**Exercícios: Configurando o JPA e gerando o schema do banco**

1. Vamos preparar nosso projeto com as dependências do Hibernate.

Crie uma pasta lib dentro do seu projeto e copie os JARs do Hibernate para a pasta:

* 1. Vá ao diretório \\maverick\ADS\Prof. Madeira\Linguagem 3\libs\hibernate
  2. Selecione todos os JARs, clique com o botão direito e escolha *Copy* (ou CTRL+C);
  3. Cole todos os JARs na pasta **lib** do projeto (CTRL+V)

1. Anote a classe Tarefa como uma entidade de banco de dados. Lembre-se que essa é uma anotação do pacote javax.persistence.

*Obs: Não apague nenhuma anotação, apenas adicione as anotações do JPA.*

@Entity

**public** **class** Tarefa {

}

Na mesma classe anote seu atributo id como chave primária e como campo de geração automática:

@Id

@GeneratedValue

**private** Long id;

Agora é preciso mapear o atributo dataFinalizacao para gravar apenas a data (sem hora) no banco:

@Temporal(TemporalType.DATE)

**private** Calendar dataFinalizacao;

1. Vamos configurar o persistence.xml
   1. Clique com o botão direito na pasta src do projeto e escolha *New -> Folder*. Escolha **META-INF** como nome dessa pasta.
   2. Vá ao diretório \\maverick\ADS\Prof. Madeira\Linguagem 3\ e copie o arquivo persistence.xml para a pasta META-INF.
   3. Adicione dentro do elemento persistence:

**<persistence-unit** name="tarefas"**>**

**<provider>**org.hibernate.ejb.HibernatePersistence**</provider>**

**<class>**br.com.caelum.tarefas.modelo.Tarefa**</class>**

**<properties>**

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

**<property** name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/impacta" **/>**

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

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

**<property** name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect" **/>**

**<property** name="hibernate.show\_sql" value="true" **/>**

**<property** name="hibernate.format\_sql" value="true" **/>**

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

**</properties>**

**</persistence-unit>**

As duas propriedades show\_sql e format\_sql fazem com que todo SQL gerado pelo Hibernate apareça no console.

1. Crie a classe GeraTabelas no pacote br.com.impacta.tarefas.jpa.

**package** br.com.impacta.tarefas.jpa;

**public** **class** GeraTabelas {

**public** **static** **void** main(String[] args) {

EntityManagerFactory factory = Persistence.

createEntityManagerFactory("tarefas");

factory.close();

}

}

1. Crie sua tabela executando a classe GeraTabelas. Para isso, clique da direita no código e vá em *Run As -> Java Application*.
2. Agora, abra um **Terminal** do mysql. Após isto, digite *use impacta;* e em seguida, *show tables;* Se seu banco de dados já existia, e não foi preciso criá-lo no passo anterior, você irá notar a presença de uma tabela chamada *tarefas*. Não é esta a tabela que queremos. Procuramos pela tabela **T***arefa*, com **T**, em maiúsculo (igual ao nome da classe Tarefa).
3. Crie uma classe chamada AdicionaTarefa no pacotebr.com.caelum.tarefas.jpa, ela vai criar um objeto e adicioná-lo ao banco:

**package** br.com.impacta.tarefas.jpa;

**public** **class** AdicionaTarefa {

**public** **static** **void** main(String[] args) {

Tarefa tarefa = **new** Tarefa();

tarefa.setDescricao("Estudar JPA e Hibernate");

tarefa.setFinalizado(**true**);

tarefa.setDataFinalizacao(Calendar.getInstance());

EntityManagerFactory factory = Persistence.

createEntityManagerFactory("tarefas");

EntityManager manager = factory.createEntityManager();

manager.getTransaction().begin();

manager.persist(tarefa);

manager.getTransaction().commit();

System.out.println("ID da tarefa: " + tarefa.getId());

manager.close();

}

}

1. Rode a classe AdicionaTarefa e adicione algumas tarefas no banco. Saída possível:



1. Verifique os registros no banco, se conecte com o cliente do mysql:

select \* from Tarefa;

Rode novamente a classe AdicionaTarefa e verifique o banco.

1. Vamos carregar tarefas pela chave primária.

Crie uma classe chamada CarregaTarefa no pacote br.com.impacta.jpa:

**package** br.com.impacta.tarefas.jpa;

**public** **class** CarregaTarefa {

**public** **static** **void** main(String[] args) {

EntityManagerFactory factory = Persistence.

createEntityManagerFactory("tarefas");

EntityManager manager = factory.createEntityManager();

Tarefa encontrada = manager.find(Tarefa.class, 1L);

System.out.println(encontrada.getDescricao());

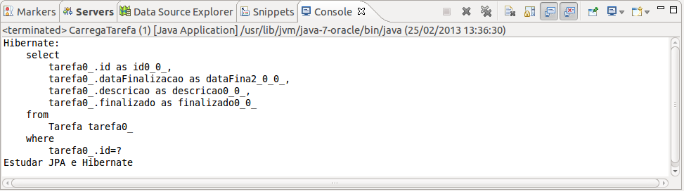
manager.close();

}

}

1. Rode a classe CarregaTarefa e verifique a saída.

Caso recebeu uma exception, verifique o id da tarefa. Ele deve existir no banco.



1. Crie uma classe chamada BuscaTarefas no pacote br.com.impacta.tarefas.jpa:

Cuidado, a classe Query vem do pacote javax.persistence:

**import** javax.persistence.Query;

**public** **class** BuscaTarefas {

**public** **static** **void** main(String[] args) {

EntityManagerFactory factory = Persistence.

createEntityManagerFactory("tarefas");

EntityManager manager = factory.createEntityManager();

//cuidado, use o import javax.persistence.Query

Query query = manager

.createQuery("select t from Tarefa as t "+

"where t.finalizado = :paramFinalizado");

query.setParameter("paramFinalizado", **true**);

List<Tarefa> lista = query.getResultList();

**for** (Tarefa t : lista) {

System.out.println(t.getDescricao());

}

manager.close();

}

}

1. Rode a classe BuscaTarefas e verifique a saída.

