**Estabelecimento:**

package br.com.bandtec.estabelecimentosjpa.model;

import java.util.Calendar;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import org.hibernate.annotations.CreationTimestamp;

@Entity

public class Estabelecimento {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id\_estabelecimento")

private Long id;

@Column(length = 50, nullable = false)

private String nome;

@CreationTimestamp

@Column(name = "data\_criacao")

private Calendar dataCriacao;

@JoinColumn(name = "id\_tipo\_estabelecimento")

@ManyToOne(fetch = FetchType.LAZY)

private TipoEstabelecimento tipo;

public Long getId() {

return id;

}

public void setId(Long id) {

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;

}

public TipoEstabelecimento getTipo() {

return tipo;

}

public void setTipo(TipoEstabelecimento tipo) {

this.tipo = tipo;

}

@Override

public String toString() {

return "Estabelecimento{" + "id=" + id + ", nome=" + nome + ", tipo=" + tipo + '}';

}

}

**TipoEstabelecimento**

package br.com.bandtec.estabelecimentosjpa.repository;

import br.com.bandtec.estabelecimentosjpa.db.Conexao;

import br.com.bandtec.estabelecimentosjpa.model.Estabelecimento;

import br.com.bandtec.estabelecimentosjpa.model.TipoEstabelecimento;

import java.util.List;

import javax.persistence.criteria.CriteriaQuery;

public class TipoEstabelecimentoRepository

extends GenericRepository<TipoEstabelecimento> {

public TipoEstabelecimentoRepository(Conexao conexao) {

super(conexao);

}

public List<TipoEstabelecimento> recuperarTodos() {

return this.em.createQuery("from TipoEstabelecimento").getResultList();

// outra forma de escrever o jpql

// createQuery("select t from TipoEstabelecimento t").getResultList();

}

// consulta de todos usando Criteria

public List<TipoEstabelecimento> recuperarTodos2() {

CriteriaQuery<TipoEstabelecimento> query = this.em

.getCriteriaBuilder()

.createQuery(TipoEstabelecimento.class);

query.from(TipoEstabelecimento.class);

return this.em.createQuery(query).getResultList();

}

public List<TipoEstabelecimento> recuperar3Ultimos() {

return this.em.

createQuery("from TipoEstabelecimento order by id desc")

.setMaxResults(3)

.getResultList();

}

public void excluir2(Long id) {

this.em

.createQuery("delete from TipoEstabelecimento where id=?")

.setParameter(1, id)

.executeUpdate();

}

public List<TipoEstabelecimento> recuperarPorOrdemAlfabetica() {

return this.em.createQuery(

"from TipoEstabelecimento order by nome").getResultList();

}

public List<TipoEstabelecimento> recuperarPorNome(String nome) {

return this.em.createQuery(

"from TipoEstabelecimento where nome = ?")

.setParameter(1, nome)

.getResultList();

}

public List<TipoEstabelecimento> recuperarPorNomeContendo(String termo) {

return this.em.createQuery(

"from TipoEstabelecimento where nome like ?")

.setParameter(1, "%"+termo+"%")

.getResultList();

}

public String

getRelatorioEstabelecimentos(TipoEstabelecimento tipo) {

String retorno = "Total de estabelecimentos: "

+tipo.getEstabelecimentos().size()

+"\n";

for (Estabelecimento estabelecimento : tipo.getEstabelecimentos()) {

retorno += " \* "+estabelecimento+"\n";

}

return retorno;

}

@Override

protected Class getClasse() {

return TipoEstabelecimento.class;

}

}