

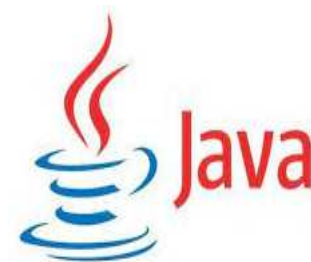


# Residência em Software

## Residência em Tecnologia da Informação e Comunicação JPA - Transações

Professor:

Alvaro Degas Coelho



INSTITUIÇÃO EXECUTORA



COORDENADORA



APOIO



# Vamos começar

- Uma base de dados (pode ser no clever cloud)
  - Uma Tabela
  - Estudante
    - **Id** INT NOT NULL AUTO\_INCREMENT
    - **Nome** VARCHAR(50) NOT NULL
    - **Email** VARCHAR(30) NOT NULL
    - **Matricula** VARCHAR(12) NOT NULL
      - PRIMARY KEY (Id)
      - UNIQUE (Email)
      - UNIQUE (Matricula)

# Criando um projeto

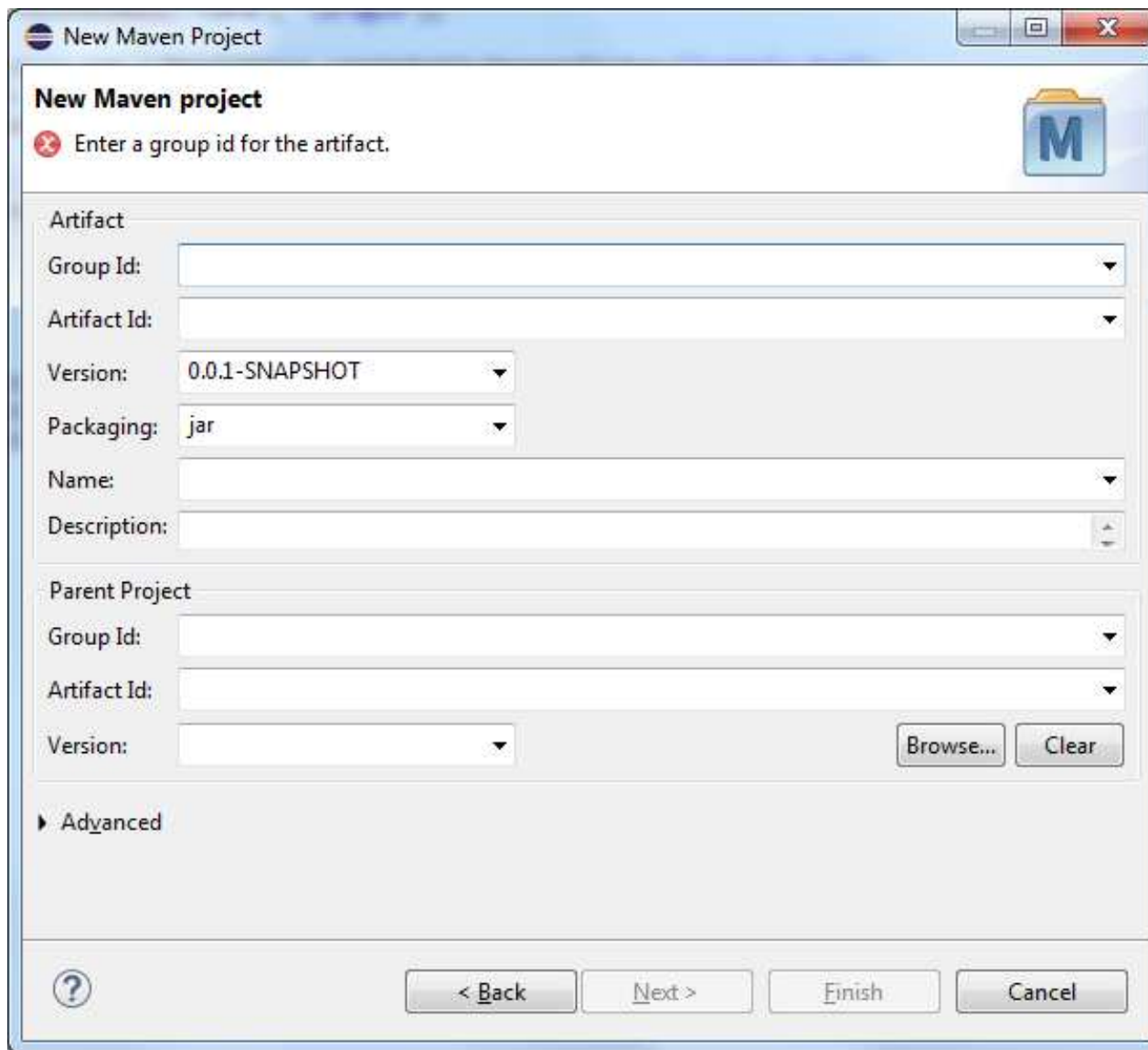
- O Maven
  - Controle de dependências
- Permite que o ambiente de desenvolvimento esteja sempre com todos os recursos necessários disponíveis
- Funciona muito bem para JPA e SpringBoot

# Criando um projeto

- File -> New -> Other -> Maven Project
- Marque Create Simple Project
- Next

GroupId: domínio  
do pacote da  
empresa/organização

ArtifactId: nome do  
projeto



New Maven Project

New Maven project

Enter a group id for the artifact.

Artifact

Group Id:

Artifact Id:

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

Parent Project

Group Id:

Artifact Id:










Version:

Browse... Clear

Advanced

? < Back Next > Finish Cancel

# Projeto criado

- ▲  SistemaAcademico
  -  src/main/java
  -  src/main/resources
  -  src/test/java
  -  src/test/resources
- ▷  JRE System Library [J2SE-1.5]
- ▷  src
-  target
-  pom.xml

# Classe Estudante

- Package academico
- Compatível com a tabela Estudante
  - **private int** Id;
  - **private String** Nome;
  - **private String** Email;
  - **private String** Matricula;
- Construtor sem parâmetros e construtor com os atributos
- Getters e Setters
- ToString()

A partir deste ponto, certifique-se de que tem o arquivo de texto com as configurações que usaremos.

Assim você poderá copiar e colar

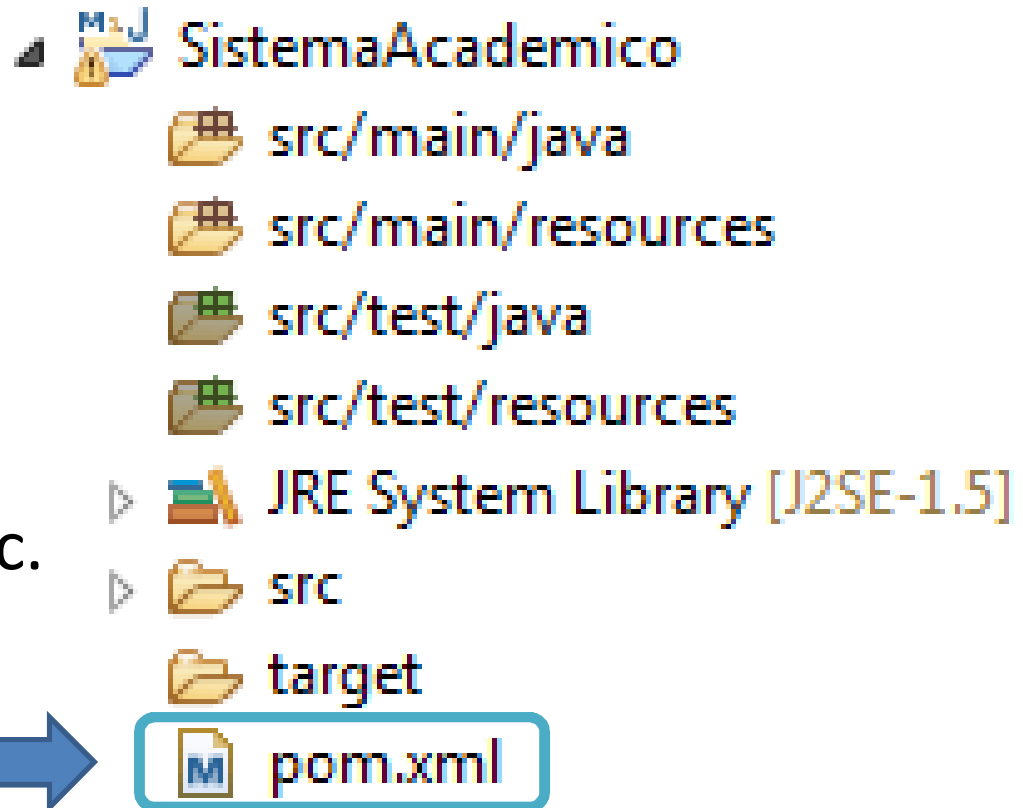


# Versão do Java

- Por padrão, maven cria projeto na versão 5
- Vamos usar a versão 11
  - LTS (Long Term Support)
    - Versões 7, 8, 11 e 17
- Como altera?
- Lembra que o JPA tem uma parte de Configuração e outra de Mapeamento?
- Configuração...

# Arquivo pom.xml

- Arquivo com configurações do Maven (dependências)
  - Dependências, versões, build, etc.



# Aumentar a versão do Java

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xs
<project xmlns="http://maven.apache.org/POM/4.0.
  <modelVersion>4.0.0</modelVersion>
  <groupId>degas</groupId>
  <artifactId>SistemaAcademico</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
  <modelVersion>4.0.0</modelVersion>
  <groupId>degas</groupId>
  <artifactId>SistemaAcademico</artifactId>
  <version>0.0.1-SNAPSHOT</version>
```



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

```
</project>
```

# Salve pom.xml

- Salve pom.xml
- Botão direito no projeto
- Escolha Maven -> Update Project
- Certifique-se que a versão mudou
  - Java SE-11

# Ainda no pom.xml

- Dependências maven
- Inclua `<dependencies> </dependencies>`
- Serão 3
  - Hibernate Core
  - Hibernate EntityManager
  - Java MySQL Connector

# Encontrar dependências

- Nos respectivos sites é possível encontrar as dependências e as Strings de dependências
  - Google Hibernate Maven
  - Certifique-se de pegar versões compatíveis
  - Google MySQL Connector maven
- Ou use o arquivo com as configurações que estamos usando na aula

# Pom.xml

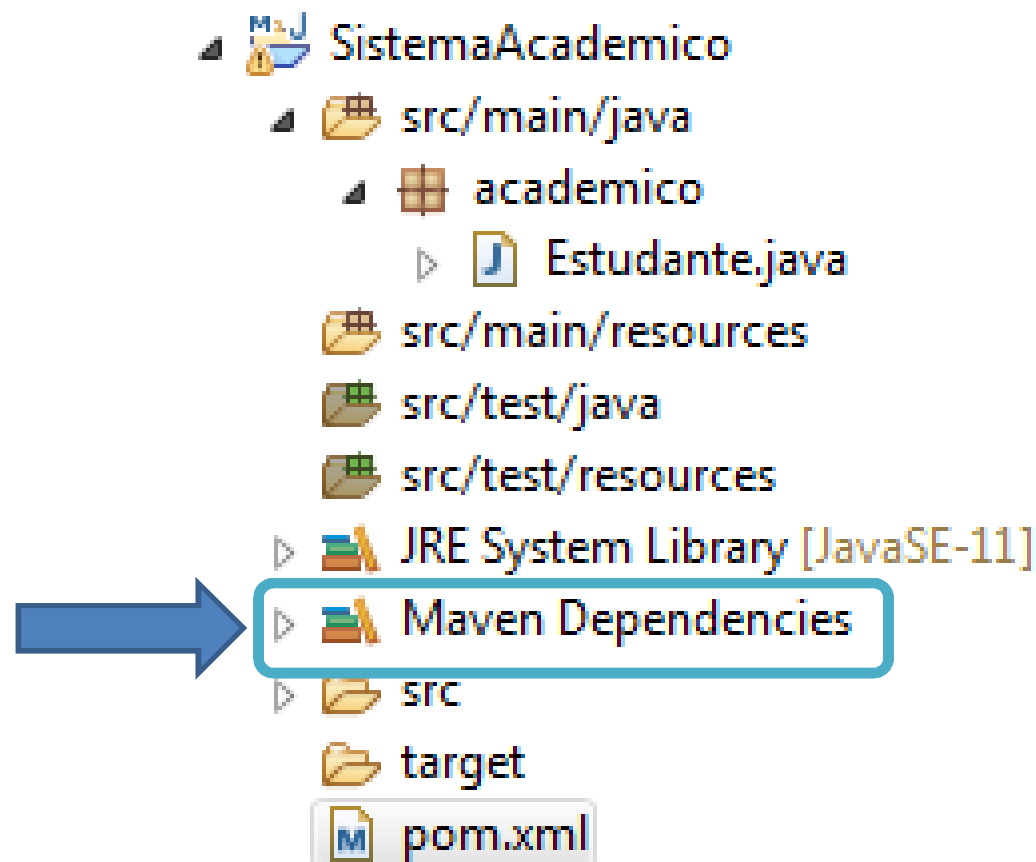
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <modelVersion>4.0.0</modelVersion>
    <groupId>degas</groupId>
    <artifactId>SistemaAcademico</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <properties>
        <maven.compiler.source>11</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>
</project>
```

# Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="
  <modelVersion>4.0.0</modelVersion>
  <groupId>degas</groupId>
  <artifactId>SistemaA</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
    <maven.compiler.<
    <maven.compiler.<
  </properties>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.hibernate
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.4.12.Final</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.hibernate
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-entitymanager</artifactId>
      <version>5.4.12.Final</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-c
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.19</version>
    </dependency>
  </dependencies>
</project>
```



# Salve pom.xml e aguarde o maven baixar as dependências...



# Continuando as configurações

- Criar o arquivo persistence.xml
- Atenção! Este arquivo deve ser criado dentro de uma pasta (folder) chamada META-INF
- A pasta META-INF deve ser criada dentro da pasta src/main/resources
- Respeite os nomes da pasta e do arquivo
  - Maiúsculas e minúsculas inclusive

# Crie o persistence.xml

- Pode copiar do arquivo com as configurações que estamos usando
- O que há no persistence.xml
  - Persistence Unit: nome da Unidade de Persistência qe estamos definindo
    - Mude o nome dela para um a seu gosto

# Mais sobre o persistence.xml

- Propriedade javax.persistence.jdbc.url
- Indica qual a URL que usaremos no SGBD
  - Mesma regra do JDBC (na verdade é JDBC)
  - Concatene “jdbc:” à URI que você tem de seu banco
    - Ficará algo como
      - jdbc:mysql://lkhrduhwkigpoqas:pslqçrjabaepdnghqzlr...

# Mais sobre o persistence.xml

- Propriedade javax.persistence.jdbc.driver
  - Driver jdbc. Deixe como está
- javax.persistence.jdbc.user
  - Seu usuário
- javax.persistence.jdbc.password
  - A senha

# Mais sobre o persistence.xml

- `hibernate.hbm2ddl.auto`
  - Update: o banco será atualizado junto com as classes
  - Create: recria o BD toda vez que roda
- `hibernate.dialect`
  - O SQL gerado pelo Hibernate vai rodar no seu BD?
  - Depende da versão...
  - Para o clever-cloud, deixe como está

Maven configurado. Agora vamos programar...

# Annotations

- Classe Estudante
  - @Entity
  - Lembre de importar `javax.persistence.Entity`
- Atributo Id
  - @Id
  - @GeneratedValue (strategy=GenerationType.**AUTO**)
  - Lembre de importar
    - `javax.persistence.Id`
    - `javax.persistence.GeneratedValue`
    - `javax.persistence.GenerationType`



# Importante

- O Id será gerado automaticamente pelo BD. Logo, não podemos enviá-lo
  - Instancie como null no construtor
- Isto forçará que o id seja do tipo Integer (classe) não int (primitivo)

# Programando pra ver

- Método main
- Procedimento
  - Vamos criar três estudantes
    - e1, e2 e e3 (instancie corretamente)
  - Vamos criar um EntityManagerFactory
    - Usando nossa JPA-UNIT (definida no persistence.xml)
  - Vamos criar um EntityManager
  - Usando o EntityManager
    - Iniciar uma transação
    - Persistir os dados
    - Commit
  - Fechar o EM e o EMF

# Como fica

```
public static void main(String[] args) {  
    Estudante e1 = new Estudante(null, "Tõe", "toe@tutu", "111111");  
    Estudante e2 = new Estudante(null, "Lia", "lia@tutu", "222222");  
    Estudante e3 = new Estudante(null, "Tuca", "tuca@tutu", "333333");  
  
    EntityManagerFactory emf =  
        Persistence.createEntityManagerFactory("exemplo-jpa");  
    EntityManager em = emf.createEntityManager();  
    em.getTransaction().begin();  
    em.persist(e1);  
    em.persist(e2);  
    em.persist(e3);  
    em.getTransaction().commit();  
    em.close();  
    emf.close();  
}
```

# Recuperando dados


- Pelo Id
  - Olhe um ID válido no seu BD (no nosso exemplo, 2)
- Procedimento
  - Obtenha o EntityManager
  - Execute o método Find(Classe, Chave)
  - Guarde num objeto do tipo adequado

## Como fica

```
EntityManagerFactory emf =  
Persistence.createEntityManagerFactory("exemplo-jpa");  
EntityManager em = emf.createEntityManager();  
Estudante r = em.find(Estudante.class, 8);  
System.out.println(r);  
em.close();  
emf.close();
```

# Lembram disso?

## Problemas mais concretos

- 
- Contexto de Persistência
    - E se um objeto não estiver no contexto de salvar/alterar no SGBD (veremos em detalhes)?
  - Mapa de Identidade
    - Quais objetos de memória estão persistidos corretamente no SGBD? Quais não estão?
  - Carregamento Tardio
    - E se eu quiser pegar o Usuário, mas não as postagens dele?
      - Um DAO só pra usuário e outro para Usuários e Postagens?

# Tentar excluir um estudante

- Método remove do EM
- Procedimento “Natural”
  - Pegar um objeto com o ID correto
  - Executar o Remove
  - Try it!
- **IMPORTANTE!** Operações DML só são efetivamente executadas dentro de uma transação!

# Como fica

```
EntityManagerFactory emf =  
Persistence.createEntityManagerFactory("exemplo-jpa");  
EntityManager em = emf.createEntityManager();  
Estudante r = new Estudante(8, "Lia", "lia@tutu", "222222");  
em.getTransaction().begin();  
em.remove(r);  
em.getTransaction().commit();  
em.close();  
emf.close();
```



# Resultado

Exception in thread "main" java.lang.IllegalArgumentException: Removing a detached instance academico.Estudante#8

- JPA só pode operar sobre objetos que
  - Acabaram de serem inseridos
  - Tenham sido recuperados do banco
    - Sem fechar o EntityManager
- Fora destas condições o dado está sem contexto
  - Destacado (Detached)

# Como funciona?

```
EntityManagerFactory emf =  
Persistence.createEntityManagerFactory("exemplo-jpa");  
EntityManager em = emf.createEntityManager();  
Estudante r = em.find(Estudante.class, 8);  
em.getTransaction().begin();  
em.remove(r);  
em.getTransaction().commit();  
em.close();  
emf.close();
```

# Exercício

- Implemente um pequeno trecho que atualize um Estudante
- Dica: o método `persist()` serve para inserir ou atualizar
  - Contexto de persistência!

# Exercício

- Recrie o projeto de rede social
  - Usando Hibernate (configure de acordo)
- Crie a classe Usuario na package `src/main/java/model`
- Crie sua correspondente no BD
- Crie trechos de programa (método main) para inserir, excluir, alterar e recuperar usuários

# Exercício

- Crie um projeto chamado Clinica
  - Usando Hibernate (configure de acordo)
- Crie as classes Medico, Paciente e Procedimento (Id e Nome apenas) na package `src/main/java/model`
- Execute persistências usando JPA e veja a tabela ser criada no BD
- Crie trechos de programa (método main) para inserir, excluir, alterar e recuperar dados