## Comunicação e Acesso a Banco de Dados em Java<sup>TM</sup>

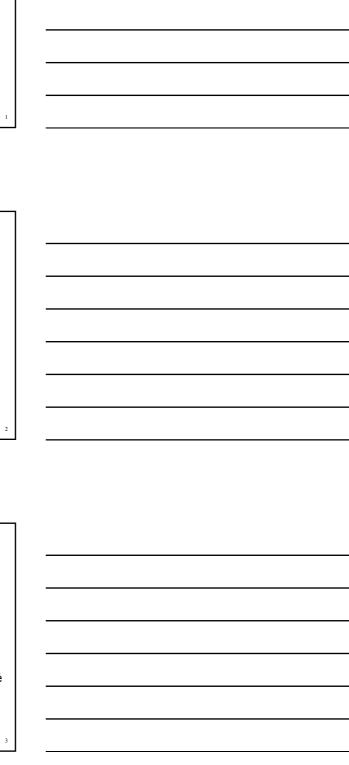
riccioni@univali.br

### Sumário

- Cliente-Servidor
  - Socket e ServerSocket
  - Interação no MVC: AplicacaoCliente e AplicacaoServidor
- · Multithreading
  - classe Thread e
  - interface Runnable
- Acesso a Banco de Dados
  - JDBC

### Cliente-Servidor

- · No pacote java.net, há duas classes que permitem implementar comunicação clienteservidor:
  - Socket e
  - ServerSocket.
- · O servidor deve aguardar um pedido de conexão, aceitá-lo e responder ao cliente de acordo com o protocolo implementado.
- · Como veremos, no servidor com multithread é possível atender a mais de um cliente simultaneamente.



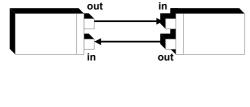
### A Classe Socket

- · Utilizada para representar a conexão entre o cliente e o servidor.
- Socket(String host, int porta)
  - cria um socket e o conecta ao servidor definido pelo host e a porta.
- · void close()
  - libera os recursos alocados.
- InputStream getInputStream()
  - obtém o stream de entrada associado ao socket.
- OutputStream getOutputStream()
  - obtém o stream de saída associado ao socket.

Programação em Java: Teoria e Prática Prof. Riccioni

### Cliente-Servidor

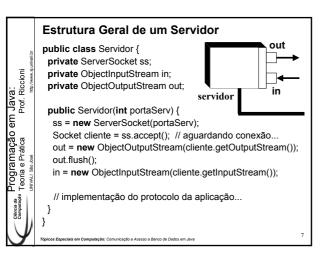
- Em Java, os canais de comunicação são unidirecionais (one-way).
- · Assim, tanto o cliente quanto o servidor precisam de dois streams:

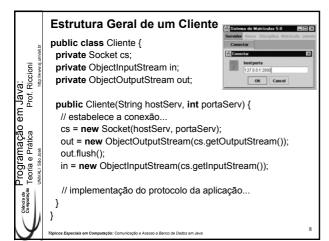


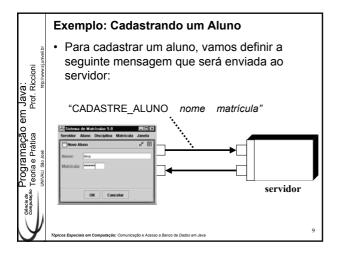
### A Classe ServerSocket

- · Para aceitar conexões, o servidor deve utiliza um ServerSocket.
- ServerSocket(int porta)
  - cria um socket nesta porta da máquina local.
- Socket accept()
  - aguarda até que uma conexão seja aceita. (bloqueante) Em seguida, retorna o Socket que representa a conexão com determinado cliente.
- void close()
  - libera os recursos alocados.

5		
-		
ır		
6		







```
Implementação do Cliente
public class Cliente {

//...

void cadastreAluno(String nome, String matricula) {

out.writeBytes("CADASTRE_ALUNO" +

nome + " " +

matricula + "\n");

out.flush();

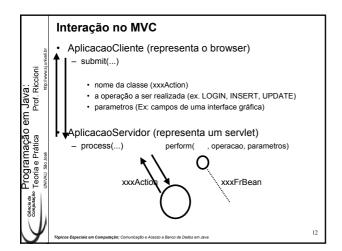
}

Exemplo: "CADASTRE_ALUNO Ana 9912345 \n"

Topicos Expecisis em Computação: Comunicação e Acesso à Basco de Dados em Java

10
```

```
Implementação do Servidor
         public class Servidor {
           public Servidor(int portaServ) {
Programação em Java:
Teoria e Prática Prof. Riccioni
              //...
              String mensagem = in.readLine();
                                                                  "Ana"
              // decompõe a mensagem.
              StringTokenizer st;
              st = new StringTokenizer(mensagem);
              String comando = st.nextToken();
              if(comando.equals("CADASTRE_ALUNO")) {
   Aluno novoAluno = new Aluno(st.nextToken(),
                                                      st.nextToken());
                // ...insere em cadAlunos...
                                                            ١
                                                            "9912345"
```



```
AplicacaoCliente
        public Mensagem submit(String nomeClasse,
                                    String operacao,
ramação em Java:
                             Hashtable parametros) throws
           Exception {
           Mensagem mensagem = new Mensagem();
           mensagem.setComando(nomeClasse);
           mensagem.setOperacao(operacao);
           mensagem.setObjeto(parametros);
           out.reset();
           out.writeObject(mensagem);
           out.flush();
           // aguarda a resposta do servidor
           return (Mensagem)in.readObject();
          os Especiais em Computação: Comunicação e Acesso a Banco de Dados em Java
```

```
AplicacaoServidor
public void process(String nomeClasse, String operacao,
                   Hashtable parametros) throws Exception {
 Action action = (Action)Class.forName(nomeClasse).newInstance();
 String nomeFrBean = ... xxxFrBean
 FrBean frBean = (FrBean)
  Class.forName(nomeFrBean).newInstance();
 PropertyUtils.copyProperties(frBean, parametros);
 String resultado = action.perform(frBean, operacao, parametros);
 // retorna a resposta
 Mensagem resposta = new Mensagem();
 resposta.setComando(resultado); // "OK" ou "...ERRO..."
 resposta.setOperacao(operacao);
 resposta.setObjeto(frBean); // Ex: linhas de um result set, etc.
 out.reset();
 out.writeObject(resposta);
 out.flush();
```

```
Classe Action

public abstract class Action {
    protected Class estaClasse = this.getClass();
    protected Class tiposParametros[];

public Action() {
        tiposParametros = new Class[] {
            Class.forName("FrBean"),

            Class.forName("java.util.Hashtable") };

}

public String perform(...) {...}

}
```

```
ramação em Java:
```

```
Nome do Método Resolvido Dinamicamente
public String perform(FrBean frBean, String operacao, Hashtable
parametros) try {
   String nomeMetodo = operacao.toLowerCase();
   StringTokenizer st = new StringTokenizer(nomeMetodo, "_");
   nomeMetodo = st.nextToken();
                                      // INSERT → insert
   while(st.hasMoreTokens()) {
                                      // SELECT_COUNT →
   selectCount
    String s = st.nextToken();
    nomeMetodo += ("" + s.charAt(0)).toUpperCase() +
   s.substring(1);
   Method metodo = estaClasse.getDeclaredMethod(nomeMetodo,
   tiposParametros);
   metodo.invoke(this, new Object[] { frBean, parametros } );
   return "OK";
  } catch(Exception ex) {
   return ex.toString();
```

```
Método Executado na Subclasse
         public class AlunoAction extends Action {
Programação em Java:
Tantia e Prática Prof. Riccioni
           public void insert(FrBean frBean, Hashtable
            parametros) throws Exception
             Aluno novo = new Aluno();
             PropertyUtils.copyProperties(novo, frBean);
             AlunoSQL.insert(novo,
            AplicacaoServidor.getConnection());
```

## Multithreading

- · Há várias formas de implementação para atender a vários clientes simultaneamente.
- Mas o uso de threads torna a implementação bastante simples.
- · Na API do Java, no próprio pacote java.lang, está a classe Thread, que implementa a seguinte interface:

interface Runnable { void run(); ..... Ponto de entrada de uma thread.

```
A Classe Thread (java.lang)

public class Thread implements Runnable {

//...

void run() {

// deve ser redefinido para

// implementar o comportamento desejado.

}

//...

//...

// ...

// ...

// ...

// ...

// ...
```

```
Servidor com Multithread
public class Servidor {
ServerSocket ss;

public Servidor()
{
ss = new ServerSocket(porta);

while(true) {
Socket conexao = ss.accept(); // aguarda conexão...

// Conectado!
ServidorThread st = new ServidorThread(conexao);
st.start();
}
}
}
```

```
A Classe ServidorThread

class ServidorThread extends Thread {
    private ObjectInputStream in;
    private ObjectOutputStream out;

public ServidorThread(Socket cs) {
    out = new ObjectOutputStream(cs.getOutputStream());
    out.flush();
    in = new ObjectInputStream(cs.getInputStream());
}

public void run() {
    // ...implementação do protocolo da aplicação....
}

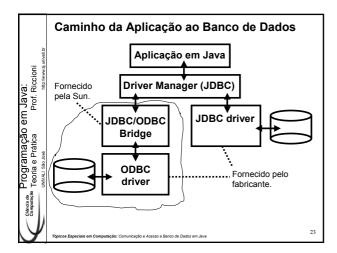
}
```

## ramação em Java: a e Prática Prof. Riccioni Programação Teoria e Prática

### Acesso a Banco de Dados com JDBC

- JDBC Java DataBase Conectivity API (pacote java.sql)
- Oferece uma interface em linguagem Java para acesso a banco de dados.
- JDBC 1 (1996)
- JDBC 2 (1998) oferece scrollable cursors,...
- JDBC 3 (2001) incluído na J2SE API v1.4: avanços em connection pools, ...
- Alguns gerenciadores de banco de dados incluem uma máquina virtual para executar stored procedures em Java (SQLJ).

22



### JDBC (Conceitos Básicos)

Programação em Java: Teoria e Prática Prof. Riccioni

URL(Uniform Resource Locator) o localizador uniforme de recursos de um banco de dados é semelhante ao endereco de algo na Internet, especifica o tipo de protocolo utilizado na ransferência, seguido de dois pontos e barra dupla, para depois indicar o nome do computador, o diretório e o nome do documento procurado:

jdbc:nome do protocolo:informações adicionais Exemplos:

- jdbc:odbc://sj.univali.br:3228/MEUBANCO;PWD=riccioni
  - Acesso ao banco de dados MEUBANCO, na porta 3228 de sj.univali.br, usando o atributo ODBC PWD configurado como riccioni

jdbc:odbc:SISMAT

· acesso a uma fonte de dados ODBC cujo nome é SISMAT.

jdbc:oracle:thin:@150.162.50.201:1521:orcl

acesso ao banco de dados da oracle na porta 1521 da máquina 150.162.50.201 cujo identificador SID = orcl.

iais em Computação: Comunicação e Acesso a Banco de Dados em Java

## ramação em Java: a e Prática Prof. Riccioni

### JDBC (Conceitos Básicos)

- DriverManager é uma classe do pacote java.sql e serve para:
  - Fazer a conexão com o banco de dados;
  - Gerenciar o conjunto de drivers JDBC correspondente;
  - Controlar login;
  - Controlar as mensagens entre o banco de dados e o driver.
- · Antes que um gerenciador de drivers possa ativar um driver, este deve ser registrado.
  - Para registrar um driver deve-se utilizar o método forName da classe Class (nome\_classe). Exemplo:
    - Class.forName( ("sun.jdbc.odbc.JdbcObdcDriver")
- Após registrar e carregar o driver, voce abre uma conexão de banco de dados.

Programação em Java:

### JDBC (Conceitos Básicos)

- Obtendo uma Conexão
  - As classes envolvidas são:
    - · Java.sql.Connection;
    - · Java.sql.DriverManager;
  - A classe Connection serve de base para várias outras classes. Vários métodos são implementados no contexto de um objeto da classe Connection
  - Para criar uma conexão com o banco de dados, deve-se utilizar o método DriverManager.getConnection(jdbc:url, login, senha)

### Estabelecendo uma Conexão

String url = "jdbc:odbc:SISMAT"; String user = "anonymous"; String password="guest"; Connection con;

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

con = DriverManager.getConnection(url, user, password);

} catch (ClassNotFoundException excecao1) { // Impossível carregar o bridge JDBC/ODBC. } catch (SQLException excecao2) { // Impossível estabelecer a conexão.

### Executando uma Instrução SQL

- As classes envolvidas nesta etapa são:
  - java.sgl.Statement;
  - java.sql.ResultSet;
  - java.sql.ResultSetMetaData;
  - Criação do Objeto Statement( usado para executar um comando SQL). Exemplo:
    - Statement stmt = con.CreateStatement();
      - O objeto Connection que você obteve na chamada a DriverManager.getConnection são usados para criar os objetos de instrução(Statement).
  - Em seguida, você insere a instrução que deseja executar em uma string. Exemplo:
- String comando = "Select \* from Alunos where nome = 'cdf' ";
- Então, você chama o método executeQuery da classe Statement:
- stmt.executeQuery(comando);
- O método Execute da interface Statement deve ser usado como executeQuery para consultas e executeUpdate para atualizações em um banco de dados.

os Especiais em Computação: Comunicação e Acesso a Banco de Dados em Java

Programação em Java: Teoria e Prática Prof. Riccioni

### **Executando Comandos SQL**

- · Para INSERT, UPDATE, DELETE, CREATE TABLE,...
- 1. Utilize a conexão aberta para criar um Statement: Statement stmt = con.createStatement();
- 2. Crie uma String contendo o comando SQL:

String comando = "UPDATE Alunos" +

"SET nota = 10.0" +

"WHERE nome = 'cdf' ":

- 3. Finalmente, chame o método executeUpdate: stmt.executeUpdate(comando);
  - o método retorna a quantidade de linhas da tabela que foram afetadas pelo comando executado.

## Commit e Rollback

- · Por default, a conexão automaticamente faz o commit. Isto pode ser alterado: con.setAutoCommit(false);
- Esta condição também pode ser verificada: boolean condicao = con.getAutoCommit();
- · Ao final, precisamos confirmar ou cancelar as operações realizadas:

con.commit();

con.rollback();

_					

### **Executando Comandos SQL**

- · Para SELECT.
- 1. Crie um Statement, como feito anteriormente.
- 2. Crie uma String contendo o comando SQL: String comando = "SELECT \* FROM Disciplinas";

3. Chame o método executeQuery:

ResultSet rs = stmt.executeQuery(comando);

4. Utilize o result set para obter o resultado da busca: while(rs.next()) {

// obtém os dados de uma linha do result set.

}

### ResultSet

Para ler o resultado da busca, precisamos obter os valores de cada coluna da linha atual:

```
while(rs.next()) {
    int codigo = rs.getInt("Codigo");
    String nome = rs.getString("Nome");
    int codigo = rs.getInt(1);
    String nome = rs.getString(2);
}
```

### **PreparedStatement**

· Podemos preparar um comando e utilizá-lo várias vezes, apenas definindo parâmetros: SELECT nome

SELECT HOME	
FROM Alunos	
WHERE matricula = matrícula digitada	
String busca = "SELECT nome " +	
"FROM Alunos" +	
"WHERE matricula = ?";	
PreparedStatement stmt = con.prepareStatement(busca);	
tmt.setString(1, jtfMatricula.getText());	
ResultSet rs = stmt.executeQuery();	
·	
ticos Especiais em Computação: Comunicação e Acesso a Banco de Dados em Java	- 3

# Computado Programação em Java: Computado Teoria e Prática Prof. Riccioni

Programação em Java: متمت Prof. Riccioni

### Conjunto de Resultados Roláveis e Atualizáveis

- O método next da classe ResultSet faz a interação sobre as linhas de um conjunto de resultados. Para que o usuário se mova para frente e para trás o JDBC 2, oferece os métodos: previous, isFirst, isLast,....
- No JDBC 1 para implentar esta facilidade tinhamos que trabalhar com os resultados em cache.
- Para obter conjunto de resultados roláveis, a partir de suas consultas, você deve obter um objeto Statement diferente com o método;
  - Statement stmt = con.createStatement(tipo, concorrencia);
- Para Instrução preparada use:
  - PreparedStatement = con.prepareStatement(comando,tipo,concorrencia);

Tópicos Especiais em Computação: Comunicação e Acesso a Banco de Dados em Java

3

## Conjunto de Resultados Roláveis e Atualizáveis

· Valores possíveis de tipo e concorrência:

### **TIPO**

TIPE_FORWARD_ONET	O conjunto de Resultados não e folavei
TYPE_SCROLL_INSENSITIVE	O conjunto de Resultados é rolável, mas não é sensível as alteração no Banco de Dados.
TYPE_SCROLL_SENSITIVE	O conjunto de Resultados é rolável, e é sensível as alteração no Banco de Dados.

## CONCORRÊNCIA

	usado para atualizar o Banco de Dados.
CONCUR_UPDATABLE	O conjunto de Resultados pode ser usado para atualizar o Banco de Dados.

Tópicos Especiais em Computação: Comunicação e Acesso a Banco de Dados em Java

3

### Conjunto de Resultados Roláveis e Atualizáveis

 Por exemplo, se você quer simplesmente rolar por um conjunto de resultados, mas não quer editar seu dados, então use:

### Statement stmt =

con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_READ\_ONLY);

•A partir de então todos os conjuntos de resultado que são retornados pela chamada do método:

- ResultSet rs = stmt.executeQuery(consulta)

Agora são roláveis. Um conjunto de resultados rolável tem um *cursor* que indica a posição corrente do registro na tabela.

Tópicos Especiais em Computação: Comunicação e Acesso a Banco de Dados em Java

-	
-	

## ramação em Java:

ramação em Java:

Teoria e Prática

### **MetaDados**

- Permite obter a estrutura de um Banco de Dados e de suas Tabelas:
- Por exemplo você pode obter uma lista das tabelas de banco de
- dados específico ou os nomes e tipos de coluna de uma tabela; Em SQL, os dados que descrevem o banco de dados ou uma de suas partes são chamados de metadados (para distingui-los dos dados reais que são armazenados no banco de dados.
- Tem dois tipo de metadados:
  - 1. Sobre o banco de dados usa-se o objeto DatabaseMetaData
    - Exemplo: DatabaseMetaData md = com.getMetadata(); Existem mais de 100 métodos desta classe, veremos anenas um
  - deles, o qual permite listar todas as tabelas de um banco de dados. Exemplo que retorna um conjunto de informações sobre todas as tabelas
    - ResultSet rs = md.getTables(null, null, null, null,new String[] {"TABLE"})
  - Cada linha do conjunto de resultados contém informações sobre a tabela, por exemplo: rs.getString(3) ou rs.getString("TABLE NAME") é o nome da tabela.

"ópicos Especiais em Computação: Comunicação e Acesso a Banco de Dados em Java

### MetaDados

A linha do conjunto de resultados tem 5 atributos:

1	Table_Cat	Catálogo da Tabela
2	Table_Schem	Esquema da Tabela
3	Table_Name	Nome da Tabela
4	Table_Type	Tipo da Tabela
5	Remarks	Comentário sobre a Tabela

2. O segundo tipo é sobre o conjunto de resultado de uma consulta, pode se perguntar sobre o número de colunas e nome, tipo e largura de campo de cada coluna.

## ResultSetMetaData

· Permite obter a estrutura de um result set:

String sql = "SELECT \* FROM Alunos"; ResultSet rs = stmt.executeQuery(sql); ResultSetMetaData <u>rsmd</u> = rs.getMetaData(); for(int i=1; i <= rsmd.getColumnCount(); i++) {</pre> String nomeColuna = rsmd.getColumnLabel(i); int larguraColuna = rsmd.getColumnDisplaySize(i); JLabel jlColuna = new JLabel(nomeColuna); JTextField jtfColuna = new JTextField(larguraColuna); //... podemos construir um frame genericamente!

## Computed Programação em Java: Computed Teoria e Prática Prof. Riccioni INVAL 180 Jose INVAL 180 Jose

### Fechando uma Conexão

 Uma conexão que não será mais utilizada deve ser fechada para liberar os recursos utilizados pelo JDBC:

```
try {
    con.close();
} catch(SQLException excecao) {
    excecao.printStackTrace();
}
```

 Para verificar se uma conexão ainda está aberta ou fechada:

```
boolean fechada = con.isClosed();
```

ipicos Especiais em Computação: Comunicação e Acesso a Banco de Dados em Javi

40

Programação em Java: Teoria e Prática Prof. Riccioni INIVALI 380-2006 INEPINAMA SANDAILE

## Recuperando Chaves (Auto) – JDBC 3.0

Statement stmt = con.createStatement();

ResultSet rs = stmt.getGeneratedKeys();

if(rs.next()) { // Recupera a chave gerada
 automaticamente.
 int codigo = rs.getInt(1); // código da disciplina
}

Tópicos Especiais em Computação: Comunicação e Acesso a Banco de Dados em Jav

4