JDBC - Java DataBase Connectvity

Prof. Msc. Christien Lana Rachid

API Java para a execução de comandos SQL

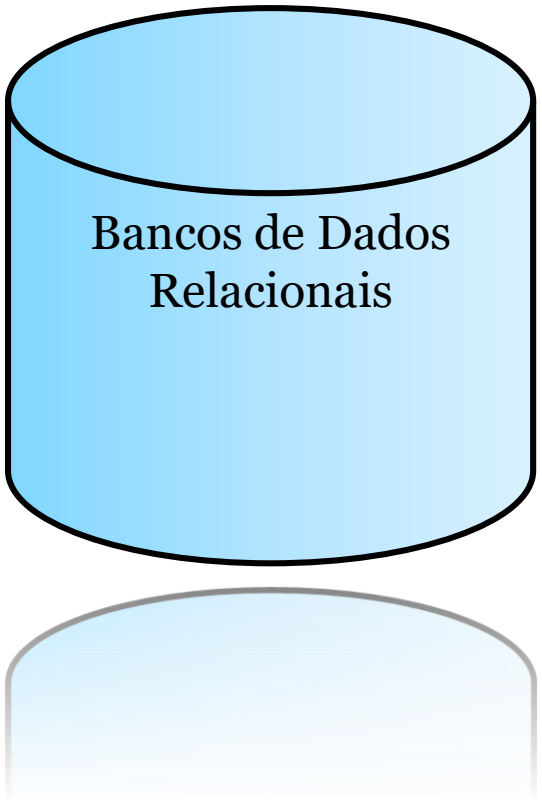
JDBC (Java DataBase Connectvity)

Consiste de um conjunto de classes e interfaces escritas em Java

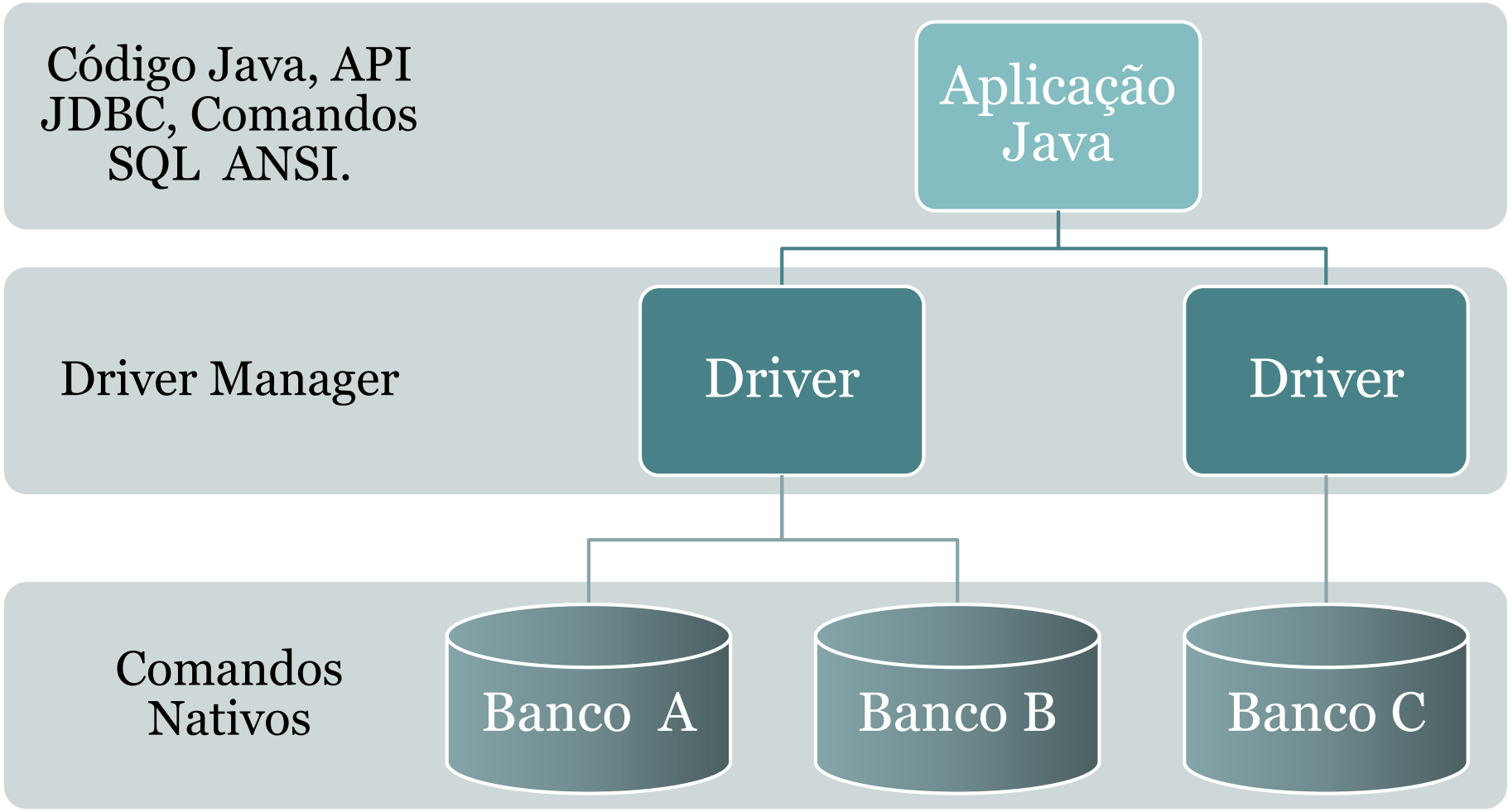
- Pacote java.sql e javax.sql (avançado)

Propósito:

- Enviar comandos SQL para qualquer banco relacional existente, independentemente da arquitetura e do banco de dados



Arquitetura JDBC

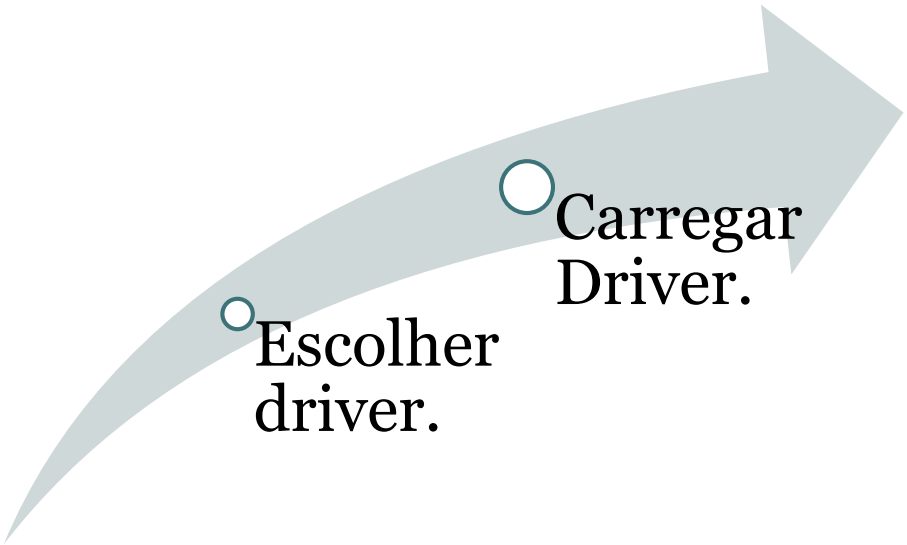


Passos do uso do JDBC

1. Carregar driver JDBC
2. Abrir a conexão ao banco de dados
3. Enviar comandos SQL
4. Processar os resultados

Passos do uso do JDBC

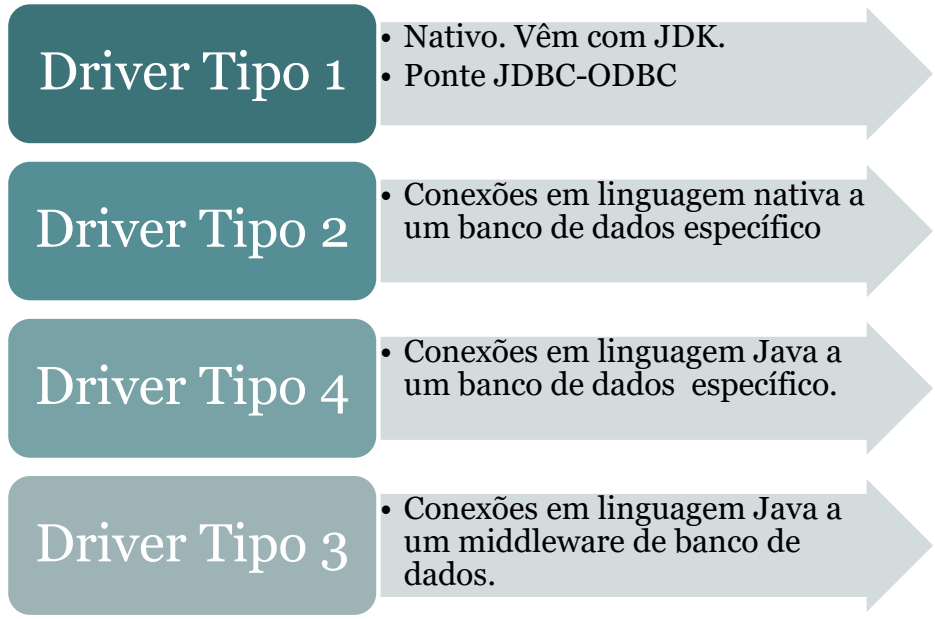
1. Carga do driver JDBC



Passos do uso do JDBC

1. Carga do driver JDBC

- Carregar Driver.
- *Escolher driver.*



Passos do uso do JDBC

1. Carga do driver JDBC

- Escolher driver.
- *Carregar Driver.*

Exemplo de carga do driver do tipo 01

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); ou
new sun.jdbc.odbc.JdbcOdbcDriver();
```

Passos do uso do JDBC

2. Abertura da Conexão ao Banco de Dados

- Classe DriverManager
  - Possui método para estabelecer conexão para a URL que contém o banco de dados. URL é o endereço de redes banco de dados.
- Exemplo:

```
String URL = "jdbc:odbc:MEUBANCO";
String usuario = "admin";
String senha = "";
Connection conexao = DriverManager.getConnection(url, usuario, senha).
```

Passos do uso do JDBC

3. Envio de Comandos SQL

- Classe Statement
  - Possui métodos para processar registros (tuplas) das tabelas.
- Exemplo:

```
Statement stmt = conexao.createStatement();
stmt.executeQuery("SELECT * FROM CLIENTE");
```

Passos do uso do JDBC

4. Processamento de Resultados

- Classe ResultSet
  - Possui métodos para enviar comandos SQL ao banco de dados.
- Exemplo:

```
ResultSet rs = stmt.executeQuery(
                        "SELECT * FROM CLIENTE");
  while (rs.next()) {
        int id = rs.getInt("ID");
        String nome = rs.getString("NOME");
        System.out.printf("%02d %s", id, nome);
  }
```

Código JDBC Mínimo

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url =“jdbc:odbc:MeuBanco”;
Connection conexao = DriverManager.getConnection(url, “admin”,””);
Statement stmt = conexao.createStatement();
ResultSet rs = stmt.executeQuery(“SELECT * FROM CLIENTE”);
while (rs.next()) {
        int id = rs.getInt(“ID");
        String nome = rs.getString(“NOME");
        System.out.printf(“%02d %s”, id, nome);
}
rs.close();
stmt.close();
conexao.close();
```

Código JDBC - Erros

Códigos JDBC/BD são altamente sensíveis a problemas de:

- Rede.
- Drivers.
- Sintaxe SQL.
- Esquemas de banco de dados.

Problemas são reportados em dois níveis:

- Exceções SQLException
- Avisos (Warnings)

Endereçamento de Exceções

```
try {
   // Codigo que manipule banco de dados deve usar  um
   //     tratamento de erros similar ao abaixo para tratar exceções SQL!
} catch(SQLException ex) {
   System.out.println("\n--- Exceção SQLException capturada  ---\n");
   while (ex != null) {
                  System.out.println("Mensagem: " + ex.getMessage ());
                                 System.out.println("Estado SQL: " +
   ex.getSQLState ());
         System.out.println("Código de erro " + ex.getErrorCode ());
         ex = ex.getNextException();
                  System.out.println("");
         }
}
```

JDBC Avançado

1. Variação do Driver JDBC
2. Uso do Pool de Conexões
3. Variação do Tipo de Comando ao Banco de Dados
4. Outros Comandos de manipulação do banco (DML)
5. Cursores

JDBC Avançado

1. Variação do Driver JDBC

Tipo	Escrito em Java	Maior Desempenho	Portabilidade de Banco de Dados	Portabilida de de Sistema Operacional
1	Não	Não	Sim	Não
2	Não	Sim	Não	Não
3	Sim	Não	Sim	Sim
4	Sim	Sim	Não	Sim

JDBC Avançado

1. Variação do Driver JDBC

Download do driver do fabricante

- Arquivo .JAR

Carga do nome do driver

- Dependente do fabricante. Manual/help deve ser consultado.

Configuração da URL da conexão.

- Dependente do fabricante. Manual/help deve ser consultado.

Código JDBC p/ Oracle

```
Class.forName("oracle.jdbc.driver.OracleDriver");
String servidor = "127.0.0.1"; String porto="1521";
String sid="MEUBANCO";
String url ="jdbc:oracle:thin:@" + servidor+ ":" +porto+ ":" + sid ;
Connection conexao = DriverManager.getConnection(url, "scott","tiger");
Statement stmt = conexao.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM CLIENTE");
while (rs.next()) {
        String nome = rs.getString("NOME");
        System.out.printf("%s", id, nome);
}
rs.close(); stmt.close(); conexao.close();
```

Código JDBC p/ mySQL

```
Class.forName("org.gjt.mm.mysql.Driver"); // MySQL MM JDBC driver
String servidor = "127.0.0.1";
String sid="MEUBANCO";
String url = "jdbc:mysql://" + serverName + "/" + mydatabase;
Connection conexao = DriverManager.getConnection(url,
   "admin","admin");
Statement stmt = conexao.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM CLIENTE");
while (rs.next()) {
            String nome = rs.getString("NOME");
            System.out.printf("%s", id, nome);
}
rs.close(); stmt.close(); conexao.close();
```

JDBC Avançado

1. Variação do Driver JDBC

Oracle mantém lista de drivers certificados Java SE e Java EE.

- http://developers.sun.com/product/jdbc/drivers

Mais de 220 drivers e diversos são open-source

JDBC Avançado

2. Uso de Pool de Conexões

JDBC possui o recurso de otimização de criação e gerência de conexões (Pool de conexões).

DataSource é objeto que gerencia conexões em um Pool de Conexões.

- Parâmetros e uso do DataSource variam conforme fornecedor do Driver.

Um pool de conexões opera como um pool de atendentes em um call center.

- Cada *atendente* endereça um cliente.

Código JDBC com DataSource

```
DataSource ds = (DataSource) new org.apache.derby.jdbc.ClientDataSource()
ds.setPort(1527); ds.setHost("localhost");
ds.setUser("APP") ; ds.setPassword("APP");
Connection con = ds.getConnection();
Statement stmt = conexao.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM CLIENTE");
while (rs.next()) {
        String nome = rs.getString("NOME");
        System.out.printf("%s", id, nome);
}
rs.close(); stmt.close(); conexao.close();
// Close na conexão devolve a conexão ao pool.

...
```

JDBC Avançado

3. Variação do Tipo de Comando ao Banco de Dados

Statement	• Comandos simples.
PreparedStatement	• Comandos pré-compilados. • Comandos com parâmetros.
Callable Statement	• Invocação a funções e procedimentos armazenados.

Código JDBC com PreparedStatement

```
DataSource ds = (DataSource) new org.apache.derby.jdbc.ClientDataSource()
ds.setPort(1527); ds.setHost("localhost");
ds.setUser("APP") ; ds.setPassword("APP");
Connection con = ds.getConnection();
PreparedStatement pstmt = conexao.prepareStatement(
                          "SELECT * FROM CLIENTE WHERE ID = ?1);
stmt.setInt(1, "34675");
ResultSet rs = pstmt.executeQuery();
while (rs.next()) {
        String nome = rs.getString("NOME");
}
rs.close(); stmt.close(); conexao.close();

...
```

Código JDBC com Callable Statement

Stored Procedure Criado no Banco:

```
create procedure SHOW_SUPPLIERS as select SUPPLIERS.SUP_NAME, COFFEES.COF_NAME
    from SUPPLIERS, COFFEES where SUPPLIERS.SUP_ID = COFFEES.SUP_ID order by
    SUP_NAME
```

Código Java

```
....
Connection con = ds.getConnection();
CallableStatement cs = con.prepareCall("{call SHOW_SUPPLIERS}");
ResultSet rs = cs.executeQuery();
while (rs.next()) {
            String cafe= rs.getString("COF_NAME");
}
rs.close(); stmt.close(); con.close();


…
```

JDBC Avançado

4. Outros Comandos de manipulação do banco (DML)

Consultas	• executeQuery
Inserção, Atualização e Remoção	• executeUpdate
Transações	• setAutoCommit // true ou false • commit • Rollback
Comandos genéricos	• execute() • Ex: Create Table

JDBC Avançado

5. Cursores: Andarilhos sobre um conjunto de registro de dados

TYPE_FORWARD_ONLY	• Curso unidirecional. Move-se do começo para o final do conjunto de registros.
TYPE_SCROLL_INSENSITIVE	• Curso bidirecional. Permite movimentação para um ponto absoluto ou relativo à posição corrente.
TYPE_SCROLL_SENSITIVE	• Curso bidirecional. Permite movimentação para um ponto absoluto ou relativo à posição corrente.

JDBC Avançado

5. Cursores: Cada andarilho (cursor) pode percorrer a coleção em modo de leitura apenas ou leitura/gravação.

CONCUR_READ_ONLY	• Cursor com permissão de leitura apenas.
CONCUR_UPDATABLE	• Cursor com permissão de leitura/gravação apenas.

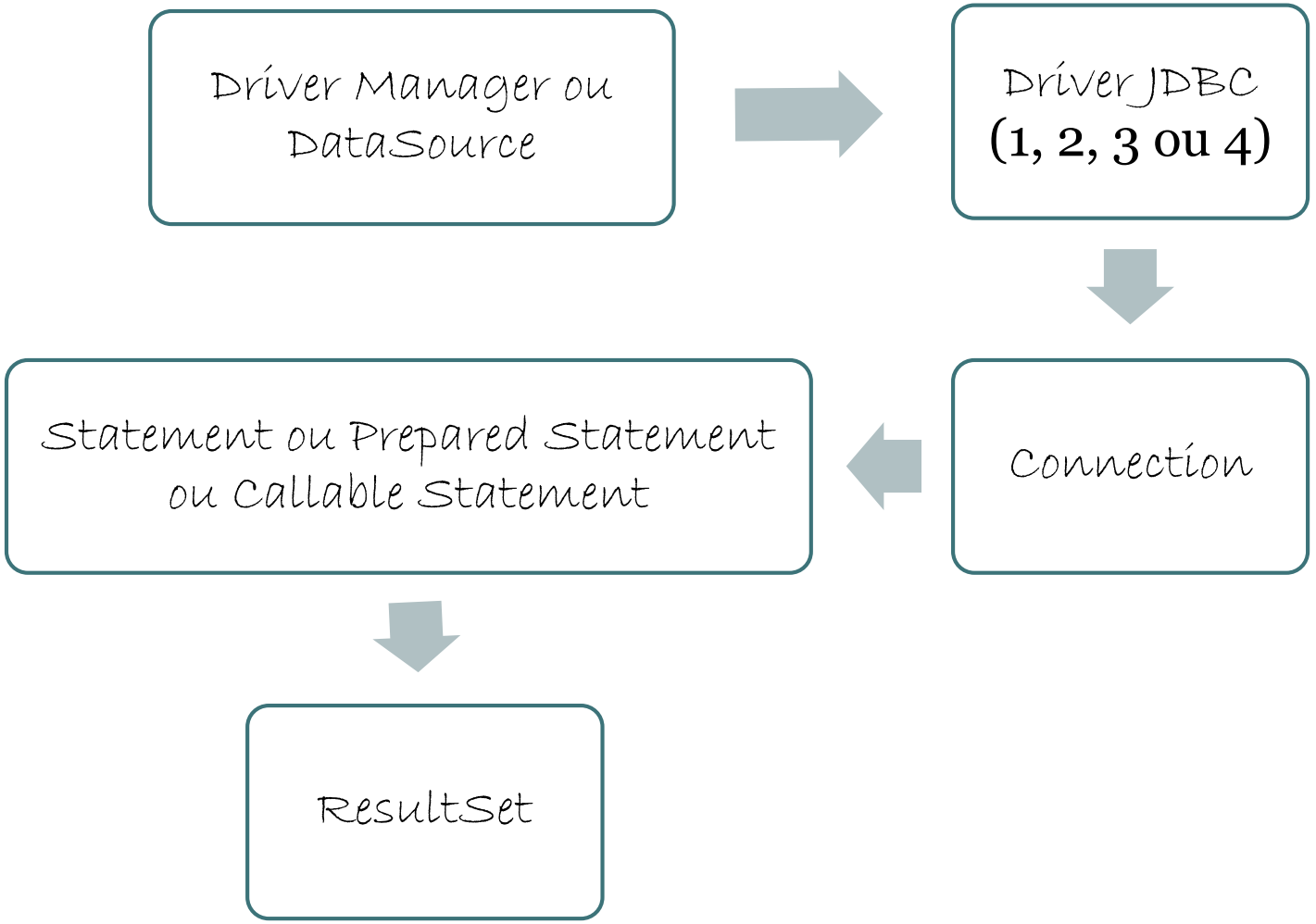
JDBC Avançado – Cursores

```
Statement stmt  =
   con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                ResultSet.CONCUR_READ_ONLY);
ResultSet srs = stmt.executeQuery( "SELECT COF_NAME, PRICE FROM
   COFFEES");
while (srs.next()) {
    String name = srs.getString("COF_NAME");
    System.out.println(name + " " + price);
}
// Saída seria: CAFÉ do SUL de MINAS e CAFÉ do Triangulo Mineiro
```

JDBC Avançado – Métodos Úteis de Cursores

next()	• Avança para a próxima posição.
previous()	• Avança para a posição anterior
first()	• Avança para a primeira posição.
last()	• Avança para a última posição.
beforeFirst()	• Avança para a posição anterior a primeira posição (BOF)
afterLast()	• Avança para a posição posterior à ultima posição (EOF)
relative(int rows)	• Move o cursor relativo à posição corrente.
absolute(int rows)	• Move o curso para a posição absoluta especificada no argumento.

JDBC - Resumo


Driver Manager ou DataSource
Driver JDBC (1, 2, 3 ou 4)
Connection
Statement ou Prepared Statement ou Callable Statement
ResultSet

JDBC – para saber mais

jGuru JDBC 2.0

- http://java.sun.com/developer/onlineTraining/Database/JDBC20Intro/index.html

JDBC Short Course

- http://java.sun.com/developer/onlineTraining/Database/JDBCShortCourse/index.html

Basic Tutorial

- http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html

Advanced Tutorial

- http://java.sun.com/developer/Books/JDBCTutorial/index.html

RowsetTutorial (Cursores)

- http://java.sun.com/developer/Books/JDBCTutorial/chapter5.html

Conclusões

- JDBC é API java para acesso a qualquer banco de dados relacional
- JDBC é parte obrigatória da especificação Java SE e Java EE
- JDBC possui drivers para diversos fabricantes de bancos de dados.
- Frameworks de persistência objeto relacional ainda usam JDBC, mas fornecem produtividade para desenvolvedores Java.