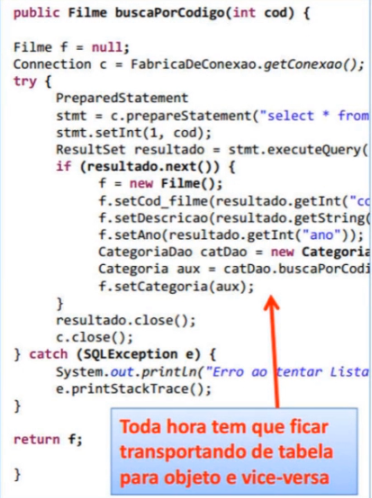
Mapeamento objeto relacional (JPA – Java Persistence API): veio para evitar várias linhas de código Java para transformar o que vem do banco de dados em objeto para poder ser tratado. Ex:



A JPA fica entre os objetos e o banco de dados, primeiro mapeando seus objetos e configurando o JPA.

EntityManager: instanciando esta classe, é fazer a conexão com o BD.

Quando o usuário faz uma requisição na Web significa que a aplicação vai instanciar um único EntityManager pra trabalhar com todos os dados da requisição.

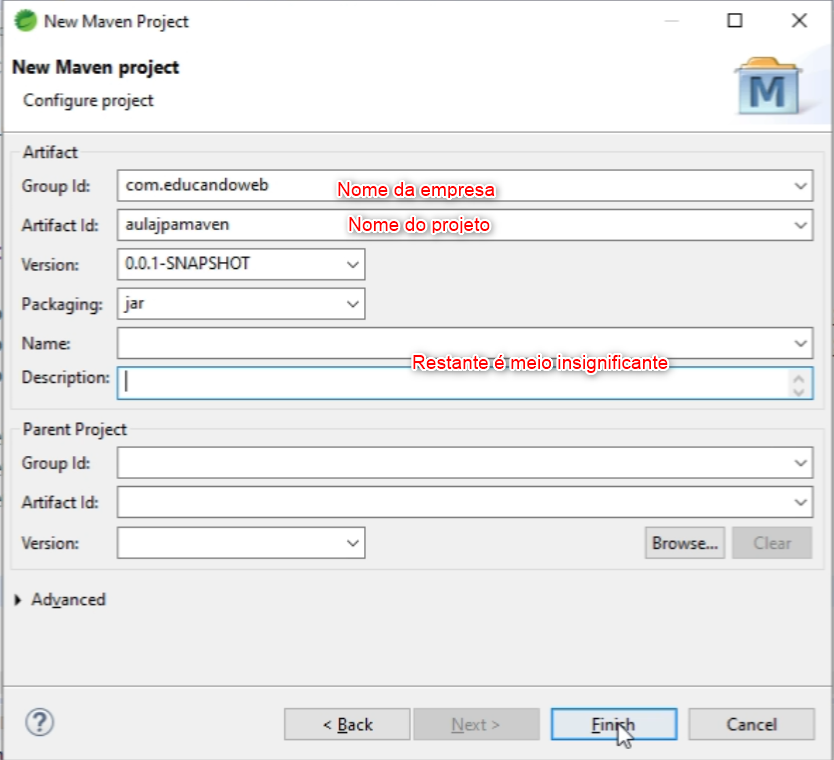
Para utilizar o EntityManager deve-se instanciar o EntityManagerFactory, apenas um para todo o projeto.

Iniciamos o Xampp e demos start no Apacha e MySql, Apache é o servidor web que possui aplicativo de gerenciador do SQL.

Xampp: é um pacote com servidores (MySQL, Apache) com linguagens PHP e Perl.

XAMPP: **X** **A**pache **M**ysql **P**hp **P**erl. Seu objetivo é criar de modo prático servidores WEB local para testes.

Maven: gerenciado de dependências e build do Java.



Porque utilizar o Java 11? Ele é LTS: Long Term Support. As empresas costumam utilizar pelo fato do suporte...Sempre as LTS se mantém. Quando sai novas versões, a última antiga não possui suporte, este é o fato de utilizar a versão LTS.

Arquivo POM: ele mostra as configurações do Maven.

OBS: O que é o Xampp? Pacote com os principais servidores open source (MySQL, Apache, etc)

Dependências Maven: sempre pesquisar na web os nomes e links; Quando colamos as dependências no pom e damos um CTRL S, o Maven vai atrás de buscar através dos links e baixa-los e colocar no projeto!

Então ele baixa os arquivos JAR para o projeto.

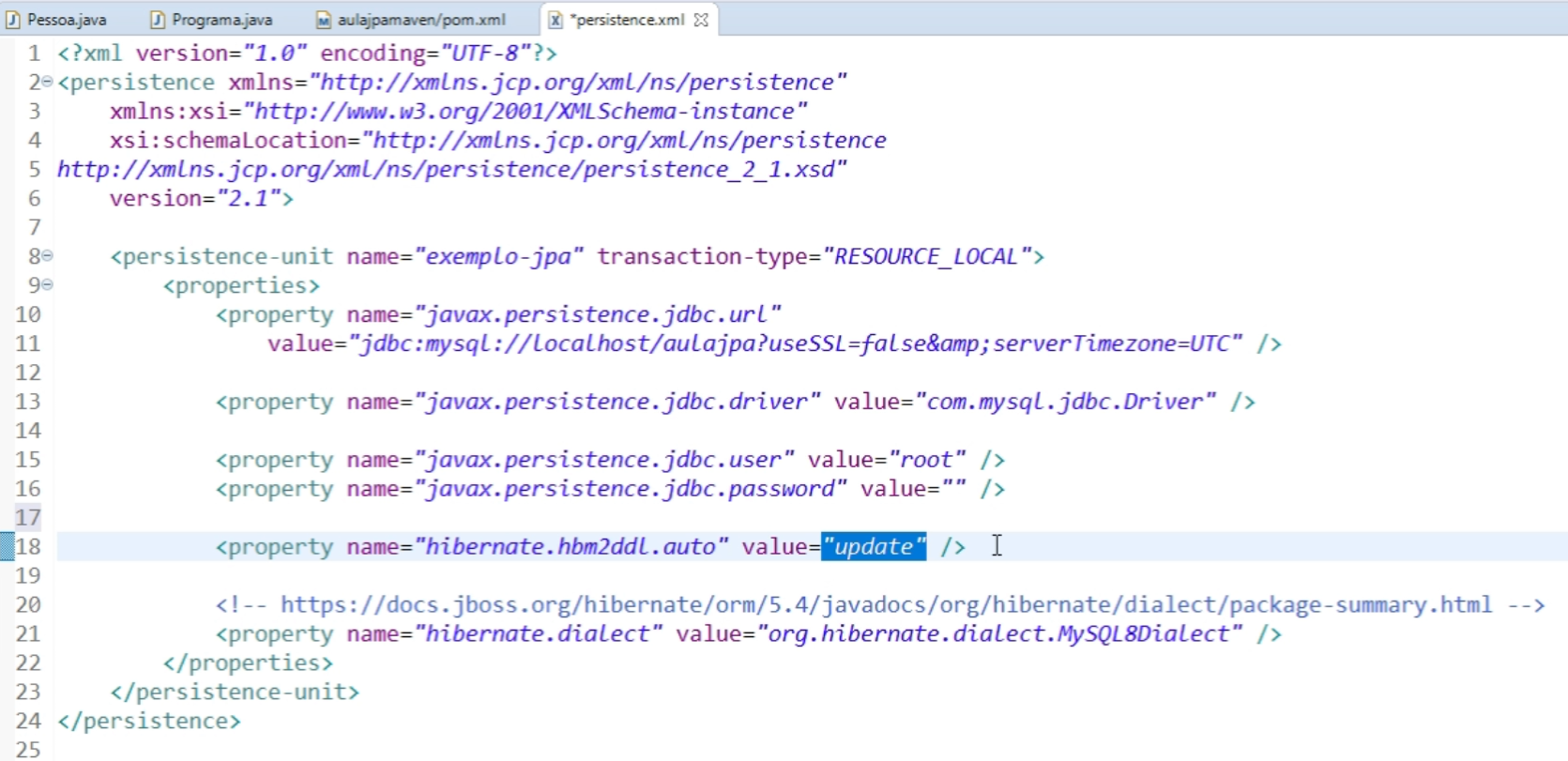
JAR: Java ARchive, são arquivos integrados ao projeto com especificações de compilação para Oracle. Assim, através destes arquivos é que veem as informações como referência ou como biblioteca para seu projeto. Lembrando que os JAR podem ser aplicativos autônomos para celulares e computadores instalados junto do JRE (Java Runtime Environment), pois conteúdos e recursos de programas podem ser incluídos neste arquivo.

Sobre o persistence.xml: (arquivo de configuração do JPA)

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

value=*"jdbc:mysql://localhost/aulajpa?useSSL=false&amp;serverTimezone=UTC"* />

Neste caso acima é local host, se fosse endereço web teria apenas que tirar o local host e por o IP. Esta é a url de conexão com o banco.



Acima vemos a configuração do JDBC e banco de dados, sendo uso do SSL falso, timeZone UTC, usuário do Banco root e password vazio (padrão do XAMPP – utilizado neste projeto), hibernate auto value update: gera automaticamente o BD e sempre que rodando a aplicação, ele atualiza automaticamente as tabelas do banco/por isso interessante o uso do Hibernate.

Ex: 

Desta forma estamos recriando o banco de dados toda vez que rodarmos a aplicação. Update seria a atualização automática...etc.

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

Acima é para gerar o banco de dados automaticamente, ou seja, alterando uma classe de domínio, automaticamente ele fará a alteração na tabela do banco de dados! Esse é o grande legal do negócio!

Quando estamos no Programa.java vemos a instanciação do Entity Manager Factory. Nele colocamos o nome que está no ***persistence.xml***!!

O framework Spring Data faz as instanciações automaticamente EM e EMF e as transações.

Entendendo o que é o ***objeto monitorado***: quando foi recém colocado no BD, ou caso ainda você não tenha dado o em.close(); e emf.close(); E ainda utilizar o comando find para buscar e poder excluir, depois dando os comandos de commit...

Spring date: dá os comandos Entity sem vermos e fica mais fácil ainda a configuração do banco de dados.

Hibernate: foi utilizado para este exemplo nas configurações do Maven, o que é?

O Hibernate é um framework para o mapeamento objeto-relacional escrito na linguagem Java. Nada mais é que uma API da Java Persistence API para transformar os objetos criado em tabelas para o banco de dados.