

Banco de Dados II

Persistência

Vanessa Cristina Oliveira de Souza



HIBERNATE



Hibernate



- Desenvolvimentos:
 - □ Top Down
 - Modelo de domínio -> MR
 - **Botton Up**
 - MR -> Modelo de domínio



Hibernate



- Criar projeto
- 2. Arquivo de configuração
 - Configura a conexão com o banco de dados
- 3. Arquivo de engenharia reversa
 - Arquivo xml com os dados do banco (tabelas)
- 4. Mapeamento Objeto-Relacional
 - POJOs de todas as tabelas do banco
- 5. Arquivo HibernateUtil
 - Para abrir as sessões



Hibernate



- Prática
 - Banco northwind



Hibernate Inserir um objeto no banco

```
package javaapplication1;
import map.*;
import org.hibernate.*;
 * @author vanessa
public class JavaApplication1 {
    * @param args the command line arguments
    public static void main(String[] args) {
        try{
            //cria um objeto categoria
            Categories cat = new Categories();
           //atribui valores a seus atributos
            cat.setCategorvid(11);
            cat.setCategoryname("teste");
            cat.setDescription("categoria teste");
           //abre uma sessão
            Session sessao = HibernateUtil.getSessionFactory().openSession();
           //salva objeto cat - objeto transiente
            sessao.save(cat);
           //abre uma transação
            Transaction tr = sessao.beginTransaction();
            //comita no banco o que foi salvo até agora
            //torna um objeto transiente em objeto persistente
            tr.commit();
        catch (Exception e)
            e.printStackTrace();
        HibernateUtil.getSessionFactory().close();
```



Hibernate Inserir um objeto no banco

categoryid [PK] integer	categoryname character varying (50)	description character varying (100)
1	Beverages	Soft drinks
2	Condiments	Sweet and savory sauces
3	Confections	Desserts, candies, and swe
4	Dairy northwind.products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, an
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

categoryid [PK] integer	categoryname character varying (50)	description character varying (100)
1	Beverages	Soft drinks
2	Condiments	Sweet and savory sauces
3	Confections	Desserts, candies, and swe
4	Dairy northwind.products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, an
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish
11	teste	categoria teste



Hibernate Apagar um objeto do banco

```
public class JavaApplication1 {
    public static void main(String[] args) {
        try{
            //cria um objeto categoria
            Categories cat = new Categories();
           //abre uma sessão
            Session sessao = HibernateUtil.getSessionFactory().openSession();
           //recupera um objeto do tipo categoria
            //torna um objeto persistente em um transiente
          sessao.load(cat, 11);
          sessao.delete(cat);
           //abre uma transação
            Transaction tr = sessao.beginTransaction();
            tr.commit();
        catch (Exception e)
            e.printStackTrace();
        HibernateUtil.getSessionFactory().close();
```



Hibernate Inserir um objeto no banco

categoryid [PK] integer	categoryname character varying (50)	description character varying (100)
1	Beverages	Soft drinks
2	Condiments	Sweet and savory sauces
3	Confections	Desserts, candies, and swe
4	Dairy northwind.products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, an
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish
11	teste	categoria teste

categoryid [PK] integer	categoryname character varying (50)	description character varying (100)
1	Beverages	Soft drinks
2	Condiments	Sweet and savory sauces
3	Confections	Desserts, candies, and swe
4	Dairy northwind.products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, an
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish



Update



```
public class JavaApplication1 {
    public static void main(String[] args) {
        try{
           //cria um objeto categoria
            Categories cat = new Categories();
            //abre uma sessão
            Session sessao = HibernateUtil.getSessionFactory().openSession();
            //recupera um objeto do tipo categoria
            //torna um objeto persistente em um transiente
          >sessao.load(cat, 4);
            //altera o valor do atributo
           cat.setCategoryname("Dairy Products");
           sessao.update(cat);
            //abre uma transação
            Transaction tr = sessao.beginTransaction();
            tr.commit();
        catch (Exception e)
            e.printStackTrace();
        HibernateUtil.getSessionFactory().close();
```



Hibernate Inserir um objeto no banco

categoryid [PK] integer		categoryname character varying (50)	description character varying (100)
	1	Beverages	Soft drinks
	2	Condiments	Sweet and savory sauces
	3	Confections	Desserts, candies, and swe
		Deier en en de en de en en	CI
	4	Dairy northwind.products	Cheeses
	4	Grains/Cereals	Breads, crackers, pasta, an
	4		
	5	Grains/Cereals	Breads, crackers, pasta, an

categoryid [PK] integer	categoryname character varying (50)	description character varying (100)
1	Beverages	Soft drinks
2	Condiments	Sweet and savory sauces
3	Confections	Desserts, candies, and swe
4	Dairy Products	Cheeses
	Dairy Products Grains/Cereals	Cheeses Breads, crackers, pasta, an
5	1	
5	Grains/Cereals	Breads, crackers, pasta, an





- Cadastrar um novo pedido (northwind.order):
 - Customerid CHOPS
 - □ Employeeid 9
 - Os demais dados podem ser aleatórios



```
package javaapplication2;
import java.math.BigDecimal;
import java.text.*;
import map.*;
import org.hibernate.*;
import java.util.*;
```

```
private int orderid;
private Customers customers;
private Employees employees;
private Date orderdate;
private Date requireddate;
private Date shippeddate;
private BigDecimal freight;
private String shipname;
private String shipaddress;
private String shipcity;
private String shipregion;
private String shippediate;
private String shippeddate;
private String shippeddate;
private String shippeddate;
private String shippedia;
```

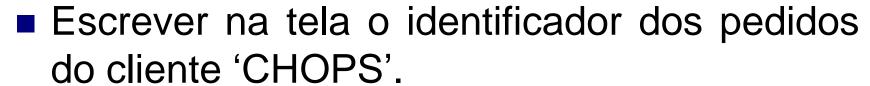
```
public class JavaApplication2 {
   public static void main(String[] args) {
       try{
            SimpleDateFormat ft = new SimpleDateFormat ("yyyy-MM-dd");
            //cria um objeto categoria
          Orders pedido = new Orders();
           .Customers cliente = new Customers();
           Employees func = new Employees();
           Date Orderdate = ft.parse("2018-03-10");
            Date requireddate = ft.parse("2018-03-20");
            Date shippeddate = ft.parse("2018-03-30");
          BigDecimal Freight = new BigDecimal(53.26);
            pedido.setOrderid(5);
            pedido.setOrderdate(Orderdate);
            pedido.setRequireddate(requireddate);
            pedido.setShippeddate(shippeddate);
            pedido.setFreight(Freight);
            pedido.setShipname("UNIFEI");
            pedido.setShipaddress("Av. BPS");
            pedido.setShipcity("Itajuba");
            pedido.setShipregion("MG");
            pedido.setShippostalcode("37501-000");
            pedido.setShipcountry("Brasil");
            pedido.setShipperid(3);
```



```
private int orderid;
private Customers customers;
private Employees employees;
private Date orderdate;
private Date requireddate;
private BigDecimal freight;
private String shipname;
private String shipaddress;
private String shiprity;
private String shipregion;
private String shippostalcode;
private String shippostalcode;
private String shippostalcode;
private Integer shipperid;
```

```
//abre uma sessão
Session sessao = HibernateUtil.getSessionFactory().openSession();
//recupera um objeto do tipo categoria
//torna um objeto persistente em um transiente
sessao.load(cliente, "CHOPS");
sessao.load(func, 9);
pedido.setCustomers(cliente);
pedido.setEmployees(func);
//altera o valor do atributo
sessao.save(pedido);
//abre uma transação
Transaction tr = sessao.beginTransaction();
tr.commit();
```





```
private String customerid;
private String companyname;
private String contactname;
private String contacttitle;
private String address;
private String city;
private String region;
private String postalcode;
private String country;
private String country;
private String phone;
private String fax;
```



```
public static void main(String[] args) throws Exception{
    //instanciar um novo objeto professor, turmas e turmasId
    Professores prof = new Professores();
    Turmas turma = new Turmas();
    //abrir a sessão
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    //carrega o NOVO professor
   prof = (Professores) sessao.load(Professores.class, (short)1);
    System.out.print(prof.getNome());
    System.out.print(prof.getNroTurmas());
    Iterator i = prof.getTurmases().iterator();
    while(i.hasNext())
        turma = (Turmas) i.next();
        System.out.print(turma.getId().getCodigo());
    //fechar sessao
    sessao.close();
```



Consultas



HQL

- ☐ Hibernate Query Language
- □ Linguagem de consulta que se parece muito com a SQL
- □ A HQL é totalmente orientada a objeto, incluindo os paradigmas de herança, polimorfismo e encapsulamento.
- Executar os pedidos SQL sobre as classes de persistência do Java ao invés de tabelas no banco de dados, aumentando, assim, a distância entre o desenvolvimento da regras de negócio e o banco de dados.
- □ http://docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html



Consultas



CRITERIA

- □ API alternativa à HQL
- □ Voltado para consultas onde o número de parâmetros não é conhecido, como aquelas que contém N filtros
- Nesses casos, agiliza a consulta na base de dados.
- □ Consulta por Critério (QBC)
- □ Consulta por Exemplo (QBE)
- http://docs.jboss.org/hibernate/core/3.3/reference/en/html/querycriteria.html



Consulta Simples



- Selecionar todos os registros de professores
 - □ HQL

```
public static void main(String[] args) throws Exception{
    Professores prof = new Professores();
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    Iterator i = sessao.createQuery("from Professores").list().iterator();
    while(i.hasNext())
    {
        prof = (Professores) i.next();
        System.out.printf("%s\t%s\n", prof.getNome(), prof.getTitulacao());
    }
    sessao.close();
}
```



Consulta Simples

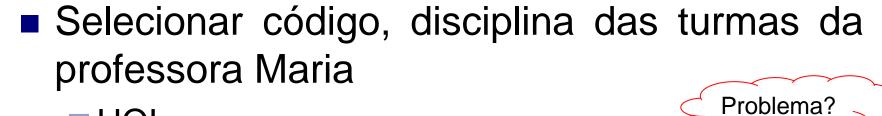


- Selecionar as turmas da professora Maria
 - □ SQL
 - SELECT * from Turmas, Professores WHERE Turmas.professor = Professores.matricula AND Professores.nome like 'MARIA%';
 - HQL

```
public static void main(String[] args) throws Exception{
   Turmas turma = new Turmas();
   Session sessao = HibernateUtil.getSessionFactory().openSession();
   Iterator i = sessao.createQuery("from Turmas where professores.nome like '%Maria%'").list().iterator();
   while(i.hasNext())
   {
      turma = (Turmas) i.next();
      System.out.printf("%s\t%s\n", turma.getId().getCodigo(), turma.getId().getDisciplina());
   }
   sessao.close();
}
```



Projeção



HQL

```
public static void main(String[] args) throws Exception{
   Turmas turma = new Turmas();
   Session sessao = HibernateUtil.getSessionFactory().openSession();
   Iterator i = sessao.createQuer("codigo, disciplina from Turmas where professores.nome like '%Maria%'").list().iterator();
   while(i.hasNext())
   {
     turma = (Turmas) i.next();
        System.out.printf("%s\t%s\n", turma.getId().getCodigo(), turma.getId().getDisciplina());
   }
   sessao.close();
}
```

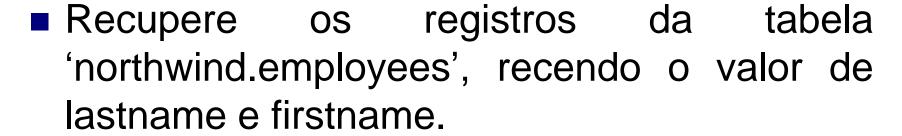


Projeção



- Selecionar código, disciplina das turmas da professora Maria
 - HQL







EXERCÍCIO



Criar a tabela conta_seuNome

Atributo	Domínio do Atributo	Extras
<u>numero</u>	Int	Not null
saldo	double	Not null

Povoar a tabela

<u>Numero</u>	Saldo
123	1000
234	800





- Criar uma transação que execute a seguinte operação:
 - □ Transfira R\$50,00 da conta 123 para a conta 234.