014. 335 p. ISBN 978-85-7522-403-8.

Denilson Bonatti. Desenvolvimento de Jogos em HTML5, 2014. (Biblioteca Digital)

W3SCHOOL. The world's largest web development site. Acessado em 2018. Disponível em: http://www.w3schools.com/.

W3C. World Wide Web Consortium. Acessado em 2018. Disponível em: http://www.w3.org

MORRISON, Michael. Use a cabeça JavaScript. Rio de Janeiro (RJ): Alta Books, 2008. 606 p.

https://br.freepik.com/fotos-vetores-gratis/desenvolvimento-web

Material do Professor Fabrício Tonetto Londero, 2023.

Obrigado pela atenção!!

Email: **andre.flores@ufn.edu.br**

Santa Maria – RS
2025