




INFORMA, FORMA, TRANSFORMA.

Programação Desktop

Fabício Curvello Gomes



INFORMA, FORMA, TRANSFORMA.




Conexão ao Banco de Dados

2

Sistema

FIRJAN




INFORMA. FORMA. TRANSFORMA.

JDBC

Java Database Connectivity


É um conjunto de classes e interfaces escritas em Java que fazem o envio de instruções SQL para o banco de dados.



3

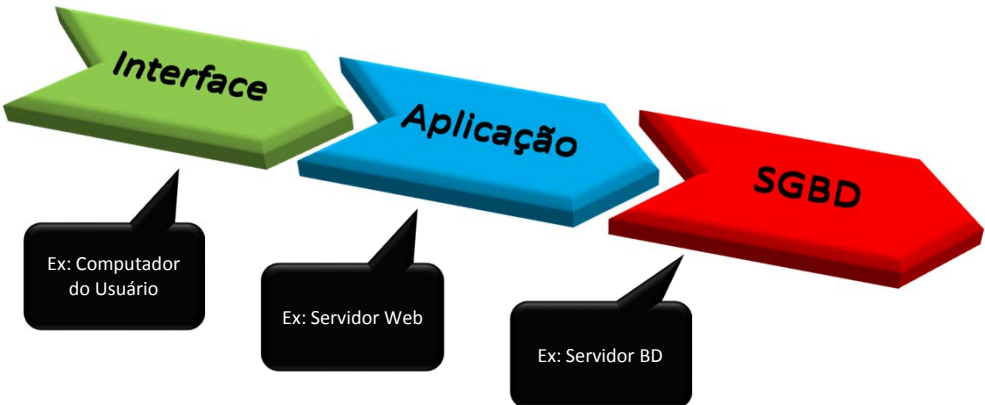
Sistema


FIRJAN





INFORMA. FORMA. TRANSFORMA.

Modelo de 3 Camadas






4

  INFORMA. FORMA. TRANSFORMA.

Classe *SQLException*

É a Classe que trata os erros que acontecem na camada SGBD. Seus métodos mais utilizados são:

- **getMessage()** – Retorna uma string descrevendo o erro ocorrido;
- **getErrorCode()** – Retorna um código específico do banco de dados que está sendo utilizado;
- **getSQLState()** – Retorna a instrução SQL que deu erro.



5

  INFORMA. FORMA. TRANSFORMA.

Conexão com SGBD





É necessário fazer o download do Framework que possui o driver do respectivo SGBD.

Endereço web:
<http://www.mysql.com/downloads/connector/j/>




6



  INFORMA. FORMA. TRANSFORMA.

Conexão com SGBD (Cont.)

- Abrir o navegador no site
<http://www.mysql.com/downloads/connector/j/>
- Clicar em download do **Connector/J**
- Selecione a plataforma “*Platform Independent*” e clique no download do tipo “*ZIP Archive*”
- Descompactar
- Copiar o arquivo **mysql-connector-java- . . . -bin.jar** e colar dentro da pasta da Workspace.




7

  INFORMA. FORMA. TRANSFORMA.

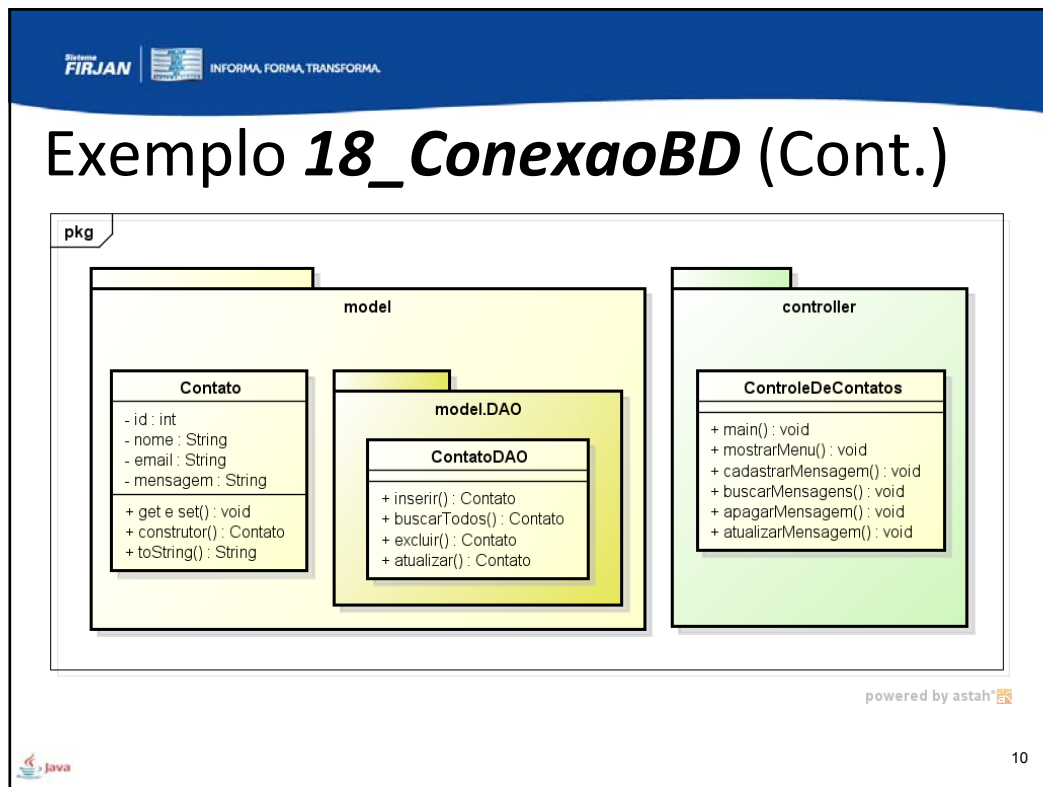
Exemplo **18_ConexaoBD**

Trabalharemos neste projeto, que será o nosso exemplo de utilização do SGBD MySQL para conexão com Java

- Crie no Eclipse o projeto **18_ConexaoBD**
- Nos próximos slides trabalharemos na criação de diversas Classes e pacotes dentro deste projeto, para exemplificarmos a utilização de conexão ao banco de dados.



8



Criando a referência ao conector:

1. No Eclipse, clicar com o **botão direito** do mouse sobre o projeto.
2. Selecionar a opção **Build Path / Configure Build Path...**
3. Clicar na aba **Libraries**, e depois no botão **Add External JARs...**
4. Localizar a pasta da Workspace e **selecionar** o arquivo do conector que você já salvou dentro desta pasta.
5. Clicar em **OK**.

Indicação do conector no projeto

INFORMA. FORMA. TRANSFORMA.

ATENÇÃO:

Indicação de que a IDE localizou corretamente o conector

Indicação de que a IDE NÃO localizou corretamente o conector

11

INFORMA. FORMA. TRANSFORMA.

Criando o Banco de Dados

Objetivos:

- Criar um novo Banco de Dados no MySQL
- Criar uma tabela com as colunas referentes às informações que serão salvas via conexão com Java.

12

INFORMA. FORMA. TRANSFORMA.

Criando o Banco de Dados (*Cont.*)

1. Abrir o **MySQL Command Line Client** e se logar.
2. Criar e ativar o Banco de Dados com o nome **18_conexaobd**
3. Criar a tabela **Contato** com a seguinte estrutura:

Colunas	Domínios
id	int primary key auto_increment
nome	varchar(255)
email	varchar(100)
mensagem	varchar(255)

Estamos criando um BD com o mesmo nome do projeto Java apenas para facilitar o entendimento. Este BD poderia ter qualquer outro nome.

13

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *Contato*

Contato



- id : int
- nome : String
- email : String
- mensagem : String

+ get e set() : void
+ construtor() : Contato
+ toString() : String

Esta classe possui os atributos correspondentes à tabela **contato** do Banco de Dados.

Crie o pacote **model** e, dentro deste, a Classe **Contato**, conforme o diagrama ao lado.

14



INFORMA. FORMA. TRANSFORMA.

Criando a Classe *Contato*

Parte 1


```

package model;
public class Contato {



    private int id;
    private String nome;
    private String email;
    private String mensagem;
    
```

Atributos

Criar também os Métodos get e set, e construtor básico.



15



INFORMA. FORMA. TRANSFORMA.


Criando a Classe *Contato*

Parte 2

```

public Contato(int id, String nome, String email, String mensagem) {
    super();
    this.id = id;
    this.nome = nome;
    this.email = email;
    this.mensagem = mensagem;
}
    
```

Construtor para *visualização* de dados a partir de um campo *id auto_increment*



16

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *Contato*

Parte 3

```

public Contato(String nome, String email, String mensagem) {
    super();
    this.nome = nome;
    this.email = email;
    this.mensagem = mensagem;
}
    
```

Construtor para **gravação**
 de dados no banco de
 dados a partir de um
 campo
id auto_increment

17

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *Contato*

Parte 4 - Fim

```



@Override
public String toString(){
    final String ENTER = "\n";
    String retValue = "";

    retValue = "Mensagem Enviada Com Sucesso:" + ENTER +
        "Nome: " + nome + ENTER +
        "E-mail: " + email + ENTER +
        "Mensagem: " + mensagem;

    return retValue;
}
    
```

Método
toString()

18




 INFORMA. FORMA. TRANSFORMA.



Criando a Classe *Conexão*

Objetivos:
 Determinar elementos necessários para a conexão do banco de dados junto à Classe de drivers baixada da internet.

São eles:

- Endereço do Banco de Dados
- Classe Java que é o driver do framework baixado na web.
- Login do usuário no SGBD
- Senha do usuário no SGBD


19



 INFORMA. FORMA. TRANSFORMA.

Criando a Classe *Conexão*

Parte 1

`package util;`

`public class Conexao {`

```


private String url; // Local do Banco de Dados.
private String driver; // Classe Java do Framework que foi baixado na web.
private String login; // Login do usuário no SGBD.
private String senha; // Senha do usuário no SGBD.



```

Criar o pacote *util*, e dentro deste, a Classe *Conexao*

Todos os *imports* desta classe são de *java.sql*

Aqui estão os 4 atributos necessários para fazer a conexão.


20

 **FIRJAN**
 **INFORMA. FORMA. TRANSFORMA.**

Criando a Classe *Conexão*

Parte 2

```


public Conexao(String url, String driver, String login, String senha){
    try{



        this.url = url;
        this.driver = driver;
        this.login = login;
        this.senha = senha;
        Class.forName(driver);
        // registro da Classe de driver na conexão através de JDBC.

    } catch (ClassNotFoundException e){
        System.out.println(e.getMessage());
    }
}

```

É por este método que o *JDBC* é informado sobre qual é o driver.


21

 **FIRJAN**
 **INFORMA. FORMA. TRANSFORMA.**

Criando a Classe *Conexão*


Parte 3

```

public Connection obterConexao(){
    Connection con = null;
    try{
        con = DriverManager.getConnection(url, login, senha);
    } catch (SQLException e){
        System.out.println(e.getMessage());
    }
    return con;
}

```

Este método informa ao *DriverManager* a *url* do BD, o *login* e *senha* do usuário.


22

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *Conexão*

```

public String getDriver(){
    return driver;
}

public String getLogin() {
    return login;
}

public String getSenha() {
    return senha;
}
                
```

Parte 4 - Fim

Utilizam-se somente os métodos **get**, porque as entradas de dados são feitas pelo programador.
 (url , driver , login , senha)

23

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *ContatoDAO*



```

classDiagram
    class Contato {
        - id : int
        - nome : String
        - email : String
        - mensagem : String
        + get e set() : void
        + construtor() : Contato
        + toString() : String
    }
    class ContatoDAO {
        + inserir() : Contato
        + buscarTodos() : Contato
        + excluir() : Contato
        + atualizar() : Contato
    }
    ContatoDAO --> Contato
                
```

DAO = Data Access Object

Objetivo:
 Estabelecer o contato efetivo entre o Java e o SGBD, realizando as ações de seus métodos na tabela do Banco de Dados


24

INFORMA. FORMA. TRANSFORMA.

Inserir Mensagens...

O objetivo agora é possibilitar a inclusão de novas mensagens no Banco de Dados.


25




INFORMA. FORMA. TRANSFORMA.

Criando a Classe *ContatoDAO*

Criar o pacote **model.DAO** dentro do pacote **model**.
 Criar Classe **ContatoDAO** dentro do pacote **model.DAO**

Parte 1

```

public class ContatoDAO {

    public static Contato inserir(String nome, String email, String mensagem){
        Contato contato = null;
        try {
            // Criação do insert
            String sql = "insert into contato (nome, email, mensagem) values (?,?,?)";

            // Obter a conexão com o banco de dados
            Conexao conex = new Conexao("jdbc:mysql://localhost:3306/18_conexaobd",
                "com.mysql.jdbc.Driver","root","aluno");

            // Abrir a conexão
            Connection con = conex.obterConexao();
        
```

Trabalharemos por enquanto apenas com o método **inserir**

Continua ...


26

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *ContatoDAO*

```

// Preparar o comando para ser executado
PreparedStatement comando = con.prepareStatement(sql);

comando.setString(1,nome);
comando.setString(2,email);
comando.setString(3,mensagem);

// Comando executado
comando.executeUpdate();

} catch(Exception e){
    System.out.println(e.getMessage());
}
contato = new Contato (nome, email, mensagem);
return contato;
}
                
```

Continuando ...

Parte 2 - Fim

27

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *ControleDeContatos*

powered by astah®

Objetivo:
Implementar as ações do sistema.

OBS: Antes de criar esta Classe, copie a Classe ***Teclado*** do projeto ***09_EntradaDeDados*** e cole dentro do pacote ***util*** deste projeto atual.

28

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *ControleDeContatos*

Estudaremos a elaboração desta Classe em etapas, começando pela criação da Classe e implementação das constantes abaixo:

```
package controller;
public class ControleDeContatos {

    private static final int CADASTRAR_MENSAGEM = 1;
    private static final int VISUALIZAR_MENSAGENS = 2;
    private static final int APAGAR_MENSAGEM = 3;
    private static final int ATUALIZAR_MENSAGEM = 4;
    private static final int SAIR = 5;
}
```

Criar pacote *controller*
e, dentro deste, a
Classe
ControleDeContatos

29

INFORMA. FORMA. TRANSFORMA.

Criando a Classe *ControleDeContatos*

```
public static void main(String[] args) {
    ControleDeContatos cdc = new ControleDeContatos();
    int opcao = SAIR;
    do {
        cdc.mostrarMenu();
        opcao = Teclado.lerInt("Digite sua opção: ");



        switch (opcao) {
            case CADASTRAR_MENSAGEM:
                cdc.cadastrarMensagem();
                break;
            default:
                System.out.println("Opção inválida!");
        }
        Teclado.lerTexto("Pressione uma tecla para continuar...");
    } while (opcao != SAIR);
}
```

Método *main()*

Leitura de dados do
Teclado

Por enquanto, apenas o
case para a opção de
cadastrarMensagem.

30




 INFORMA. FORMA. TRANSFORMA.



Criando a Classe *ControleDeContatos*

Método *mostrarMenu()*

```

public void mostrarMenu() {
    System.out.println("=====");
    System.out.println("          Cadastro de Mensagens          ");
    System.out.println("=====");
    System.out.println("1 - Cadastrar Mensagem");
    System.out.println("2 - Mostrar Mensagens");
    System.out.println("3 - Apagar Mensagem");
    System.out.println("4 - Atualizar Mensagem");
    System.out.println("5 - Sair");
}
  
```


 31



 INFORMA. FORMA. TRANSFORMA.

Criando a Classe *ControleDeContatos*


Método *cadastrarMensagem()*



```

public void cadastrarMensagem() {
    System.out.println("=====");
    System.out.println("          Cadastro de Mensagens          ");
    System.out.println("=====");
    String nome = Teclado.lerTexto("Nome: ");
    String email = Teclado.lerTexto("E-mail: ");
    String mensagem = Teclado.lerTexto("Mensagem: ");

    Contato contato = ContatoDAO.inserir(nome, email, mensagem);


    System.out.println(contato);
}
  
```




 32



 INFORMA. FORMA. TRANSFORMA.

Consultar Mensagens...

O objetivo agora é possibilitar a consulta de mensagens já cadastradas no Banco de Dados.


33



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe *ContatoDAO*

```

public static Contato[] buscarTodos(){
    Contato[] contatos = null;

    try {
        // Criação do select
        String sql = "Select * from contato";


        // Obter a conexão com o banco de dados
        Conexao conex = new Conexao ("jdbc:mysql://localhost:3306/18_conexaobd",
            "com.mysql.jdbc.Driver","root","aluno");



        Connection con = conex.obterConexao();
    
```

Método
buscarTodos()

Parte 1

Continua . . .


34



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe *ContatoDAO*

Método
buscarTodos()

Parte 2


```



/* Executa a confirmação direta de acesso ao banco
 * pois não são necessárias informações para a
 * Query (caracter curinga)
 */
Statement comando = con.createStatement();

/* ResultSet - Classe java que monta em memória uma matriz
 * com a resposta das linhas do banco de dados
 */
ResultSet rs = comando.executeQuery(sql);

// vetor de objetos
contatos = new Contato[10];
  
```

Continua...


 35



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe *ContatoDAO*


```



/* Passagem de linha de dados do ResultSet para o vetor de objetos
 * (uma linha de dados da matriz do ResultSet é copiada para
 * um objeto no vetor contatos)
 */
int i = 0;
while (rs.next()) {
    contatos[i++] = new Contato(
        rs.getInt("id"),
        rs.getString("nome"),
        rs.getString("email"),
        rs.getString("mensagem"));
}
  
```

Método
buscarTodos()

Parte 3

Continua...


 36



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe *ContatoDAO*


```



    // É necessário encerrar o acesso ao banco para liberar a conexão
    rs.close();
    comando.close();
    con.close();
  } catch (Exception e){
    System.out.println(e.getMessage());
  }
  return contatos;
}

```

Método *buscarTodos()*

Parte 4 - Fim


 37



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe *ControleDeContatos*


Método *buscarMensagem()*



```

public void buscarMensagens(){
    Contato[] contatos = ContatoDAO.buscarTodos();

    for (int i=0; i<contatos.length; i++){
        if (contatos[i] !=null){
            System.out.println(
                contatos[i].getId() + "-----" +
                contatos[i].getNome() + "-----" +
                contatos[i].getEmail() + "-----" +
                contatos[i].getMensagem());
        }
    }
}

```


 38


INFORMA, FORMA, TRANSFORMA.



Continuando a Classe *ControleDeContatos*

Método *main()*

Inserir após o *break* do *case CADASTRAR_MENSAGEM*:


```
case VISUALIZAR_MENSAGENS:  
    cdc.buscarMensagens();  
    break;
```



39

INFORMA, FORMA, TRANSFORMA.

Excluir Mensagem...

O objetivo agora é possibilitar a exclusão de uma mensagem já cadastrada no Banco de Dados.

40



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe

ContatoDAO

Método *excluir()*
 Parte 1


Copiar e colar do método *inserir()*, realizando apenas as seguintes alterações:



- Declaração do método:


```
public static Contato excluir(int id){
```
- Query e comentário da *String sql*:


```
// Criação do delete
String sql = "delete from contato where id = ?";
```

Continua . . .


 42



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe


ContatoDAO



Método *excluir()*
 Parte 2 - Fim

- Na sequência de ações do objeto *comando*, deixar exatamente assim:


```
comando.setInt(1,id);
comando.executeUpdate();
```
- Apagar a linha:


```
contato = new Contato (nome, email, mensagem);
```


 42



INFORMA, FORMA, TRANSFORMA.

Continuando a Classe *ControleDeContatos*


Método *apagarMensagem()*



```

public void apagarMensagem() {
    System.out.println("=====");
    System.out.println("                Apagar Mensagem                ");
    System.out.println("=====");
    int id = Teclado.lerInt("Digite o número da mensagem a ser apagada:");
    ContatoDAO.excluir(id);
    System.out.println("Mensagem apagada com sucesso");
}

```


OBS: Fazer o case referente a este método junto aos outros cases dentro do método *main()* desta Classe.




43



INFORMA, FORMA, TRANSFORMA.

Atualizar Mensagem...

O objetivo agora é possibilitar a atualização (alteração) de uma mensagem já cadastrada no Banco de Dados.


44



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe

ContatoDAO

Método *atualizar()*
 Parte 1

Copiar e colar do método *inserir()*, realizando apenas as seguintes alterações:


- Declaração do método:



```
public static Contato atualizar(String mensagem, int id){
```

- Query e comentário da *String sql*:

```
// Criação do update
String sql = "update contato set mensagem = ? where id = ? ";
```

Continua ...


 45



 INFORMA. FORMA. TRANSFORMA.

Continuando a Classe

ContatoDAO

Método *atualizar()*
 Parte 2 - Fim



- Na sequência de ações do objeto *comando*, deixar exatamente assim:

```
comando.setString(1,mensagem);
comando.setInt(2,id);
```

- Apagar a linha:

```
contato = new Contato (nome, email, mensagem);
```


 46



INFORMA. FORMA. TRANSFORMA.

Continuando a Classe *ControleDeContatos*

Método *atualizarMensagem()*

```


public void atualizarMensagem() {
    System.out.println("=====");
    System.out.println("          Atualizar Mensagem          ");
    System.out.println("=====");
    int id = Teclado.lerInt("Digite o número id da mensagem a ser
editada:");
    String mensagem = Teclado.lerTexto("Mensagem: ");



    ContatoDAO.atualizar(mensagem, id);

    System.out.println("Mensagem atualizada com sucesso");
}

```

OBS: Fazer o case referente a este método junto aos outros cases dentro do método *main()* desta Classe.


47




INFORMA. FORMA. TRANSFORMA.

Projeto *InfoNote_13*

- Copiar o projeto *InfoNote_12* e colar renomeando-o.
- Fazer com que todas as tarefas do projeto acessem o Banco de Dados.

Esta tarefa está descrita num documento específico com o seguinte nome:

PD - TI - 12.1 - Instruções Projeto InfoNote_13.pdf




48

INFORMA, FORMA, TRANSFORMA.


Dúvidas?




49

INFORMA, FORMA, TRANSFORMA.


Bibliografia



Java Como Programar 8ª Edição
Paul Deitel e Harvey Deitel
Ed. Pearson



Java 7 Ensino Didático
Sérgio Furgeri
Ed. Érica



Fundamentos de Computação e Orientação a Objetos Usando Java
Francisco A. C. Pinheiro
Ed. LTC

50