8.1 JDBC

A linguagem Java possui classes que permitem a conexão com um banco de dados. Essas classes fazem parte do pacote *JDBC* (Java Database Connectivity), uma *API* (Application Program Interface) que permite a comunicação com diversos bancos de dados padrão *SQL* (Oracle, MySQL, Firebird, SQL Server, Postgree dentre outros). Muitos desses bancos de dados utilizam um padrão de comunicação chamado *ODBC* (Open Database Connectivity) que pode ser usado para criar uma fonte de dados entre o banco e o *JDBC*.

O pacote *JDBC* fornece uma maneira bem simples de acessar tabelas em Java, utilizando comandos *SQL* (Structured Query Language). Ele é instalado em conjunto com o *JDK* e não é necessário fazer nenhuma instalação adicional para manipular banco de dados em Java.

A maioria dos fornecedores de banco de dados importantes fornece seus próprios drivers de banco de dados de *JDBC* e muitos fornecedores independentes também fornecem drivers *JDBC.*

Portanto, para efetuar a conexão de um banco de dados com Java, é necessário, primeiramente, baixar o driver (arquivo JAR) JDBC do banco de dados ao qual se deseja conectar, e adicionar as bibliotecas do projeto.

8.2 Consulta SQL com JDBC

De forma simples, para a execução de comandos SQL com JDBC, precisa-se da instancia de três classes Java. São elas:

Nome da classe	Função
Connection	Estabelece a conexão física (servidor, porta, usuário e senha) com o banco de dados.
Statement	Responsável por executar o comando SQL em uma conexão pré-estabelecida.
ResultSet	Armazena o resultado da execução de uma consulta (SELECT) SQL.

A classe *ResultSet* só é requerida quando precisa-se manipular os resultados de uma consulta com o comando SELECT. Para as demais manipulações: INSERT, UPDATE e DELETE, apenas as classes *Connection* e *Statement* são necessárias.

A classe Statement possui duas variantes: PreparedStatement e CallableStatement. A descrição completa dessas três classe segue abaixo:

Classe	Uso recomendado
Statement	Usada para propósito geral de execução de comandos SQL. Quando se tem uma string com o comando SQL final para execução, sem que seja necessário adicionar parâmetros ou validações.
PreparedStatement	Usada quando parâmetros são passados dinamicamente à instrução SQL.
CallableStatement	Usada para execução de Store Procedures e Functions. Também aceita parâmetros dinâmicos.

A seguir um exemplo que efetua a conexão JDBC com um banco de dados, executa uma instrução SELECT com um objeto da classe ***Statement***, e o processamento do resultado com um objeto da classe ***ResultSet***.

```
 1 package jdbc;
 2
 3 import java.sql.Connection;
 4 import java.sql.DriverManager;
 5 import java.sql.ResultSet;
 6 import java.sql.SQLException;
 7 import java.sql.Statement;
 8 import javax.swing.JOptionPane;
 9
10 public class ConsultaJDBC {
11    private final static String DRIVER = "oracle.jdbc.OracleDriver";
12    private final static String BANCO = "jdbc:oracle:thin:@localhost:1521:XE";
13    private final static String USUARIO = "treinamento";
14    private final static String SENHA = "treinamento";
15    private final static String SQL = "SELECT * FROM USUARIO";
16
17    public static void main(String args[]) {
18      try {
19         Class.forName(DRIVER);  //define o driver do banco
20         Connection conexao = DriverManager.getConnection(BANCO, USUARIO, SENHA);
//cria conexão
21         Statement stm = conexao.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY); //executa SQL
22         ResultSet rs = stm.executeQuery(SQL); //obtem dados da consulta SQL
23
24         // Impressão dos dados
25         System.out.println("ID     NOME");
26         while (rs.next()) {
27            String codigo = rs.getString("USU_ID");
28            String nome = rs.getString("USU_NOME");
29
30            System.out.println(codigo+"      "+nome);
31
32         }
33      } catch (ClassNotFoundException ex) {
           JOptionPane.showMessageDialog(null, "Erro na Classe de Conexao do Banco. \n"
+ ex.getMessage());
35      } catch (SQLException ex) {
36         JOptionPane.showMessageDialog(null, "Erro na Consulta SQL \n" +
ex.getMessage());
37      }
38    }
39 }
```

8.3 Incluindo registros com JDBC

Nesse exemplo, foi utilizado um objeto da classe ***PreparedStatement***, para a passagem dos parâmetros de inclusão de forma dinâmica (linhas 27, 28 e 29). Nas linhas 19 e 20 encontra-se o comando SQL com os coringas interrogação (?), referente a cada parâmetro que será tratado pelo objeto ***stm*** (***PreparedStatement***).

```
 1 package jdbc;
 2
 3 import java.sql.Connection;
 4 import java.sql.DriverManager;
 5 import java.sql.PreparedStatement;
 6 import java.sql.SQLException;
 7 import javax.swing.JOptionPane;
 8
 9 public class InclusaoJDBC {
10
11     private final static String DRIVER = "oracle.jdbc.OracleDriver";
12     private final static String BANCO = "jdbc:oracle:thin:@localhost:1521:XE";
13     private final static String USUARIO = "treinamento";
14     private final static String SENHA = "treinamento";
15
16     public static void main(String args[]) throws SQLException {
17         Connection conexao = null;
18         PreparedStatement stm = null;
19         String SQL = "INSERT INTO CIDADE (CID_ID, CID_NOME, CID_UF) "+
20                      "VALUES (?, ?, ?)";
22         try {
23             Class.forName(DRIVER);  //define o driver do banco
24             conexao = DriverManager.getConnection(BANCO, USUARIO, SENHA); //cria
conexão
25
26             stm = conexao.prepareStatement(SQL);
27             stm.setInt    (1, 0); // CAMPO CID_ID
28             stm.setString(2, "SÃO PAULO"); //CAMPO CID_NOME
29             stm.setString(3, "SP"); // CAMPO CID_UF
30
31             int r = stm.executeUpdate(); //obtem dados da consulta SQL
32
33             if (r == 1) {
34                 JOptionPane.showMessageDialog(null, "Cidade Cadastrada com
Sucesso!");
35             } else {
36                 JOptionPane.showMessageDialog(null, "Problemas na Inclusão de
Cidade");
37             }
38         } catch (ClassNotFoundException ex) {
39             JOptionPane.showMessageDialog(null, "Erro na Classe de Conexao do Banco.
\n" + ex.getMessage());
40         } catch (SQLException ex) {
41             JOptionPane.showMessageDialog(null, "Erro na Inclusão SQL. \n" +
ex.getMessage());
42         } finally {
43             stm.close();
44             conexao.close();
45         }
46     }
47 }
```

8.4 Alterando registros com JDBC

Nesse exemplo, foi utilizado um objeto da classe ***PreparedStatement***, para a passagem dos parâmetros de alteração de forma dinâmica (linhas 26, 27 e 28). Na linha 19 encontra-se o comando SQL com os coringas interrogação (?), referente a cada parâmetro que será tratado pelo objeto ***stm*** (***PreparedStatement***).

```
 1 package jdbc;
 2
 3 import java.sql.Connection;
 4 import java.sql.DriverManager;
 5 import java.sql.PreparedStatement;
 6 import java.sql.SQLException;
 7 import javax.swing.JOptionPane;
 8
 9 public class AlteracaoJDBC {
10
11     private final static String DRIVER = "oracle.jdbc.OracleDriver";
12     private final static String BANCO = "jdbc:oracle:thin:@localhost:1521:XE";
13     private final static String USUARIO = "treinamento";
14     private final static String SENHA = "treinamento";
15
16     public static void main(String args[]) throws SQLException {
17         Connection conexao = null;
18         PreparedStatement stm = null;
19         String SQL = "UPDATE CIDADE SET CID_NOME = ?, CID_UF = ? WHERE CID_ID = ?";
20
21         try {
22             Class.forName(DRIVER);  //define o driver do banco
23             conexao = DriverManager.getConnection(BANCO, USUARIO, SENHA); //cria
conexão
24
25             stm = conexao.prepareStatement(SQL);
26             stm.setString(1, "PORTO ALEGRE"); //CAMPO CID_NOME
27             stm.setString(2, "RS"); // CAMPO CID_UF
28             stm.setInt   (3, 1); // CAMPO CID_ID
29
30             int r = stm.executeUpdate(); //obtem dados da consulta SQL
32             if (r == 1) {
33                 JOptionPane.showMessageDialog(null, "Cidade Alterada com Sucesso!");
34             } else {
35                 JOptionPane.showMessageDialog(null, "Problemas na Alteração de
Cidade");
36             }
37         } catch (ClassNotFoundException ex) {
38             JOptionPane.showMessageDialog(null, "Erro na Classe de Conexao do Banco.
\n" + ex.getMessage());
39         } catch (SQLException ex) {
40             JOptionPane.showMessageDialog(null, "Erro na Alteração SQL. \n" +
ex.getMessage());
41         } finally {
42             stm.close();
43             conexao.close();
44         }
45     }
46 }
```

8.5 Excluindo registros com JDBC

Nesse exemplo, foi utilizado um objeto da classe ***PreparedStatement***, para a passagem do parâmetro de exclusão de forma dinâmica (linha 26). Na linha 19 encontra-se o comando SQL com o coringa interrogação (?), referente ao parâmetro que será tratado pelo objeto ***stm*** (***PreparedStatement***).

```
 1 package jdbc;
 2
 3 import java.sql.Connection;
 4 import java.sql.DriverManager;
 5 import java.sql.PreparedStatement;
 6 import java.sql.SQLException;
 7 import javax.swing.JOptionPane;
 8
 9 public class ExclusaoJDBC {
10
11     private final static String DRIVER = "oracle.jdbc.OracleDriver";
12     private final static String BANCO = "jdbc:oracle:thin:@localhost:1521:XE";
13     private final static String USUARIO = "treinamento";
14     private final static String SENHA = "treinamento";
15
16     public static void main(String args[]) throws SQLException {
17         Connection conexao = null;
18         PreparedStatement stm = null;
19         String SQL = "DELETE FROM CIDADE WHERE CID_ID = ?";
20
21         try {
22             Class.forName(DRIVER);  //define o driver do banco
23             conexao = DriverManager.getConnection(BANCO, USUARIO, SENHA); //cria
conexão
24
25             stm = conexao.prepareStatement(SQL);
26             stm.setInt(1, 1); // CAMPO CID_ID
27
28             int r = stm.executeUpdate(); //obtem dados da consulta SQL
29
30             if (r == 1) {
31                 JOptionPane.showMessageDialog(null, "Cidade Excluída com Sucesso!");
32             } else {
33                 JOptionPane.showMessageDialog(null, "Problemas na Exclusão de
Cidade");
34             }
35         } catch (ClassNotFoundException ex) {
36             JOptionPane.showMessageDialog(null, "Erro na Classe de Conexao do Banco.
\n" + ex.getMessage());
37         } catch (SQLException ex) {
38             JOptionPane.showMessageDialog(null, "Erro na Exclusão SQL. \n" +
ex.getMessage());
39         } finally {
40             stm.close();
41             conexao.close();
42         }
43     }
44 }
```