

	DE IND	
- A I I W II) &	7 I I I I I I I I I I I I I I I I I I I	17/11/11/14/1

Cap 4 - Traduzindo objeto para o modelo relacional - Mãos a obra

DE 06/02/2023 A 27/02/2023

PRAZO ENCERRADO

↑, ENTREGAR ATIVIDADE

Atividades entregues até 3 dias após o prazo receberão até 70% da nota.

O cálculo é feito automaticamente pelo sistema, o professor não tem controle sobre o percentual da nota atribuída.

INTRODUÇÃO

Atividade - Mãos à obra!

Faça o upload de sua atividade na plataforma FIAP ON, na seção de atividades e aguarde a nota e o feedback do professor.

Atenção: Verifique se o arquivo do upload está correto, não é possível enviar um outro arquivo após fechamento da entrega na plataforma ou correção do professor.

Até agora vimos o conceito de ORM e alguns detalhes sobre sua realização. Ficamos a par também das informações iniciais sobre o JPA e Hibernate, que são as ferramentas mais usadas em Java para o Mapeamento Objeto-Relacional. Agora chegou a hora de praticar!

Indicamos a criação de um novo projeto com JPA/Hibernate utilizando o Maven. Conforme visto neste capítulo, configure o persistence.xml para utilizar um banco de dados e crie a classe de ORM Estabelecimento. Sinta-se livre para usar o projeto-base disponibilizado no capítulo.

O código-fonte abaixo apresenta um exemplo de classe para o banco de dados Oracle. Caso utilize outro banco, não esqueça de ajustar a criação automática de índices (sequence), alguns bancos de dados relacionais não utilizam esse conceito.

package br.com.fiap.smartcities.domain;

import java.util.Calendar;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

banco de dados escolhido precisa ter acesso de qualquer computador, ou seja, precisa estar hospedado em

algum servidor que tenha acesse livre na internet.

Por fim, não é permitido o uso de banco de dados não-relacionais (NOSQL).

```
import javax.persistence.GenerationType;
import javax.persistence.ld;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import org.hibernate.annotations.CreationTimestamp;
@Entity
@Table(name = "tbl_estabelecimento")
public class Estabelecimento {
 @ld
@SequenceGenerator(name="estabelecimento",sequenceName="sq_tbl_estabelecimento",allocationSize=1)
 @GeneratedValue(strategy = GenerationType.SEQUENCE, generator="estabelecimento")
 @Column(name = "id_estabelecimento")
 private Integer id;
  @Column(name = "nome_estabelecimento", length = 50)
 private String nome;
  @CreationTimestamp
 @Temporal(TemporalType.TIMESTAMP)
  @Column(name = "dh_criacao")
 private Calendar dataCriacao;
 public Integer getId() {
```

```
return id;
VOLTAR À LISTA
      this.id = id;
   public String getNome() {
      return nome;
   public void setNome(String nome) {
      this.nome = nome;
   public Calendar getDataCriacao() {
      return dataCriacao;
   public void setDataCriacao(Calendar dataCriacao) {
      this.dataCriacao = dataCriacao;
                              Código-fonte 1 – Entidade Estabelecimento para o banco de dados Oracle
                                                 Fonte: Elaborado pelo autor (2021)
 Só com a entidade não é possível testar. Por isso, vamos criar uma classe com o método main e implementar o código-fonte a
 seguir.
 Neste momento, não precisamos entender o que está acontecendo. Veremos isso com mais detalhes no próximo capítulo.
 package br.com.fiap.smartcities.test;
 public class BancoTeste {
   public static void main(String[] args) {
      Persistence.createEntityManagerFactory("smartcities-orm").createEntityManager();
```

VOLTAR À LISTA

Fonte: Elaborado pelo autor (2021)

Certifique-se de que o parâmetro do método createEntityManagerFactory ("smartcities-orm") é o mesmo que está configurado no arquivo persistence.xml, o nome da persistence unit. Depois, é só executar! Como o arquivo persistence.xml está configurado para criar as tabelas e exibir os SQL enviados para o banco, o resultado esperado é a criação da tabela, que pode ser verificada no console do eclipse.

Figura 1 – Resultado da execução do programa Fonte: Elaborado pelo autor (2021)

Observe que, primeiramente, o programa envia os SQL para apagar a tabela e a sequência, comportamento-padrão para a configuração "create". Não se preocupe se ocorrer um erro neste ponto, porque possivelmente você não tem a tabela e a sequência no banco para serem apagadas. Depois, consegue-se ver as queries que criam a tabela e a sequência.

Fique à vontade para entrar no banco de dados verificar a tabela criada. Tente também modificar a classe de ORM, adicionando novos atributos. Crie uma classe e execute o programa. Veja que o banco de dados será modificado!

Sucesso! Já temos a entidade de estabelecimento em nosso sistema e também em nosso banco de dados. Agora, para completar a entrega, crie mais **duas entidades** (classe domain do Java) de um sistema imaginário (Ex.: para um sistema de controle de alunos, poderíamos ter as entidades de aluno, disciplinas, professores, tutores e outros). Execute a classe **main** e acompanhe a criação das tabelas no banco de dados.

O próximo passo é realizar as operações básicas do CRUD, que será tema do próximo capítulo!

<u>↑</u>	Envio	ENTREGA PENDENTE
<u></u>	Avaliação	AVALIAÇÃO PENDENTE

PRAZO DE ENTREGA

Segunda-feira, 27 de Fevereiro de 2023, às 23h59

TEMPO RESTANTE

⚠ A tarefa está atrasada 259 dias e 23 horas

ÚLTIMA MODIFICAÇÃO

-

Oprazo des prividade foi encerrado. Se você tiver dúvidas ou quiser conversar com a gente, manda uma mensagem para seu tutor no canal de comunicação da turma.



