JPA - Java Persistence API

DSC AULA 1

Introdução

- JPA é a especificação Java para criar, recuperar, atualizar e excluir dados a partir de bancos de dados relacionais.
- JPA permite que possamos trabalhar mais focados no paradigma OO, em vez de no paradigma relacional, pois
  - diminui a complexidade da integração de uma aplicação OO Java com um banco de dados relacional;
  - facilita a implementação de código que manipula informações armazenadas no banco de dados.

Introdução

- JPA Realiza persistência de objetos Java automaticamente utilizando uma técnica conhecida como ORM (Object Relational Mapping)
  - ORM é uma técnica para a conversão de dados entre sistemas de tipos diferentes: bancos de dados relacionais e linguagens de programação orientadas a objetos.
- JPA define
  - um padrão para a criação de meta dados de configuração ORM;
  - uma interface utilizada para realizar CRUD - EntityManager;
  - Uma linguagem para pesquisa e recuperação de dados - JPQL (Java Persistence Query Language).

Comparação Modelos OO x Relacional

Modelo OO	Modelo Relacional
classes	tabelas
objetos	linhas
atributos	colunas
identidade	chave primária
relacionamento / referência a outro objeto	chave estrangeira
métodos	stored procedures

JPA

- Entities – São objetos Java que são persistidos no banco de dados relacional. Metadados ORM especificam como uma entidade é mapeada para o banco de dados.
- EntityManager – interface que gerencia as entidades fornecendo serviços de persistência. Utiliza os metadados ORM para realizar operações CRUD.
- JPQL – Acrônimo para Java Persistence Query Language. Linguagem de consulta para buscar entidades salvas no banco de dados. Possui semelhanças com o SQL, mas trabalha com objetos e não com campos de tabelas.

Implementações JPA

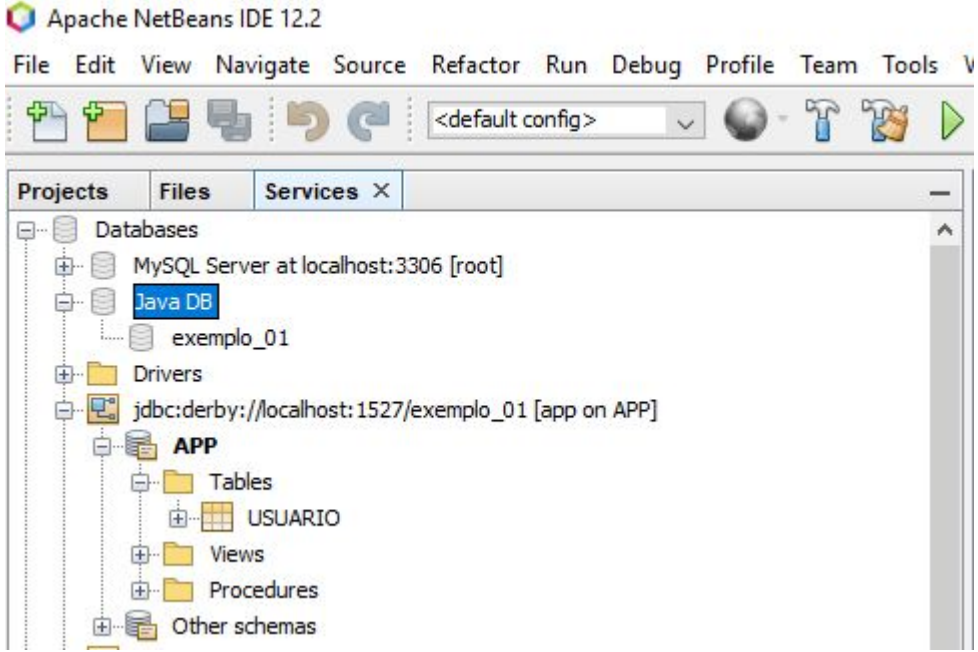
Para o nosso curso, utilizaremos apenas o Eclipselink.

Ambiente de Desenvolvimento

- JDK 8 ou superior
- IDE: Apache Netbeans 18
- Banco de dados: Derby versão 10.16.1.1
  - Basta baixar e descompactar o arquivo db-derby-10.16.1.1-bin.zip
  - O Netbeans já possui suporte ao Derby

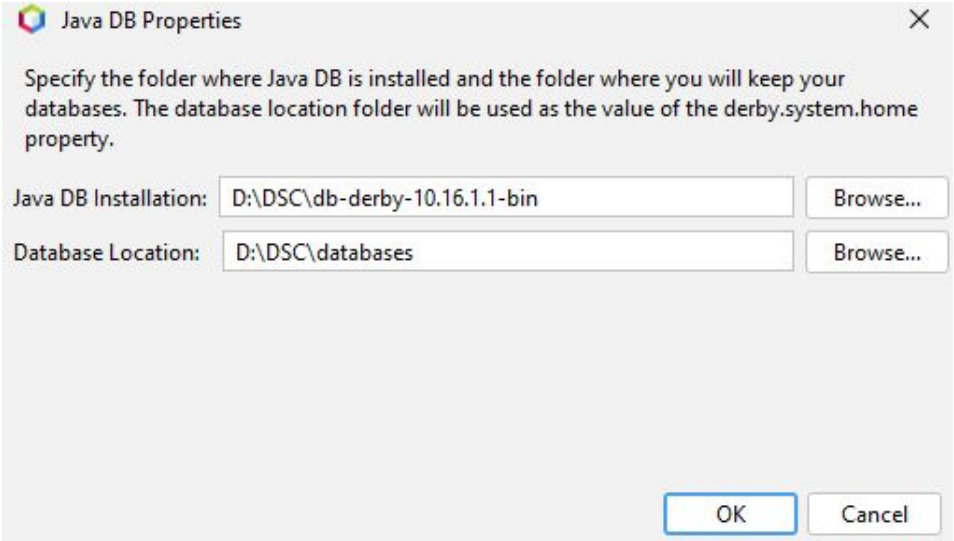
Ambiente de Desenvolvimento

- Na aba Services do Netbeans é necessário informar o caminho do Derby
  - Clique com o botão direito do mouse em Java DB e informe o caminho da instalação do banco de dados.



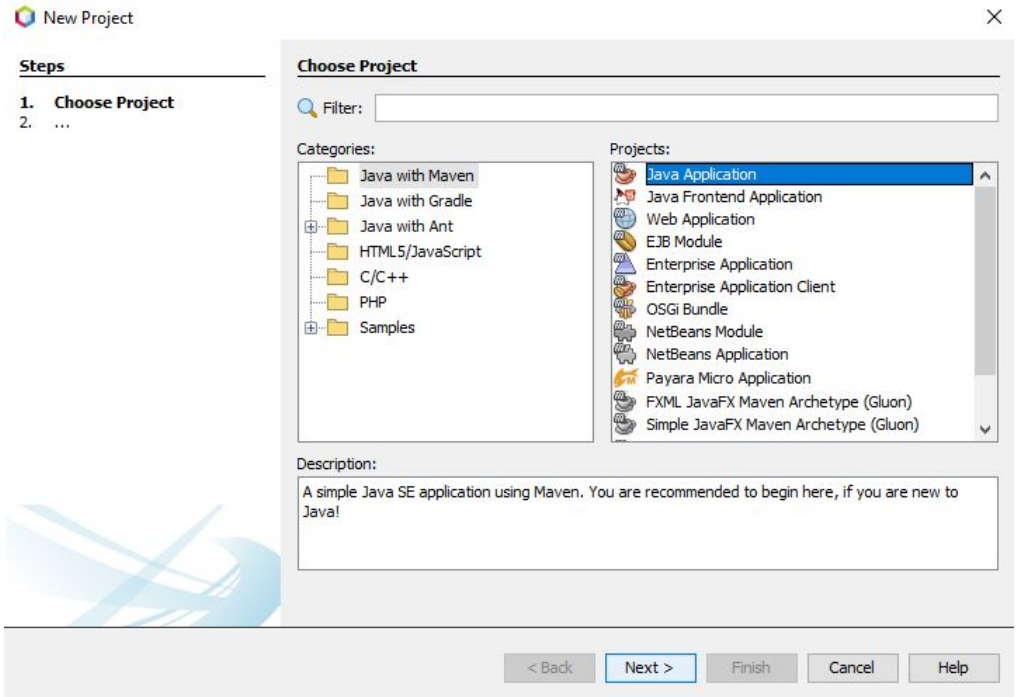
Ambiente de Desenvolvimento

- Em Java DB Properties, informe o caminho de instalação
  - No exemplo, o arquivo baixado foi descompactado em D:\DSC\db-derby-10.16.1.1-bin\
  - O Database Location é um diretório que deve ser criado para armazenamento dos arquivos dos bancos criados.



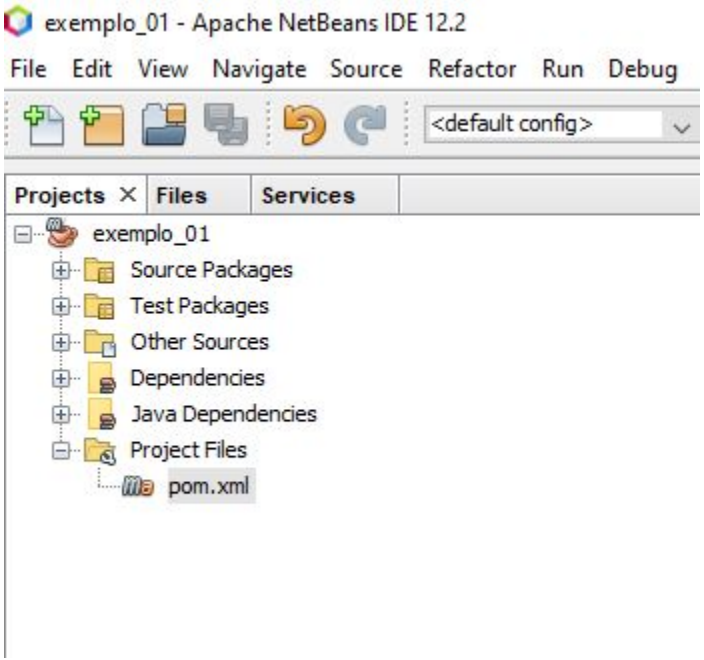
Netbeans

- Crie o projeto em
  - File -> New Project -> Java with Maven -> Java Application



pom.xml

- O arquivo que informa as bibliotecas do projeto está localizado em Project Files

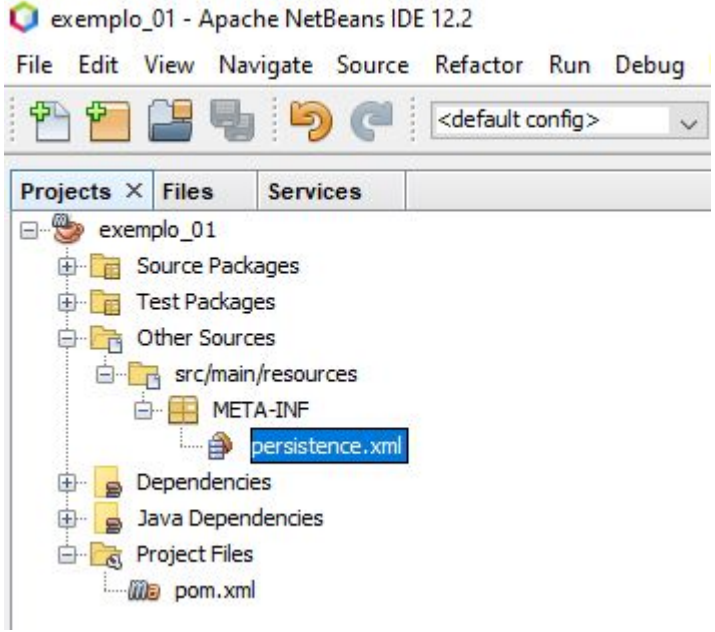


pom.xml

```
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>eclipselink</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>org.eclipse.persistence.jpa</artifactId>
    <version>4.0.4</version>
</dependency>
<dependency>
    <groupId>org.apache.derby</groupId>
    <artifactId>derbyclient</artifactId>
    <version>10.16.1.1</version>
</dependency>
<dependency>
    <groupId>org.apache.derby</groupId>
    <artifactId>derbytools</artifactId>
    <version>10.16.1.1</version>
</dependency>
```

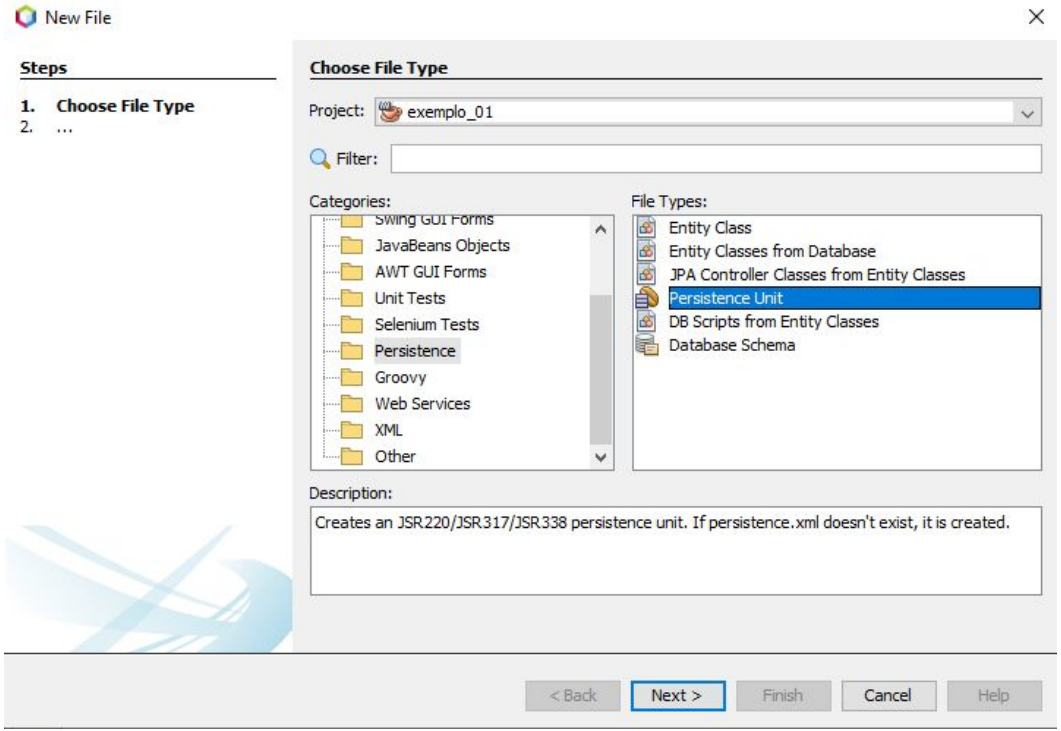
persistence.xml

- Arquivo de configuração do JPA.
- Define a unidade de persistência, que contém informações como:
  - nome;
  - tipo de transação;
  - provedor JPA;
  - definição de conexão com o banco;
  - classes de entidade.
- Este arquivo fica dentro de uma pasta chamada META-INF.



persistence.xml

- Para criá-lo, acessar o menu, a partir do projeto, com o botão direito do mouse
  - New -> Other -> Persistence -> Persistence Unit
  - O arquivo já será criado no local correto



persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="3.0" xmlns="https://jakarta.ee/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence
                                 https://jakarta.ee/xml/ns/persistence/persistence_3_0.xsd">
  <persistence-unit name="exemplo_01" transaction-type="RESOURCE_LOCAL">
    <!-- O provedor corresponde à implementação JPA utilizada. -->
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
  </persistence-unit>
</persistence>
```

persistence.xml

```
<properties>
  <property name="jakarta.persistence.jdbc.driver"
            value="org.apache.derby.jdbc.ClientDriver"/>
  <property name="jakarta.persistence.jdbc.url"
            value="jdbc:derby://localhost:1527/exemplo_01;create=true"/>
  <property name="jakarta.persistence.jdbc.user"
            value="app"/>
  <property name="jakarta.persistence.jdbc.password"
            value="app"/>
  <property name="jakarta.persistence.schema-generation.database.action"
            value="drop-and-create"/>
  <property name="eclipselink.logging.level" value="FINE"/>
  <property name="eclipselink.target-database" value="Derby"/>
  <property name="eclipselink.jdbc.native-sql" value="true"/>
</properties>
```

Entidade Usuario

```java
/*
 * @Entity informa que a classe é uma entidade JPA.
 * Por padrão, o nome da tabela é o nome da classe.
 * A entidade deve possuir um construtor padrão vazio.
 */
@Entity
public class Usuario implements Serializable {
    /*
     * @Id informa que o atributo representa a chave primária.
     * @GeneratedValue informa como gerar a chave primária.
     * GenerationType.IDENTITY é uma estratégia que pode ser utilizada no Derby.
     */
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    /*
     * Por padrão, se o nome do atributo é o nome do campo na tabela, nada
     * precisa ser feito em termos de mapeamento.
     */
    private String nome;
    private String email;
    private String login;
    private String senha;
```

EntityManager

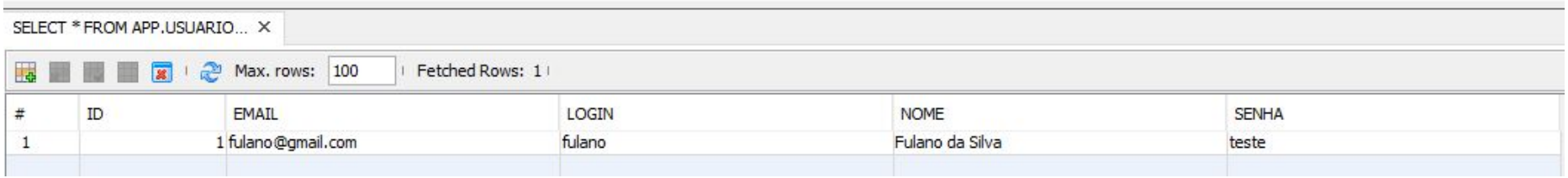
- A interface EntityManager define a API para persistência, enquanto as entidades JPA especificam como os dados da aplicação são mapeados para um banco de dados relacional.

Figure 10.1 The `EntityManager` acts as a bridge between the OO and relational worlds. It interprets the O/R mapping specified for an entity and saves the entity in the database.

Persistindo o Objeto no Banco

```
public static void main(String[] args) {
    Usuario usuario = new Usuario();
    preencherUsuario(usuario);
    EntityManagerFactory emf = null;
    EntityManager em = null;
    EntityTransaction et = null;
    try {
        //EntityManagerFactory realiza a leitura das informações relativas à "persistence-unit".
        emf = Persistence.createEntityManagerFactory("exemplo_01");
        em = emf.createEntityManager(); //Criação do EntityManager.
        et = em.getTransaction(); //Recupera objeto responsável pelo gerenciamento de transação.
        et.begin();
        em.persist(usuario);
        et.commit();
    } catch (Exception ex) {
        if (et != null)
            et.rollback();
    } finally {
        if (em != null)
            em.close();
        if (emf != null)
            emf.close();
    }
}
```

Objeto Persistido


SELECT * FROM APP.USUARIO...

Max. rows: 100 Fetched Rows: 1

#	ID	EMAIL	LOGIN	NOME	SENHA
1	1	fulano@gmail.com	fulano	Fulano da Silva	teste

Para Baixar

- O código completo do exemplo 1, acesse
  - https://svn.riouxsvn.com/dsc-ifpe/exemplo_01
- Para baixar o código, use o Tortoise SVN
  - https://tortoisesvn.net/downloads.html