



UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II

CCT 620

Persistência
Aula 10

Vanessa Cristina Oliveira de Souza



UNIVERSIDADE FEDERAL DE ITAJUBÁ

HIBERNATE



Hibernate

■ Desenvolvimentos:

☐ Top Down

- Modelo de domínio -> MR

☐ Botton Up

- MR -> Modelo de domínio



Hibernate

1. 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



Hibernate

Inserir um objeto no banco

```
package universidade;
```

```
import Mapeamento.*;  
import org.hibernate.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        //Cria objeto disciplina  
        Disciplinas disc = new Disciplinas();  
        disc.setCodigo("SIT430");  
        disc.setNome("Data Warehouse");  
        disc.setCreditos("3");  
        //cria uma sessão  
        Session sessao = HibernateUtil.getSessionFactory().openSession();  
        //salva objeto disc - Objeto transiente  
        sessao.save(disc);  
        //abre uma transação  
        Transaction tr = sessao.beginTransaction();  
        //comita no banco o que foi salvo até agora  
        //disc passa a ser um objeto persistente  
        tr.commit();  
        //fecha a sessão  
        sessao.close();  
    }  
}
```



Hibernate

Inserir um objeto no banco

```
mysql> select * from disciplinas;
```

codigo	nome	creditos
CC0013	Fundamentos de Programacao	7
CCT620	Banco de Dados II	5
CSP410	Computador e Sociedade	4
SIF110	Estrutura de Dados I	4
SIT420	Laboratório de Banco de Dados	4

```
5 rows in set (0.00 sec)
```

```
mysql> select * from disciplinas;
```

codigo	nome	creditos
CC0013	Fundamentos de Programacao	7
CCT620	Banco de Dados II	5
CSP410	Computador e Sociedade	4
SIF110	Estrutura de Dados I	4
SIT420	Laboratório de Banco de Dados	4
SIT430	Data Warehouse	3

```
6 rows in set (0.00 sec)
```



Hibernate

Apagar um objeto do banco

```
package universidade;

import Mapeamento.*;
import org.hibernate.*;

public class Main {
    public static void main(String[] args) {
        //Cria objeto disciplina
        Disciplinas disc = new Disciplinas();
        //cria uma sessão
        Session sessao = HibernateUtil.getSessionFactory().openSession();
        //LOAD - Recupera um objeto do banco e salva em disc.
        //LOAD recebe o objeto e a chave primária
        //conversão de objeto persistente em transiente.
        sessao.load(disc, "SIT430");
        //apaga o objeto do banco
        sessao.delete(disc);
        //abre uma transação
        Transaction tr = sessao.beginTransaction();
        //comita no banco o que foi salvo até agora
        //disc passa a ser um objeto persistente
        tr.commit();
        //fecha a sessão
        sessao.close();
    }
}
```




Hibernate

```
mysql> select * from disciplinas;
```

codigo	nome	creditos
CC0013	Fundamentos de Programacao	7
CCT620	Banco de Dados II	5
CSP410	Computador e Sociedade	4
SIF110	Estrutura de Dados I	4
SIT420	Laboratório de Banco de Dados	4
SIT430	Data Warehouse	3

```
6 rows in set (0.00 sec)
```

```
mysql> select * from disciplinas;
```

codigo	nome	creditos
CC0013	Fundamentos de Programacao	7
CCT620	Banco de Dados II	5
CSP410	Computador e Sociedade	4
SIF110	Estrutura de Dados I	4
SIT420	Laboratório de Banco de Dados	4

```
5 rows in set (0.00 sec)
```



Exemplo

- Cadastrar uma nova turma
 - Disciplina “CCO013”
 - Código “Teste”
 - Vagas 30
 - Professor Maria -> matricula 1

```
mysql> select * from turmas;
+-----+-----+-----+-----+
| disciplina | codigo      | vagas | professor |
+-----+-----+-----+-----+
| CCT620    | CCO_2007_1  | 25    | 2         |
| CSP410    | CCO_2008_1  | 25    | 2         |
| SIT420    | SIT_207_1   | 20    | 1         |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```



Exemplo

- Cadastrar uma nova turma
 - Disciplina “CCO013”
 - Código “Teste”
 - Vagas 30
 - Professor Maria -> matricula 1

```
public class Turmas implements java.io.Serializable {
```

```
    private TurmasId id;  
    private Disciplinas disciplinas;  
    private Professores professores;  
    private Short vagas;
```



Hibernate

Inserir uma nova Turma

```
public class Main {  
    public static void main(String[] args) {  
        //Cria objeto TurmasId já inicializado com os valores de referência.  
        TurmasId id = new TurmasId("CC0013", "Teste");  
        //Cria objeto disciplina  
        Disciplinas disc = new Disciplinas();  
        //Cria objeto professor  
        Professores prof = new Professores();  
        //Cria objeto turma  
        Turmas turma = new Turmas();  
        //cria uma sessão  
        Session sessao = HibernateUtil.getSessionFactory().openSession();  
        //Recupera a disciplina que é FK  
        sessao.load(disc, "CC0013");  
        //Recupera o professor que é FK  
        sessao.load(prof, (short)1);  
        //Atualizar os valores do objeto turma  
        turma.setId(id);  
        turma.setDisciplinas(disc);  
        turma.setProfessores(prof);  
        turma.setVagas((short)30);  
        //salva o objeto  
        sessao.save(turma);  
        //abre uma transação  
        Transaction tr = sessao.beginTransaction();  
        //comita no banco o que foi salvo até agora  
        tr.commit();  
        //fecha a sessão  
        sessao.close();  
    }  
}
```



Hibernate

Inserir uma nova Turma

```
mysql> select * from turmas;
```

disciplina	codigo	vagas	professor
CCT620	CCO_2007_1	25	2
CSP410	CCO_2008_1	25	2
SIT420	SIT_207_1	20	1

```
3 rows in set (0.00 sec)
```

```
mysql> select * from turmas;
```

disciplina	codigo	vagas	professor
CCO013	Teste	30	1
CCT620	CCO_2007_1	25	2
CSP410	CCO_2008_1	25	2
SIT420	SIT_207_1	20	1

```
4 rows in set (0.00 sec)
```



Update

- Alterar a titulação do professor João para Doutor
 - Matricula 2

```
public static void main(String[] args) throws Exception{
    //instanciar um novo objeto professor
    Professores prof = new Professores();
    //abrir a sessão
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    //iniciar a transação
    Transaction tr = sessao.beginTransaction();
    //carrega o professor
    prof = (Professores) sessao.load(Professores.class, (short)2);
    //altera o objeto
    prof.setTitulacao("Doutor");
    //update
    sessao.update(prof);
    //comitt
    tr.commit();
    //fechar a sessão
    sessao.close();
}
```



saveOrUpdate

- Alterar o professor da turma CCO013/Teste para o João
 - Matricula 2

O que acontece?

```
public static void main(String[] args) throws Exception{
    //instanciar um novo objeto professor, turmas e turmasId
    Professores prof = new Professores();
    Turmas turmas = new Turmas();
    TurmasId id = new TurmasId();
    //abrir a sessão
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    //iniciar a transação
    Transaction tr = sessao.beginTransaction();
    //carrega o NOVO professor
    prof = (Professores) sessao.load(Professores.class, (short)2);
    //setar o id da turma
    id.setCodigo("Teste");
    id.setDisciplina("CCO013");
    turmas.setId(id);
    //setar o novo professor da turma
    turmas.setProfessores(prof);
    //save or update
    sessao.saveOrUpdate(turmas);
    //commit
    tr.commit();
    //fechar sessao
    sessao.close();
}
```



Exercício

- Montar um novo projeto usando hibernate e o banco northwind
- Cadastrar um novo pedido (northwind.order)



Exemplo

- Escrever na tela os seguintes dados da professora Maria
 - ☐ Nome
 - ☐ Número de turmas
 - ☐ Código das turmas



Exemplo

```
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.getTurmas().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

- Selecionar código, disciplina das turmas da professora Maria

□ HQL

Problema?

```
public static void main(String[] args) throws Exception{
    Turmas turma = new Turmas();
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    Iterator i = sessao.createQuery("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

```
public static void main(String[] args) throws Exception{
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    Iterator i = sessao.createQuery("Select id.codigo, id.disciplina "
        + "from Turmas where professores.nome like '%Maria%'").list().iterator();
    while(i.hasNext())
    {
        Object[] turma = (Object []) i.next();
        System.out.print(turma[0].toString());
    }
    sessao.close();
}
```




Exercício

- Crie uma função onde o usuário possa recuperar o nome de todos os produtos do pedido 10252.



Exercício

- Na tabela `northwind.products`, crie as chaves estrangeiras para os atributos `supplierid` e `categoryid`.
- Faça um pequeno programa que realiza um CRUD nessa tabela, usando:
 - ☐ JDBC
 - ☐ Hibernate
 - ☐ **No update, pode considerar a alteração de apenas um campo da tabela para facilitar.*
 - ☐ **Da mesma forma, a consulta pode ser limitada a uma única chave primária.*
 - ☐ **Não se preocupe com a interface!*