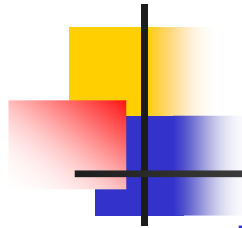


Java Enterprise Edition - JEE

04. Java Database Connectivity (JDBC)

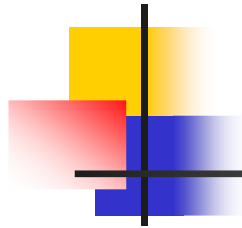


Esp. Márcio Palheta
gtalk: marcio.palheta@gmail.com



Agenda

- A base de dados MYSQL
- Conexão com banco de dados - Sockets
- Uma conexão Java
- Fábrica de Conexões
- Um pouco sobre Design Patterns
- Relembrando JavaBeans
- Data Access Object - DAO
- Cadastrar(add), alterar, excluir(remove), listar
- Exercícios de fixação – Consultar



A base de dados MYSQL

- É um banco de dados leve, relacional e não orientado a objetos;
- É amplamente utilizado no desenvolvimento de aplicações WEB;
- Será utilizado em nosso curso para que possamos simular situações reais de ambientes corporativos;

Ferramenta de administração

- Utilizaremos o projeto EasyPHP que disponibiliza MySQL, PhpMyAdmin e o servidor Apache;
- Verifique se o servidor está on-line:



- <http://localhost:8888/home/mysql/>

Administração MySQL



The screenshot displays the phpMyAdmin 3.4.5.1 web interface for a MySQL server on localhost. The interface is organized into several sections:

- Top Navigation Bar:** Includes the phpMyAdmin logo, the text "localhost / localhost | phpMyAdmin 3....", and a series of icons for database management tasks: "Banco de Dados" (Database), "SQL", "Status", "Variáveis" (Variables), "Conjuntos de caracteres" (Character sets), "Engines", "Privilégios" (Privileges), "Log binário" (Binary log), "Processos", "Exportar" (Export), and "Importar" (Import).
- Left Sidebar:** Contains the "Sem bases" (No databases) message and a set of icons for navigation.
- Main Content Area:**
 - Actions:** A section for performing database actions, including "MySQL localhost". It features a "Criar novo Banco de Dados" (Create new database) button, a text input field, a "Collation" dropdown menu, and a "Criar" (Create) button. Below this is a "Collation de conexão do MySQL:" (MySQL connection collation) dropdown menu set to "utf8_general_ci".
 - Interface:** A section for configuring the interface, including a "Linguagem - Language" (Language) dropdown menu set to "Português - Brazilian portuguese".
 - MySQL:** A section displaying server information:
 - Servidor: MySQL host info: localhost via TCP/IP
 - Versão do Servidor: 5.1.35-community-log
 - Versão do Protocolo: 10
 - Usuário: root@localhost
 - Conjunto de caracteres MySQL: UTF-8 Unicode (utf8)
 - Web server:** A section displaying web server information:
 - Apache/2.2.11 (Win32)
 - PHP/5.3.0



Banco de dados

- O processo de captura e envio de dados para um banco, é chamado de **persistência**;
- JAVA utiliza a api **JDBC** como padrão para acesso às bases de dados;
- Outra forma de acesso à base, é a utilização de projetos de Object Relational Mapping **ORM**, como o **Hibernate**;

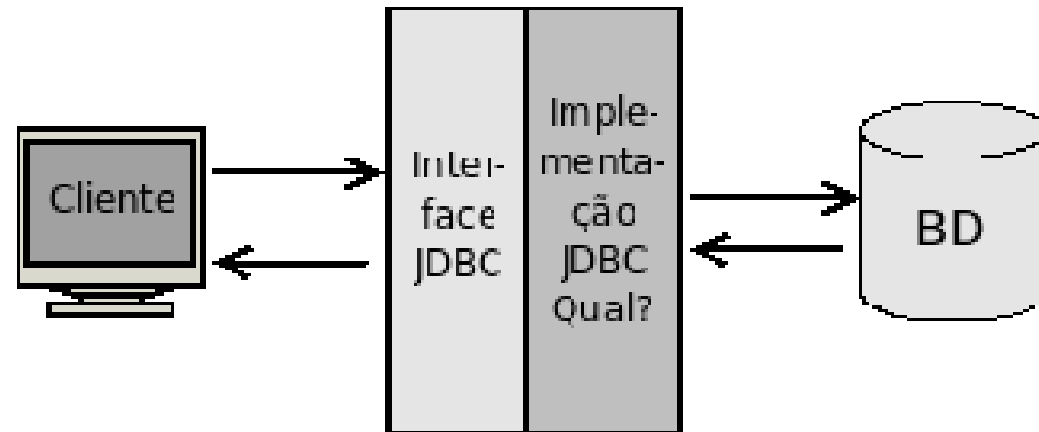
Java Database Connectivity (JDBC)



- O primeiro passo para acesso ao banco de dados é: **estabelecer uma conexão** entre a aplicação e o banco;
- A API JDBC surge para **encapsular** os métodos comuns de acesso a dados;
- A seguir, é exibida a estrutura de comunicação entre a aplicação(cliente) e o banco de dados(BD):

Conexão JDBC

- A conexão utiliza uma interface JAVA JDBC para acessar um BD:



- Cada fabricante de banco de dados cria uma implementação da interface JAVA JDBC para acesso ao seu BD;



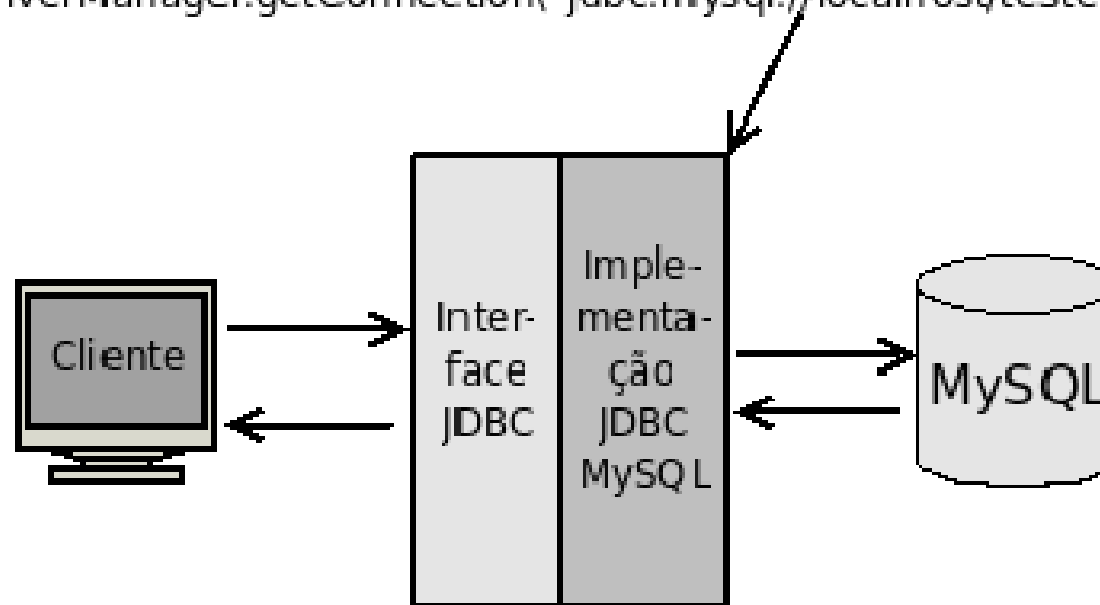
Estabelecendo a conexão

- Precisamos informar que implementação da API JDBC deve ser utilizada:
`Class.forName("com.mysql.jdbc.Driver");`
- De posse da implementação, podemos solicitar uma conexão de acesso ao banco de dados utilizado pela aplicação;
- A criação das conexões com o banco de dados fica a cargo do **Gerente de Drivers**, apresentado a seguir;

O controle das conexões

- O “Gerente de drivers” é responsável por executar a criação de conexões com a base de dados: **DriverManager**

```
DriverManager.getConnection("jdbc:mysql://localhost/teste");
```





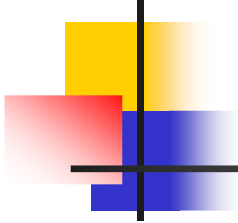
Código para conexão ao BD

```
public static void main(String[] args) {  
    //Bloco de tratamento de exceção  
    try {  
        //Carrega o driver JDBC para MySQL  
        Class.forName("com.mysql.jdbc.Driver");  
        //Criação da conexão com o BD  
        Connection connection =  
            DriverManager.getConnection(  
                "jdbc:mysql://nome_servidor/nome_BD",  
                "login", "senha");  
        System.out.println("Conexão realizada");  
        //Fechamento da conexão  
        connection.close();  
    } catch (ClassNotFoundException e) {  
        e.printStackTrace();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```



Itens a ponderar

- A criação de conexões é uma atividade complexa, que requer processamento;
- Como seriam criadas conexões em aplicações do dia-a-dia?
- Sempre replicar esse código em classes que acessam BD?
- O que fazer?



Uso de padrões de projeto: Connection Factory

- A fim de facilitar a criação de conexões, é recomendado que seja criada uma “Fábrica de Conexões”, mais conhecida como **Connection Factory**;
- Com isso, a fábrica passa a encapsular o controle e criação das conexões;
- Facilita a manutenção e a mudança de base de dados;



Código Connection Factory

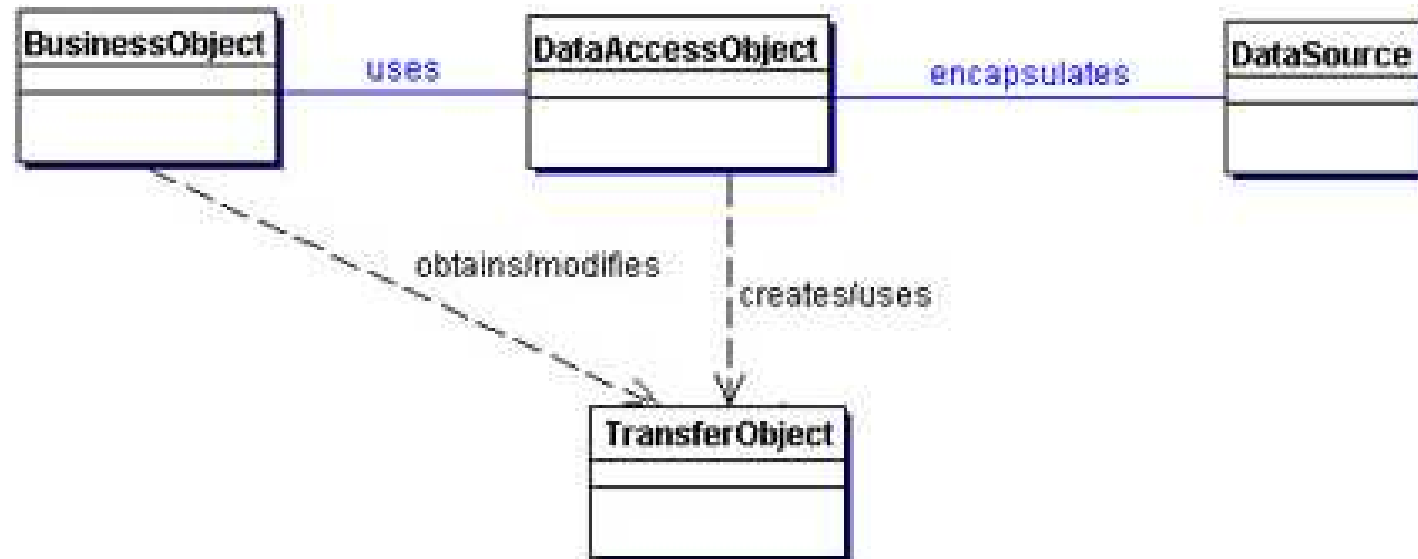
```
public class ConnectionFactory {  
    1 public static Connection getConnection() {  
        Connection resultado = null;  
        try {  
            2 //Carregar o driver na memoria  
              Class.forName("com.mysql.jdbc.Driver");  
  
            3 //Criar conexão  
              resultado = DriverManager.getConnection(  
                  "jdbc:mysql://nome_servidor/nome_bd",  
                  "login", "senha");  
        } catch (ClassNotFoundException e) {  
            e.printStackTrace();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
        4 //Devolver a conexão  
        return resultado;  
    }  
}
```



Camada de acesso a dados – Data Access Object(DAO)

- Uso de aplicações OO com banco de dados relacionais;
- **DAO** (Data Access Object) é um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados;
- Em aplicações que utilizam MVC as **funcionalidades de banco de dados** são implementadas por classes DAO;

Diagrama de classes





Camada de acesso a dados – Data Access Object(DAO)

- Com o uso de classes DAO, isolamos o restante da aplicação do acesso aos dados;
- A camada de negócios solicita serviços de persistência da camada de modelo;
- Referência:
<http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>



Design Patterns

- Após o surgimento da Arquitetura JEE, a SUN identificou pequenos problemas estruturas em projetos WEB;
- A fim de solucioná-los, a SUN propôs uma série de padrões de projetos, que foram rapidamente aceitos pelo mercado;
- Essas estruturas ficaram conhecidas como **Design Patterns**



Design Patterns - Factory

- Nossa classe **ConnectionFactory.java** implementa o padrão **Factory**;
- Segundo o Design Pattern Factory, devemos manter o encapsulamento da construção (fabricação) de objetos complexos;



Meu 1º projeto JDBC:

Estudo de caso – Empretech

- A Empretech é uma consultora especializada na administração de conglomerados empresariais.
- Inicialmente, a Empretech deseja automatizar o cadastro de empresas que administra, informando:
 - cnpj, razão social, endereço, telefone, site, email e nome do presidente;

Projeto Empretech

Lista de Atividades Iniciais

- Abaixo, serão listadas as atividades iniciais do projeto da Empretech:
- Criação do Banco de Dados e Tabelas;
- Criação dos componentes: Javabeans, Factory e DAO's;
- Realização de teste no método principal: `EmpresaDAO.main();`

Projeto Empretech: Criação do Banco de Dados

The screenshot shows the phpMyAdmin 3.2.0.1 interface in a Mozilla Firefox browser. The address bar shows the URL `http://localhost/home/mysql/`, which is highlighted with an orange box and a red number 1. The left sidebar shows the 'phpMyAdmin' logo and a list of icons. The main content area shows the 'Servidor: localhost' section with a tab for 'Banco de Dados' highlighted with an orange box and a red number 2. Below this, the 'Criar novo Banco de Dados' section is visible, showing a text input field with the name 'cursojee' (highlighted with an orange box and a red number 3), a 'Collation' dropdown menu, and a 'Criar' button (highlighted with an orange box and a red number 4).

localhost / localhost | phpMyAdmin 3.2.0.1 - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

`http://localhost/home/mysql/` 1

Mais visitados Primeiros passos Últimas notícias Processo de Seleção d... SQL Server

localhost / localhost | phpMyAdmin 3....

phpMyAdmin

Servidor: localhost

2 Banco de Dados SQL Status Variáveis Conju

Banco de Dados

Sem bases

3 Criar novo Banco de Dados ?

cursojee Collation 4 Criar

Projeto Empretech: Nova base gerada

The screenshot shows the phpMyAdmin web interface in a browser window. The address bar shows 'localhost / localhost / cursojee | p...'. The page title is 'Servidor: localhost' and the selected database is 'Banco de Dados: cursojee' (labeled with a red '1'). The left sidebar shows 'cursojee (0)' (labeled with a red '2') and the message 'Nenhuma tabela encontrada no Banco de Dados'. The main content area has tabs for 'Estrutura', 'SQL', 'Procurar', and 'Procurar por exemplo'. A green message box (labeled with a red '3') displays a success message: 'Database cursojee has been created.' Below this, the SQL command 'CREATE DATABASE `cursojee` ;' is shown. At the bottom, there is a section titled 'Criar nova tabela no Banco de Dados cursojee' with input fields for 'Nome:' and 'Número de arquivos:'.

localhost / localhost / cursojee | p... x Correio :: Caixa de Entrada (124) x

phpMyAdmin

Servidor: localhost Banco de Dados: cursojee 1

Estrutura SQL Procurar Procurar por exemplo

✓ Database cursojee has been created. 3

CREATE DATABASE `cursojee` ;

cursojee (0) 2

Nenhuma tabela encontrada no Banco de Dados

Nenhuma tabela encontrada no Banco de Dados

Criar nova tabela no Banco de Dados cursojee

Nome: Número de arquivos:

Projeto Empretech:

Criação da tabela Empresa

```
create table empresa(  
    id bigint not null auto_increment,  
    cnpj varchar(20),  
    razaosocial varchar(255),  
    endereco varchar(255),  
    telefone varchar(20),  
    site varchar(255),  
    email varchar(255),  
    dtcriacao date,  
    primary key (id));
```




Inclusão de registros na tabela

- insert into empresa(cnpj, razaosocial, endereco, telefone, site, email, dtcriacao) values ("123456", "FUCAPI", "AV. DANILO AREOSA", "21273053", "portal.fucapi.br", "sac@fucapi.br", "2011-07-21");
- insert into empresa(cnpj, razaosocial, endereco, telefone, site, email, dtcriacao) values ("987654", "MP TECH", "RUA VAI E VOLTA", "21271000", "www.mptech.com", "sac@mptech.com", "2011-07-21");

Projeto Empretech:

Execução do script **Empresa**

The screenshot shows the phpMyAdmin web interface. The browser tabs at the top include 'localhost / localhost / cursojee | p...' and 'Correio :: Caixa de Entrada (124)'. The interface has a sidebar on the left with the 'phpMyAdmin' logo and navigation icons. The main content area shows the 'cursojee' database selected, with a message stating 'Nenhuma tabela encontrada no Banco de Dados'. The 'SQL' tab is active, and the 'Estrutura' tab is also visible. The SQL query editor contains the following code:

```
create table empresa(  
    id bigint not null auto_increment,  
    cnpj varchar(20),  
    razaosocial varchar(255),  
    endereco varchar(255),  
    telefone varchar(20),  
    site varchar(255),  
);
```

Below the query editor, there is a checkbox for 'Gravar essa consulta SQL:' and a checkbox for 'Deixar qualquer usuário aces...'. At the bottom, there is a checkbox for 'Mostrar esta consulta SQL novamente' and an 'Executar' button. The interface is annotated with orange boxes and numbers: '1' is next to 'Banco de Dados: cursojee', '2' is next to 'Estrutura', '3' is next to the SQL query, and '4' is next to the 'Executar' button.

Projeto Empretech: Tabela Empresa, na base cursojee

The screenshot shows the phpMyAdmin web interface. The browser tabs at the top include 'localhost / localhost / cursojee | p...' and 'Correio :: Caixa de Entrada (124)'. The interface header indicates 'Servidor: localhost' and 'Banco de Dados: cursojee'. A navigation bar contains buttons for 'Estrutura', 'SQL', 'Procurar', 'Procurar por exemplo', and 'Exportar'. On the left sidebar, the 'cursojee (1)' database is selected, and the 'empresa' table is listed. The main area displays a green success message: 'Seu comando SQL foi executado com sucesso (Consulta levou 0.0058 segundos)'. Below this message, the SQL command is shown:

```
CREATE TABLE empresa(  
  id BIGINT NOT NULL AUTO_INCREMENT ,  
  cnpj VARCHAR( 20 ) ,  
  razasocial VARCHAR( 255 ) ,  
  endereco VARCHAR( 255 ) ,  
  telefone VARCHAR( 20 ) ,  
  site VARCHAR( 255 ) ,  
  email VARCHAR( 255 ) ,  
  nomepresidente VARCHAR( 255 ) ,
```

1

2

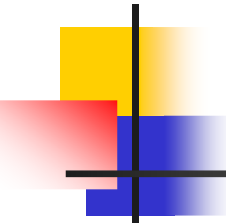


Projeto Empretech:

Criação do projeto no Eclipse

- No eclipse, selecione a perspectiva Java;
- Crie um novo projeto Java, com o nome **jdbc** e clique em **finish**;
- Copie a lib **mysql-connector-5.x.jar** para a pasta (...)/workspace/jdbc;
- No eclipse, selecione o projeto **jdbc** e clique **F5**;
- Clique com o botão direito no driver e selecione **Build Path** e **Add to Build Path**

Classe **ConnectionFactory.java**



```
1 package br.fucapi.curso.jee.model.connection;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5 public class ConnectionFactory {
6     public static Connection getConnection() {
7         Connection connection = null;
8         try {
9             //Definicao do driver JDBC
10             Class.forName("com.mysql.jdbc.Driver");
11             //Criacao da conexao
12             connection = DriverManager.getConnection(
13                 "jdbc:mysql://localhost/cursojee"
14                 "root", "");
15         } catch (ClassNotFoundException e) {
16             e.printStackTrace();
17         } catch (SQLException e) {
18             e.printStackTrace();
19         }
20         //Retorno da conexao gerada
21         return connection;
22     }
23 }
```

Projeto Empretech:

TESTE da Fábrica de conexões

- Classe para teste da ConnectionFactory

```
TesteJDBC.java X
1 package br.fucapi.curso.jee.model.connection; 1
2 import java.sql.Connection;
3 import java.sql.SQLException;
4 public class TesteJDBC {
5     public static void main(String[] args) throws SQLException {
6         System.out.println("INICIO do processamento");
7         2 Connection connection = ConnectionFactory.getConnection();
8         System.out.println("Conexão realizada com sucesso");
9         //Fechamento da conexao com o banco de dados
10        3 connection.close();
11        System.out.println("FIM do processamento");
12    }
13 }
```



Projeto Empretech: Criação de JavaBeans

- Relembrando: Javabeans são classes que, em geral, representam as tabelas do banco de dados;
- Possuem métodos de GET e SET;
- A seguir, vamos criar um Javabeans para representar a entidade Empresa;
- Use `br.fucapi.curso.jee.model.beans`;



Projeto Empretech: Entidade Empresa

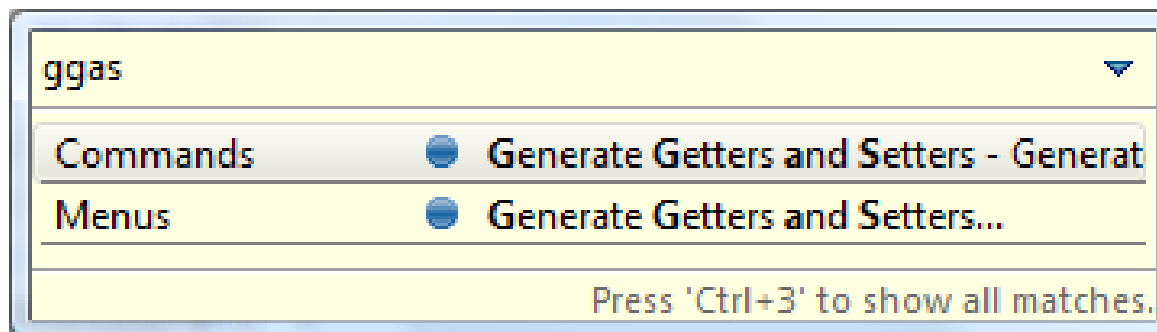
```
package br.fucapi.curso.jee.model.bean;
import java.util.Calendar;

public class Empresa {
    private Long id;
    private String cnpj;
    private String razaoSocial;
    private String endereco;
    private String telefone;
    private String site;
    private String email;
    private Calendar dataCriacao;
    @Override
    public String toString() {
        return cnpj+" - "+razaoSocial;
    }
    //Metodos de get e set aqui
```


Projeto Empretech:

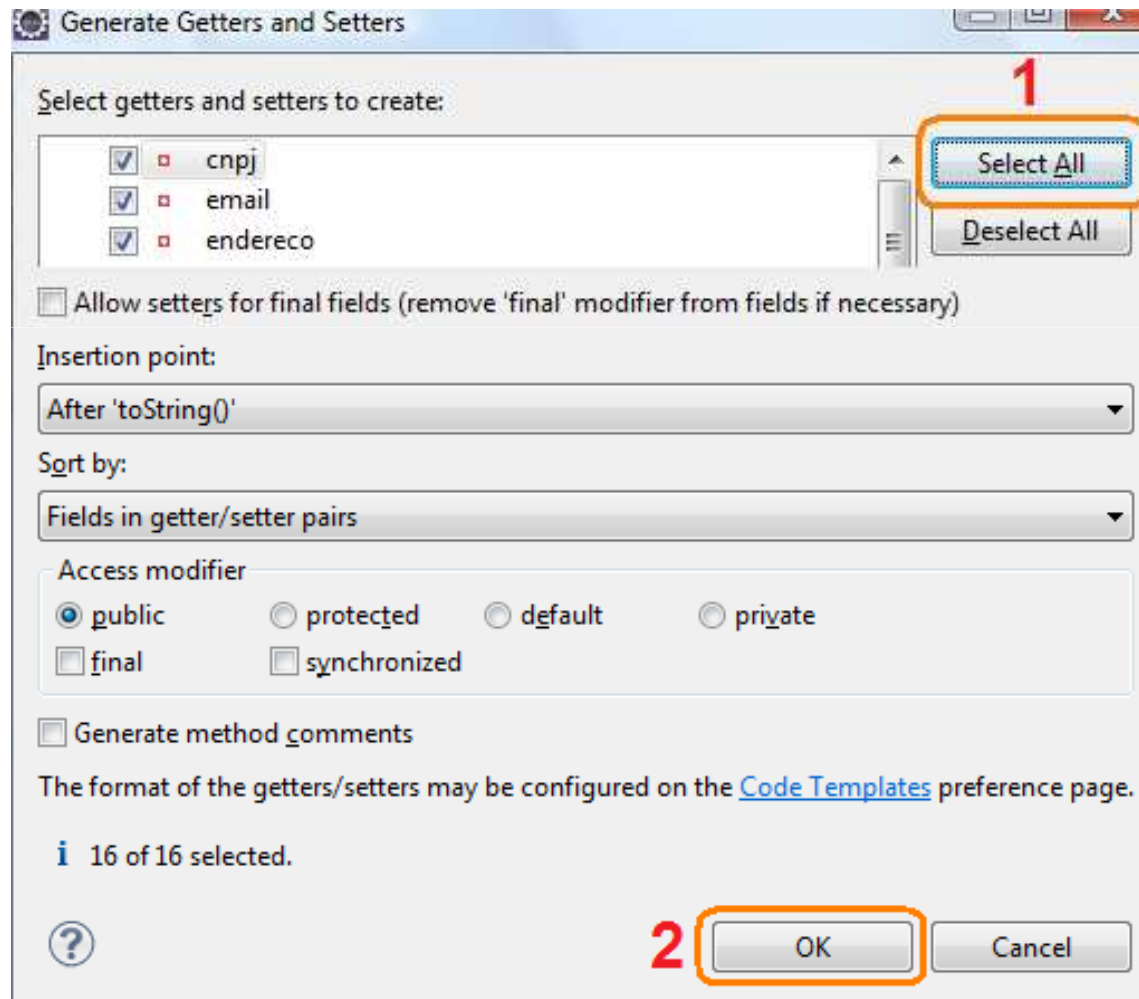
Entidade Empresa – get e set

- Gere os métodos de get e set;
- Para que o eclipse gere os métodos, pressione Ctrl+3 e, na tela de busca, digite **ggas**, conforme figura a seguir:



Projeto Empretech:

Tela de criação de métodos



Projeto Empretech:

Inclusão de novas empresa

- Revisão rápida – O que já foi criado?
- O Banco de dados `cursojee`;
- A tabela `empresa`;
- A `ConnectionFactory` responsável pela criação de conexões;
- O Javabeam `Empresa.java`;
- AGORA, vamos criar a classe `JDBCInsere`, responsável pelo envio de dados ao banco;



Classe de teste de cadastro

```
package br.fucapi.curso.jee.model.dao; 1

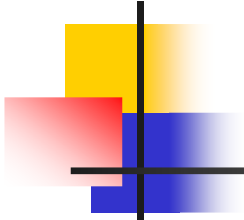
import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Calendar;
import br.fucapi.curso.jee.model.connection.ConnectionFactory;

public class JDBCInsere {
    public static void main(String[] args) throws SQLException {
        //Solicita a criacao de uma conexao
        Connection conn = ConnectionFactory.getConnection(); 3

        //Definicao do comando a ser executado
        String sql = "Insert into empresa(" +
            "cnpj, razaosocial, endereco, telefone, " +
            "site, email, dtcriacao) " +
            "values(?,?,?,?,?,?,?,?)"; 4

        //Continua...
```

Continuação do código...



```
//Continua...
1 //Criacao do objeto de execucao de comandos SQL
PreparedStatement stmt = conn.prepareStatement(sql);

2 //Carga dos parametros da instrucao SQL
stmt.setString(1, "123456789");
stmt.setString(2, "MPNews Corporation");
stmt.setString(3, "Rua Jutai, 20, DI");
stmt.setString(4, "2127-3831");
stmt.setString(5, "www.mpnews.com");
stmt.setString(6, "sac@mpnews.br");
stmt.setDate(7, new Date(
    Calendar.getInstance().getTimeInMillis()));

3 //Execucao da instrucao SQL
stmt.execute();

4 //Fechamento da conexao
stmt.close();
conn.close();
System.out.println("Operacao realizada");
}
}
```

Projeto Empretech: Conferência do dados

phpMyAdmin

Servidor: localhost ▶ Banco de Dados: cursojee ▶ Tabela: empresa

Visualizar Estrutura SQL Procurar Inserir Exportar

Importar Operações Limpar Eliminar

Mostrando registros 0 - 2 (3 total, Consulta levou 0.0007 segundos)

```
SELECT *
FROM `empresa`
LIMIT 0 , 30
```

☐ Profiling [[Editar](#)] [[Explicar SQL](#)] [[Criar código PHP](#)] [[Atualizar](#)]

Mostrar: 30 registro(s) começando de 0

no modo horizontal e repetindo cabeçalhos após 100 células

Ordenar pela chave: Nenhum

+ Opções

			id	cnj	razaosocial	endereco	telefone	site
<input type="checkbox"/>			1	123456	FUCAPI	AV. DANILO AREOSA	21273053	portal.fucapi.br
<input type="checkbox"/>			2	987654	MP TECH	RUA VAI E VOLTA	21271000	www.mptech.com
<input type="checkbox"/>			3	9988774455	MPNews Corporation	Rua Jutai, 20, Distrito	21273891	www.mpnews.com



Projeto Empretech:

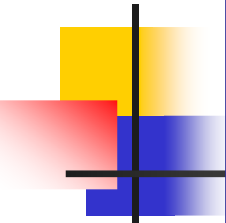
Data Access Object - DAO

- Representa um dos Design Patterns mais famosos e utilizados pelo mercado;
- A idéia é isolar todo o acesso a banco de dados em classes bem simples, cuja instância é um **objeto** responsável por **acessar dados**;
- Por padrão, o nome da classe de acesso à tabela de **Empresa** será **EmpresaDAO**
- Pacote: **br.fucapi.curso.jee.model.dao**;

Início da classe EmpresaDAO

```
EmpresaDAO.java x
1 package br.fucapi.curso.jee.model.dao;
2
3 import java.sql.Connection;
4 import java.sql.Date;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.Calendar;
10 import java.util.List;
11 import br.fucapi.curso.jee.model.bean.Empresa;
12 import br.fucapi.curso.jee.model.connection.ConnectionFactory;
13
14 public class EmpresaDAO {
15
16 //continua
```


EmpresaDAO.cadastrar()



```
public void cadastrar(Empresa empresa) {
    Connection conn = ConnectionFactory.getConnection();
    //Cria a instrução SQL a ser executada
    1 String sql = "Insert into empresa(cnpj, razaosocial, endereco,"
        + "telefone,site, email, dtcriacao) values(?,?,?,?,?,?,?)";
    PreparedStatement stmt = null;
    try {
        stmt = conn.prepareStatement(sql);
        stmt.setString(1, empresa.getCnpj());
        stmt.setString(2, empresa.getRazaoSocial());
        stmt.setString(3, empresa.getEndereco());
        stmt.setString(4, empresa.getTelefone());
        stmt.setString(5, empresa.getSite());
        stmt.setString(6, empresa.getEmail());
        stmt.setDate(7, new Date(empresa.getDataCriacao().
                                                                    getTimeInMillis()));
    2 //Execucao do comando SQL
      stmt.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
    3 //Encerramento da conexão
      closeConnection(conn, stmt);
    }
}
```



Projeto Empretech:

Lista de Empresas

- Quando precisamos realizar pesquisas no BD, também utilizamos a interface `PreparedStatement.executeQuery();`
- O objeto retornado é do tipo `ResultSet`;
- Para navegar entre os valores retornados, utilizamos o método `ResultSet.next();`
- A seguir, veremos o método `getLista()`



Método EmpresaDAO.listar()

```
public List<Empresa> listar() {  
    //Criação da conexão com o banco de dados  
    Connection connection = ConnectionFactory.getConnection();  
    //Criação do comando SQL  
    String sql = "Select * from empresa";  
    //Criação do objeto de execução de comandos SQL  
    PreparedStatement stmt = null;  
    1 //Criação da Collection a ser retornada  
    List<Empresa> lista = new ArrayList<Empresa>();  
    2 //Criação do objeto que recebe o resultado da execução SQL  
    ResultSet resultSet = null;  
  
    //Continua...
```

Continuação...

//Continua...

```
try {
    stmt = connection.prepareStatement(sql);
    resultSet = stmt.executeQuery();
    while (resultSet.next()) {
        Empresa empresa = new Empresa();
        //Carga do objeto Empresa
        empresa.setId(resultSet.getLong("id"));
        empresa.setCnpj(resultSet.getString("cnpj"));
        empresa.setRazaoSocial(resultSet.getString("razaosocial"));
        empresa.setEndereco(resultSet.getString("endereco"));
        empresa.setTelefone(resultSet.getString("telefone"));
        empresa.setSite(resultSet.getString("site"));
        empresa.setEmail(resultSet.getString("email"));
        Calendar data = Calendar.getInstance();
        data.setTime(resultSet.getDate("dtcriacao"));
        empresa.setDataCriacao(data);
        lista.add(empresa);
    }
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    closeConnection(connection, stmt);
}
return lista;
}
```

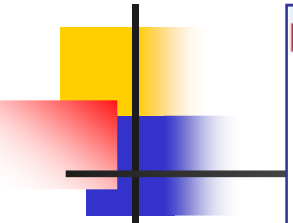


Projeto Empretech:

Métodos de acesso ao BD

- Após a criação da classe EmpresaDAO, implementamos os métodos de acesso ao banco: cadastrar e listar;
- Em aplicações comerciais, é comum a necessidade de alterar e excluir registros realizados;
- A seguir, serão exibidos mais métodos da classe **EmpresaDAO**: **alterar()** e **excluir()**

Método EmpresaDAO.alterar()



```
public void alterar(Empresa empresa) {
    Connection connection = ConnectionFactory.getConnection();
    String sql = "Update empresa set "
        + "cnpj = ?, razaosocial = ?, endereco = ?, "
        + "telefone = ?, site = ?, email = ?, dtcriacao = ? "
        + "Where id = ?";
    PreparedStatement stmt = null;
    try {
        stmt = connection.prepareStatement(sql);
        stmt.setString(1, empresa.getCnpj());
        stmt.setString(2, empresa.getRazaoSocial());
        stmt.setString(3, empresa.getEndereco());
        stmt.setString(4, empresa.getTelefone());
        stmt.setString(5, empresa.getSite());
        stmt.setString(6, empresa.getEmail());
        stmt.setDate(7, new Date(empresa.getDataCriacao().
                                getTimeInMillis()));
        stmt.setLong(8, empresa.getId());
        stmt.execute();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally{
        closeConnection(connection, stmt);
    }
}
```

1

2

3

Projeto Empretech:

Método EmpresaDAO.excluir()

```
164 public void excluir(long id){
165     //Criação da conexão
166     Connection connection = ConnectionFactory.getConnection();
167     //criação da String SQL a ser executada
168     1 String sql = "Delete from empresa Where id = ?";
169     //Criação do objeto a executar o comando SQL
170     PreparedStatement stmt = null;
171     try {
172         //Configuração do statement
173         stmt = connection.prepareStatement(sql);
174         //Carga dos parâmetros da instrução SQL
175         stmt.setLong(1, id);
176         2 //Execução do comando de exclusão
177         stmt.execute();
178     } catch (SQLException e) {
179         e.printStackTrace();
180     } finally{
181         3 //Encerramento da conexão
182         closeConnection(connection, stmt);
183     }
184 }
```



EmpresaDAO.closeConnection

- Método responsável por encerrar as conexões abertas:

```
private void closeConnection(Connection connection,
    PreparedStatement stmt) {
    try {
        1 // Fecha o statement
        stmt.close();

        2 // Fecha a conexão
        connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```




Exercícios

- Crie a classe **EmpresaDAOTeste** para testar os métodos de acesso criados;
- Atualize EmpresaDAO com o método:

```
public Empresa consultar(long id) {
```
- Para a entrada de dados, utilize:
 - String str;
 - str=JOptionPane.showInputDialog("Nome: ");
 - contato.setNome(str);



Atividades do projeto final

- **Análise de sistema**
- Refinamento dos requisitos;
- Diagrama de casos de uso;
- Diagrama de classes entidade;
- Diagrama navegacional;
- Diagrama de sequência;



O que vem a seguir?

- Servlet Container
- Servlets
- JSP - JavaServer Pages
- JSTL - JavaServer Pages Tag Library
- MVC
- Struts



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

04. Java Database Connectivity (JDBC)



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