**GenericRepository**

package br.com.bandtec.estabelecimentosjpa.repository;

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

import javax.persistence.EntityManager;

// Classe com tipo GENRIC (no caso, o T)

// Usamos a letra T, mas poderia ser quanquer letra ou palavra

public abstract class GenericRepository<T> {

protected EntityManager em;

public GenericRepository(Conexao conexao) {

this.em = conexao.getEm();

}

protected abstract Class getClasse();

public void salvar(Object entidade) {

this.em.persist(entidade);

}

// recuperar usando o recurso de GENERICS

public T recuperar(Long id) {

return this.em.find((Class<T>) this.getClasse(), id);

}

public void excluir(Long id) {

this.em.remove(this.recuperar(id));

}

}

**EstabelecimentoRepository**

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;

public class EstabelecimentoRepository extends GenericRepository<Estabelecimento> {

public EstabelecimentoRepository(Conexao conexao) {

super(conexao);

}

public List<Estabelecimento> recuperarTodos() {

return this.em.

createQuery("from Estabelecimento")

.getResultList();

}

public List<Estabelecimento> recuperarPorNomeContendo(String termo) {

return this.em.createQuery(

"from Estabelecimento where nome like ?")

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

.getResultList();

}

@Override

protected Class getClasse() {

return Estabelecimento.class;

}

}

**TipoEstabelecimentoRepository**

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;

}

}