

Java Enterprise Edition - JEE

12. Introdução ao Hibernate 3.0



Esp. Márcio Palheta
gtalk: marcio.palheta@gmail.com



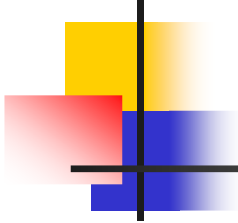
Agenda

- Usar a ferramenta Hibernate;
- Gerar as tabelas em um banco de dados qualquer a partir de suas classes de modelo;
- Desenvolver métodos para adicionar, listar, remover e procurar objetos no banco;
- Utilizar anotações para facilitar o mapeamento de classes para tabelas;
- Criar classes de DAO bem simples utilizando o hibernate.



Vantagens

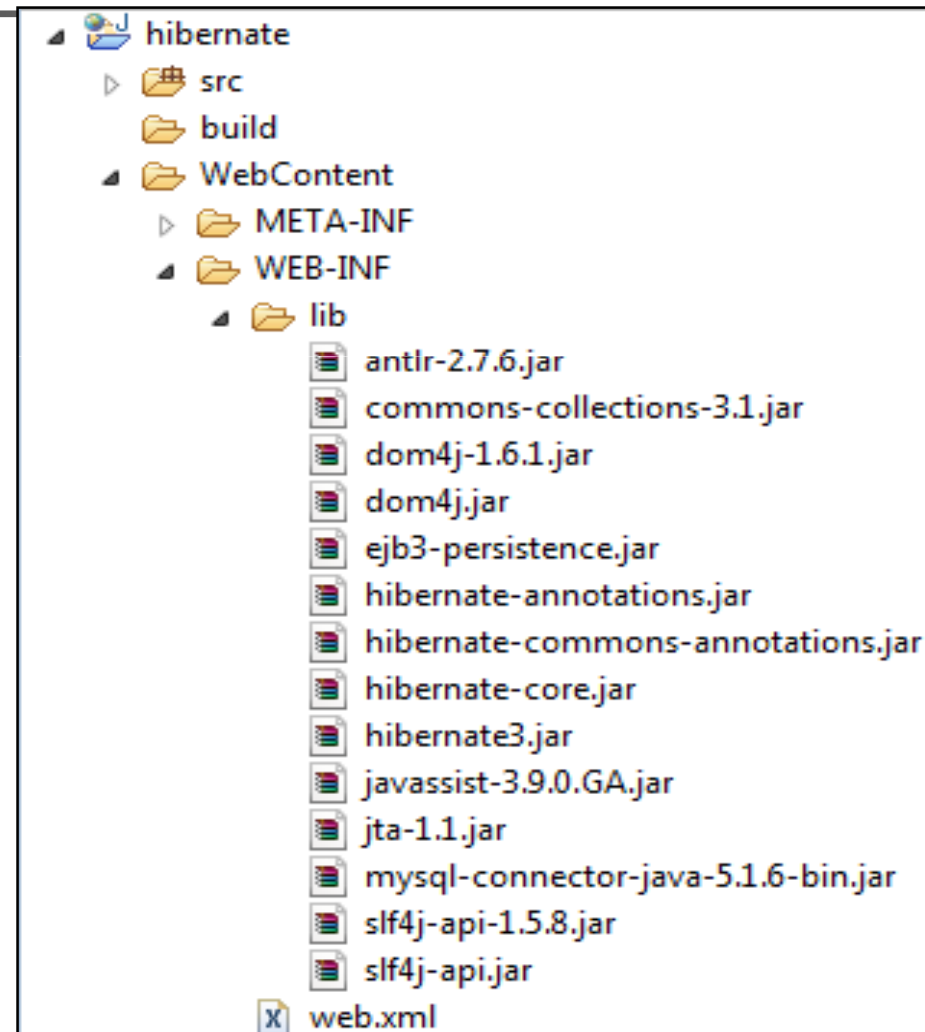
- O **Hibernate** abstrai o código **SQL** da nossa aplicação;
- Permite a mudança de base de dados sem alteração no código JAVA;
- Possui otimizações para códigos Select e Join mais elaborados;



Passos para criação e configuração do 1º projeto

- Criação do projeto WEB;
- Importação das bibliotecas do hibernate;
- Criação da classe de modelo;

Exercício 01: projeto web com as libs do hibernate





Classe Produto.java

```
package br.fucapi.hibernate.modelo;

public class Produto {
    private int id;
    private String nome;
    private String descricao;
    private double preco;

    public int getId() {..}
    public void setId(int id) {..}
    public String getNome() {..}
    public void setNome(String nome) {..}
    public String getDescricao() {..}
    public void setDescricao(String descricao) {..}
    public double getPreco() {..}
    public void setPreco(double preco) {..}
}
```



A mapeamento da classe Produto com Annotations

- A configuração da classe Produto.java é feita utilizando comentários especiais;
- **Annotations;**
- Diferenças entre Anotações e Comentários;
- Toda classe precisa de um campo que represente a chave primária;



Exercício 02: Produto.java

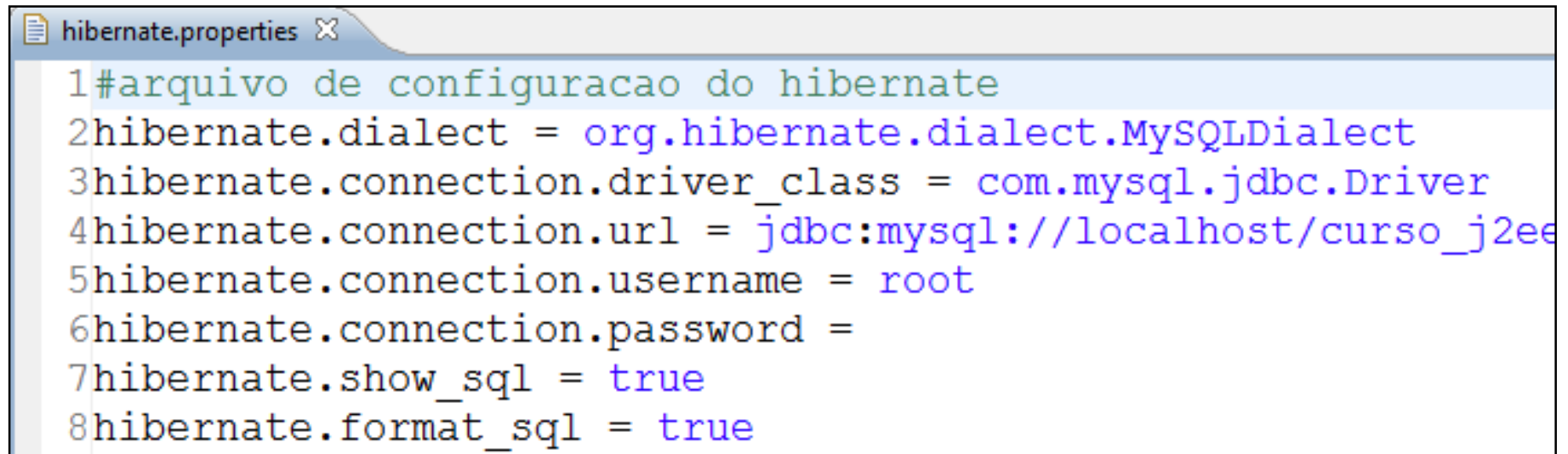
```
package br.fucapi.hibernate.modelo;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity //define a classe como entidade
public class Produto {
    @Id //define a chave primária
    @GeneratedValue //define autoincremento
    private int id;
    private String nome;
    private String descricao;
    private double preco;

    @Override
    public String toString() {
        return nome+" - "+descricao+" - "+preco;
    }
    //Metodos de Get e Set
}
```


Exercício 03: Criação do arquivo de configuração

- Na pasta `src` do seu projeto, crie o arquivo `hibernate.properties`, com os seguintes registros:

A screenshot of a text editor window titled 'hibernate.properties'. The window contains eight lines of configuration code for Hibernate, using a syntax-highlighted font. The first line is a comment in green. The subsequent lines define the dialect, driver class, connection URL, username, password, and SQL display settings.

```
1#arquivo de configuracao do hibernate
2hibernate.dialect = org.hibernate.dialect.MySQLDialect
3hibernate.connection.driver_class = com.mysql.jdbc.Driver
4hibernate.connection.url = jdbc:mysql://localhost/curso_j2ee
5hibernate.connection.username = root
6hibernate.connection.password =
7hibernate.show_sql = true
8hibernate.format_sql = true
```



Criação de tabelas

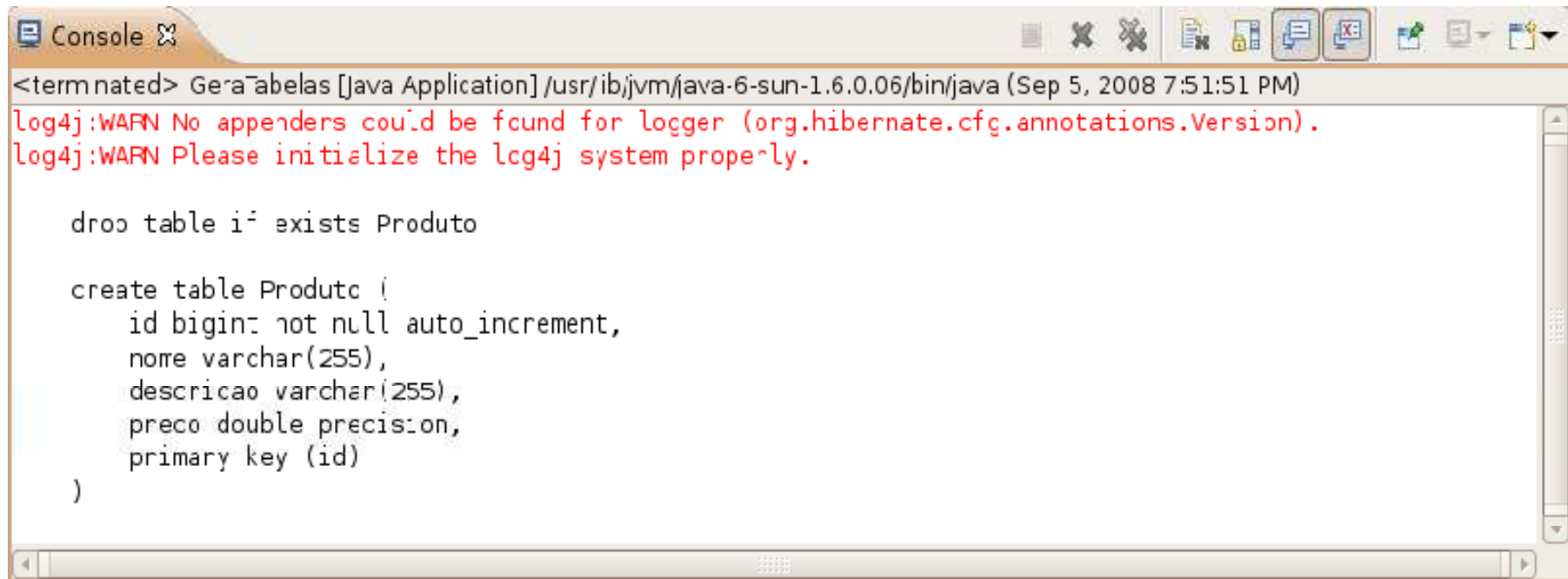
- O hibernate oferece ferramentas para criação de **tabelas**, a partir das classes mapeadas no java;
- A classe **AnnotationConfiguration** é responsável por carregar as tags do arquivo **hibernate.properties**;
- Método **addAnnotatedClass()** é utilizado para indicarmos quais classes devem ser persistidas;



Exercício 04: Criação de tabelas a partir de classes java

```
public class GeraTabelas {  
    public static void main(String[] args) {  
        //Cria uma configuração para acesso ao BD  
        AnnotationConfiguration cfg = new AnnotationConfiguration();  
        //Inclusao de classes mapeadas  
        cfg.addAnnotatedClass(Produto.class);  
        //Criacao do objeto de execucao  
        SchemaExport schema = new SchemaExport(cfg);  
        //Exceuta a criacao de tabelas  
        schema.create(true, true);  
    }  
}
```

Resultado da classe GeraTabelas.java



```
<terminated> GeraTabelas [Java Application] /usr/lib/jvm/java-6-sun-1.6.0.05/bin/java (Sep 5, 2008 7:51:51 PM)
log4j:WARN No appenders could be found for logger (org.hibernate.cfg.annotations.Version).
log4j:WARN Please initialize the log4j system properly.

drop table if exists Produto

create table Produto (
  id bigint not null auto_increment,
  nome varchar(255),
  descricao varchar(255),
  preco double precision,
  primary key (id)
)
```

Resultado do processamento

The screenshot shows the phpMyAdmin interface. On the left sidebar, the database 'curso_jee (2)' is selected, and its tables 'empresa' and 'produto' are listed. The 'produto' table is highlighted with an orange circle and a red '1'. The main panel shows the 'Estrutura' (Structure) tab for the selected database. It displays a table list with columns for selection, table name, and actions. The tables 'empresa' and 'produto' are listed. Below the table list, there is a summary row for '2 tabela(s)' and a 'Soma' (Sum) row. At the bottom, there is a link to 'Marcar todos / Desmarcar todos'.

	Tabela	Ação					
<input type="checkbox"/>	empresa						
<input type="checkbox"/>	produto						
2 tabela(s)		Soma					

↑ [Marcar todos / Desmarcar todos](#)



Sessões

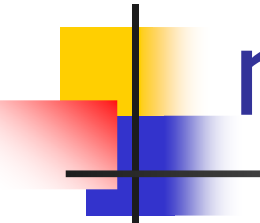
- O Hibernate está baseado no uso de sessões de banco de dados;
- Cada sessão criada é responsável por conectar ao banco de dados, pesquisar e localizar objetos;
- A seguir, criaremos a classe `HibernateUtil`, responsável pela criação de sessões;



Exercício 05: HibernateUtil

```
public class HibernateUtil {  
    private static SessionFactory factory;  
    //Bloco estático para criação da configuração inicial  
    static{  
        // Cria configuração para acesso ao BD  
        AnnotationConfiguration config = new AnnotationConfiguration();  
        // Indica as classes mapeadas da aplicação  
        config.addAnnotatedClass(Produto.class);  
        // Cria a factory de sessions  
        factory = config.buildSessionFactory();  
    }  
    public static Session getSession(){  
        //retorna uma sessão  
        return factory.openSession();  
    }  
}
```

Exercício 06: Cadastro de novos objetos



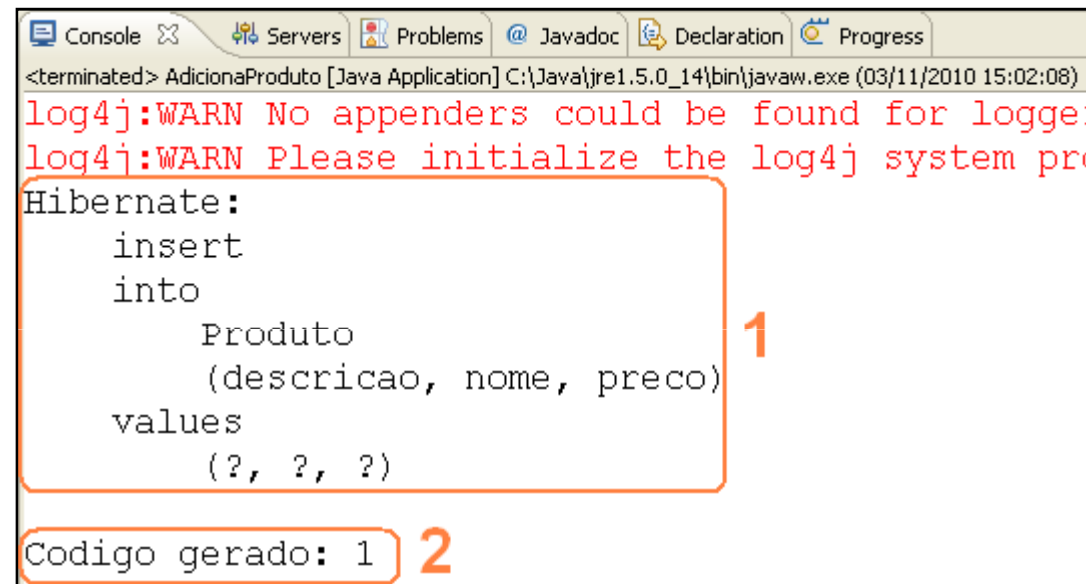
```
package br.fucapi.hibernate.util;

import org.hibernate.Session;
import br.fucapi.hibernate.modelo.Produto;

public class AdicionaProduto {

    public static void main(String[] args) {
        //Criação de um objeto java
        Produto novo = new Produto();
        novo.setNome("Camisa social");
        novo.setDescricao("Camisa masculina M");
        novo.setPreco(100.50);
        //Criação de uma sessão
        Session session = new HibernateUtil().getSession();
        //Envio do objeto para o banco de dados
        session.save(novo);
        System.out.println("Codigo gerado: "+novo.getId());
        //Encerramento da sessão
        session.close();
    }
}
```


Resultado gerado no console do eclipse e no MySql:



The screenshot shows the Eclipse IDE console window. At the top, there are tabs for Console, Servers, Problems, Javadoc, Declaration, and Progress. The console output includes a terminated message for 'AdicionaProduto', two log4j warnings about appenders and initialization, and an SQL insert statement from Hibernate. The insert statement is highlighted with an orange box and labeled with a red '1'. Below the SQL statement, the text 'Codigo gerado: 1' is highlighted with an orange box and labeled with a red '2'.



```
<terminated> AdicionaProduto [Java Application] C:\Java\jre1.5.0_14\bin\javaw.exe (03/11/2010 15:02:08)
log4j:WARN No appenders could be found for logger
log4j:WARN Please initialize the log4j system properly
Hibernate:
    insert
    into
        Produto
        (descricao, nome, preco)
    values
        (?, ?, ?)

Codigo gerado: 1
```



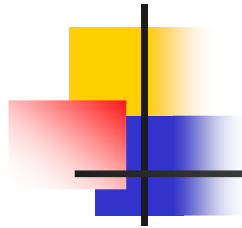
The screenshot shows the MySQL database interface. At the top, there is a '+ Opções' button. Below it is a table with columns 'id', 'descricao', 'nome', and 'preco'. The table contains one record with 'id' 1, 'descricao' 'Camisa regata G', 'nome' 'Camisa regata', and 'preco' 100.5. The record is highlighted with an orange box and labeled with a red '3'. Below the table, there are buttons for 'Marcar todos' and 'Desmarcar todos', and a 'Com marcados:' section with a pencil icon and a red 'X' icon. At the bottom, there is a 'Mostrar:' button, a text box with '30', and a label 'registro(s) começando de 0'. Below that, there is a 'no modo' dropdown menu set to 'horizontal', and a label 'e repetindo cabeçalhos após 100 células'.

	id	descricao	nome	preco
<input type="checkbox"/>	1	Camisa regata G	Camisa regata	100.5

Marcar todos / Desmarcar todos Com marcados:   

Mostrar: 30 registro(s) começando de 0

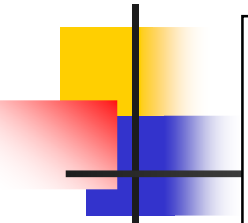
no modo horizontal e repetindo cabeçalhos após 100 células



Uso de padrões de projeto

- Padrões de projeto;
- Java Database Connectivity – JDBC;
- Data Access Object – DAO;
- Mapeamento Objeto-Relacional;
- Combinação de Hibernate e DAOs

Exercício 07: ProdutoDAO.java



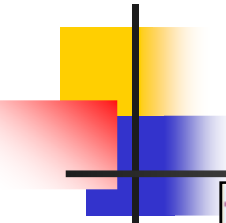
```
1 package br.fucapi.hibernate.modelo;
2
3 import org.hibernate.Session;
4
5 public class ProdutoDAO {
6     private Session session;
7     public ProdutoDAO(Session session) {
8         this.session = session;
9     }
10    public void cadastrar(Produto produto) {
11        this.session.save(produto);
12    }
13    public void alterar(Produto p) {
14        this.session.update(p);
15    }
16    public void excluir(Produto produto) {
17        this.session.delete(produto);
18    }
19    public Produto consultar(Long id) {
20        return (Produto)
21            this.session.load(Produto.class, id);
22    }
23 }
```

Exercício 08:

Inclusão do método **listar()**

```
public List<Produto> listar(){  
    //Criação da lista de retorno  
    List<Produto> listaResultado = null;  
    //Criação do objeto de consulta  
    1 Query query = null;  
    query = session.createQuery(  
        "from "+Produto.class.getName());  
    2 //Execução da consulta  
    listaResultado = query.list();  
    //Retorno da coleção resultado  
    return listaResultado;  
}
```

Exercício 08: Testes de persistência



```
public class TesteProdutoDAO {  
    public static void main(String[] args) {  
        1 //Criação da sessão de acesso ao banco de dados  
        Session session = new HibernateUtil().getSession();  
  
        2 //Criação do objeto DAO  
        ProdutoDAO dao = new ProdutoDAO(session);  
  
        3 //Chamada ao método listar()  
        List<Produto> lista = dao.listar();  
  
        4 //Impressão dos produtos  
        for (Produto produto : lista) {  
            System.out.println(produto);  
        }  
  
        5 session.close();  
    }  
}
```



Resultado da execução

```
log4j:WARN No appenders could be found for log  
log4j:WARN Please initialize the log4j system
```

```
Hibernate:
```

```
select  
    produto0_.id as id0_,  
    produto0_.descricao as descricao0_,  
    produto0_.nome as nome0_,  
    produto0_.preco as preco0_  
from  
    Produto produto0_
```

1

```
Lista de produtos:
```

```
Camisa regata - Camisa regata G: 100.5  
Bermuda masculina - Bermuda M: 50.0
```

2



O que vem a seguir?

- Revisão da arquitetura MVC;
- Revisão de struts;
- Revisão de hibernate;
- Arquitetura MVC com JSP, JSTL, Struts, DAOs e Hibernate;



Referências

- Hall, Marty, "Core Servlets and Java Server Pages", Janeiro 2002, Sun Microsystems Press;
- <http://java.sun.com/>
- <http://java.sun.com/j2ee/1.6/docs/tutorial/doc/index.html>
- <http://java.sun.com/products/jndi/docs.html>
- <http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html>

Java Enterprise Edition - JEE

12. Introdução ao Hibernate 3.0



Esp. Márcio Palheta
gtalk: marcio.palheta@gmail.com