UNIVERSIDADE PAULISTA

Aplicações de Linguagem de Programação Orientada a Objetos

MÓDULO 6

Professor: Me. Pablo I. Gandulfo
Data: 01/08/2020
Versão: 1.1

Professor: Me. Pablo I. Gandulfo
Data: 01/08/2020
Versão: 1.1

Manipulação de dados com linguagem SQL

- Passos Envolvidos
- Comandos SQL:
  - Statement
  - PreparedStatement
  - CallableStatement
- Exemplo Completo de um CRUD - Classe Banco de Dados
- Metadados e DDL
- Exercícios

Passo 1: Registrar um *driver* JDBC

- O *driver* é utilizado para conectar ao Banco de Dados
- A API JDBC usa o primeiro *driver* encontrado que seja capaz de estabelecer com sucesso uma conexão à URL informada
- É possível carregar múltiplos *drivers* ao mesmo tempo
- Formas de registrar um *driver*:
  - Usando o *class loader*
  - Instanciando um *driver*
  - Usando a propriedade jdbc.drivers

Passo 1: Registrar um *driver* JDBC (cont.)

- Usando o *class loader*, a definição da classe é carregada
  - Sintaxe:
    - Class.forName(nomeDriver);
  - Exemplo:
    - Class.forName("oracle.jdbc.driver.OracleDriver");

- Se for necessário ter uma referência explícita do objeto do Driver, use a palavra chave **new**
  - Sintaxe:
    - Driver varRefDrv = new ConstrutorDoDriver();
  - Exemplo:
    - Driver drv = new oracle.jdbc.driver.OracleDriver();

Passo 2: Estabelecer uma conexão ao Banco de Dados

- O DriverManager chama getConnection(stringUrl), que chama Driver.connect(stringUrl)
- A URL é interpretada
- Quando um *driver* corresponder positivamente a URL, o DriverManager tenta estabelecer a conexão
- Se o *driver* não corresponder, um valor nulo é retornado e o próximo driver na coleção é verificado
- Se uma conexão não for estabelecida, uma SQLException é lançada

Passo 2: Estabelecer uma conexão ao Banco de Dados (cont.)

- O método DriverManager.getConnection:
  - getConnection(String url)
  - getConnection(String url, java.util.Properties info)
  - getConnection(String url, String user, String password)
- Sintaxe:
  - Connection varRefCon =
    DriverManager.getConnection(argumentos);
- Exemplo:
  - Connection conn =
    DriverManager.getConnection(
    "Jdbc:hsqldb:file:test", "sa", "");

Passo 3: Criar um *statement*

- Pode ser de três tipos:
  - Statement
  - PreparedStatement
  - CallableStatement

Passo 3: Criar um *statement*

- Objeto **Statement**
  - Sintaxe:
    - Statement varRefStmt = conRefVar.createStatement();
  - Exemplo:

```
Statement stmt;
try {
    stmt = con.createStatement();
} catch (SQLException e) {
    System.out.println (e.getMessage());
}
ResultSet rs = stmt.executeQuery("SELECT * FROM funcionario WHERE matricula = 2");
```

Passo 3: Criar um *statement*

- Objeto **PreparedStatement**
  - É um comando SQL pré-compilado que apresenta melhor desempenho que chamar repetidamente o mesmo comando SQL
  - Estende a interface Statement

  - Sintaxe:
    - PreparedStatement varRefPstmt =
      varRefCon.prepareStatement(String sql);

Passo 3: Criar um *statement*

- Objeto **PreparedStatement**
  - Exemplos:

```
PreparedStatement pstmt = con.prepareStatement("SELECT * FROM cafes WHERE nome_cafe = ?");
pstmt.setString(1, "Expresso");
ResultSet rs = pstmt.executeQuery();
```

```
PreparedStatement pstmt = con.prepareStatement("UPDATE cliente SET nome = ? WHERE id = ?");
pstmt.setString(1, "Carlos Brito");
pstmt.setInt(2, 219);
int result = pstmt.executeUpdate();
```

Passo 3: Criar um *statement*

- Objeto **CallableStatement**
  - Permite executar comandos que não são SQL no Banco de Dados
  - Estende a interface PreparedStatement

  - Sintaxe:
    - CallableStatement varRefCstmt =
      varRefCon.prepareCall(String sql);

Passo 3: Criar um *statement*

- Objeto **CallableStatement**
  - Exemplos:

```
String nomeCafe = "Expresso";
CallableStatement cstmt = varRefCon.prepareCall("{call total_vendas[?, ?]}");
cstmt.setString(1, nomeCafe);
cstmt.registerOutParameter(2, Types.REAL);
cstmt.execute();
float vendas = cstmt.getFloat(2);
```

```
CallableStatement cstmt = varRefCon.prepareCall("{call get_nome_cliente[?, ?]}");
cstmt.setInt(1, 219);
cstmt.registerOutParameter(2, Types.INTEGER);
cstmt.execute();
int valorRetorno = cstmt.getInt(2);
```

Passo 3: Criar um *statement*

- Interface **Statement**
  - Métodos:
    - ResultSet executeQuery(String sql)
      - Executa uma instrução SQL de pesquisa e retorna o objeto ResultSet gerado pela pesquisa
    - int executeUpdate(String sql)
      - Executa uma instrução SQL de atualização de dados (INSERT, UPDATE ou DELETE) ou de metadados (CREATE, DROP, etc.) e retorna o número de linhas afetadas
    - boolean execute(String sql)
      - Executa qualquer instrução SQL

Passo 4: Executar o comando SQL

- A instrução SQL é passada inalterada para a conexão com o Banco de Dados
- O resultado é uma tabela de dados acessível através de java.sql.ResultSet
- O *statement* fica acoplado com o ResultSet correspondente

Passo 5: Processar os resultados

- Para recuperar dados do objeto ResultSet, use seus métodos acessores

- Recupere esse dados usando um nome de coluna ou índice

- Um ResultSet:
  - Mantém um cursor posicionado na linha atual
  - É inicialmente posicionado antes da primeira linha

Passo 5: Processar os resultados

- Sintaxe:

```
while (varRefRs.next()) {
    System.out.println ("Registro: " + varRefRs.getMetodoXxx(coluna));
}
```

- Exemplo:

```
while (rs.next()) {
    System.out.println();
    System.out.println("Nome do Café         : " + rs.getString(1));
    System.out.println("ID do Fornecedor     : " + rs.getInt(2));
    System.out.println("Preço                : " + rs.getFloat(3));
    System.out.println("Total de Vendas      : " + rs.getInt(4));
}
```

Passo 6: Fechar objetos JDBC

- Fechar os ResultSets, Statements e Conexões

- Exemplo:

```
try {
    rs.close();
    stmt.close();
    con.close();
}
catch (SQLException ex) {
    ex.printStackTrace();
}
```

Exercício 1

- Desenvolva um programa (com interface gráfica AWT / Swing ou não) com as seguintes características:
  - Conecte num BD qualquer
  - Execute um comando SQL INSERT para incluir um registro na tabela
  - Execute um comando SQL SELECT para ler os dados do registro incluído
  - Imprima mensagens informando o usuário de cada passo