JDBC - Tutorial

JDBC é uma API que permite que aplicações Java interajam com bases de dados. Fornece métodos para conectar-se a uma base de dados, executar consultas SQL e manipular os resultados. Cada proprietário de uma base de dados deve fornecer a implementação.

Importar uma implementação

```
<dependencies>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>2.1.214</version>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

Abrir ligação a base de dados

```
String url = "jdbc:h2:mem:testdb;MODE=PostgreSQL";
Connection conn = DriverManager.getConnection(url, "sa", "");
```

Executar a criação de uma tabela (DDL)

- **Statement**: onde é definida uma instrução a ser executada. É possivel definir o SQL ou mais tarde no `executeUpdate...`
- **statement.executeUpdate**: onde o sql é executado
- Necessita sempre de uma **Connection**

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

public class DDLExample {
    public static void main(String[] args) {
        String createTableSQL = "CREATE TABLE IF NOT EXISTS users ("
                + "id BIGINT AUTO_INCREMENT PRIMARY KEY, "
                + "name VARCHAR(255) NOT NULL)";

        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement()) {

            // Executar o comando DDL
            stmt.executeUpdate(createTableSQL);
```

```
            System.out.println("Tabela 'users' criada com sucesso!");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Realizar um SELECT

- **ResultSet**: Contem as linhas resultado da query (Iterator)
- **executeQuery**: indica que pretende receber linhas

```
import java.sql.*;

public class SelectExample {
    public static void main(String[] args) {
        try (Connection conn = DriverManager.getConnection("jdbc:h2:mem:test",
"sa", "")) {
            String query = "SELECT id,nome FROM pessoas";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {
                int id = rs.getInt("id");
                String nome = rs.getString("nome");
                System.out.println("ID: " + id + ", Nome: " + nome);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

É possivel parameterizar os **Statement** usando o "?" no lugar do valor:

- **PreparedStatement**: permite parameterizar as instruções

```
PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM pessoas WHERE nome
= ?");
pstmt.setString(1, "João");
ResultSet rs = pstmt.executeQuery();
// ...
```

Realizar INSERT, UPDATE, DELETE

- **PreparedStatement**: permite parameterizar as instruções

- **executeUpdate**: indica que o Statement não recebe linhas, mas valores como escalares, normalmente o número de linhas afectadas

```
import java.sql.*;

public class InsertComParametros {
    public static void main(String[] args) {
        try {
            // Conexão com o banco de dados
            Connection conn =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/mydatabase", "user",
"password");

            // Preparando a instrução SQL com placeholders (?)
            String sql = "INSERT INTO pessoas (nome, idade) VALUES (?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);

            // Valores dos parâmetros
            pstmt.setString(1, "João da Silva");
            pstmt.setInt(2, 30);

            // Executando a instrução
            int rowsInserted = pstmt.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("Um novo registro foi inserido.");
            }

            // Fechando as conexões
            pstmt.close();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Realizar a chamada a um stored procedure

- **CallableStatement**: o Statement para chamar stored procedures

```
import java.sql.*;

public class ChamadaStoredProc {
    public static void main(String[] args) {
        try {
            // Conexão com o banco de dados H2
            Connection conn = DriverManager.getConnection("jdbc:h2:mem:test",
"sa", "");

            // Criando um CallableStatement
```

```
            CallableStatement stmt = conn.prepareCall("{call inserirPessoa(?,
?)}");

            // Setando os parâmetros
            stmt.setString(1, "João");
            stmt.setInt(2, 30);

            // Executando o Stored Procedure
            stmt.execute();

            System.out.println("Pessoa inserida com sucesso!");

            // Fechando as conexões
            stmt.close();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```