**Acesso a Banco de Dados com JDBC**

**Preparação do Ambiente**

1. **Instale o JDBC Driver for SQL Server em um arquivo zip no site learn.microsoft.com para instalar o driver necessário para que sua aplicação Java se conecte a uma banco de dados SQLServer.**
2. **Exporte os arquivos jar de dentro do arquivo zip para uma pasta java-libs no Disco Local: C do computador.**
3. **Insira a nova biblioteca no Eclipse.**
   1. **No Eclipse, vá para Window>Preferences>Java>Build Path>User Libraries;**
   2. **Crie uma nova biblioteca com o nome SQLServerConnector;**
   3. **Vincule os arquivos .jar exportados à biblioteca SQLServerConnector.**
4. **Adicione a biblioteca ao seu Java Project.**

**Programando o Sistema**

1. **Crie um arquivo db.properties dentro do Java Project e fora da pasta src.**
2. **Insira dentro dele:**

url=jdbc:sqlserver://(nome\_do\_servidor):1433;database=(nome\_do\_bancoDeDados);**encrypt**=true;**trustServerCertificate**=false;**loginTimeout**=30;

user=(usuário)@(nome\_do\_servidor)

password=(senha)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**encrypt**: Exige que a conexão com o SQL Server seja criptografada com SSL/TLS.

**trustServerCertificate**: Define se o JDBC deve verificar a validação do certificado SSL do servidor. Definido como false indica para o programa que ele deve verificar se o certificado é emitido por uma autoridade certificadora válida.

**loginTimeOut**: Define o tempo máximo que o JDBC vai esperar para estabelecer a conexão com o Banco de Dados.

1. **Crie uma exceção DbException no pacote exception.**

**package** exception;

**public** **class** DbException **extends** RuntimeException{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** DbException(String msg) {

**super**(msg);

}

}

1. **Crie os objetos Seller e Department que serão resgatados do banco de dados.**
2. **Crie a classe DB para utilizá-la para estabelecer conexão com o banco de dados.**
3. **Crie o objeto de acesso a dados DAO (Data Access Object) para que a aplicação manipule e resgate dados do banco de dados com base no objeto pretendido. A princípio, deve-se criar as interfaces que serão implementadas pelo DAO. Isso é fundamental para que futuras mudanças de tecnologia sejam possíveis e que exijam manipular apenas a classe que implementa a interface e não o sistema como um todo.**

package model.dao;

import java.util.List;

import model.Department;

import model.Seller;

public interface SellerDao {

void insert(Seller obj);

void update(Seller obj);

void deleteById(Integer id);

Seller findById(Integer id);

List<Seller> findAll();

List<Seller> findByDepartment(Department department);

}

package model.dao;

import java.util.List;

import model.Department;

public interface DepartmentDao {

void insert(Department obj);

void update(Department obj);

void deleteById(Integer id);

Department findById(Integer id);

List<Department> findAll();

}

1. **Crie a exceção DBIntegrityException para lançar operar sob operações críticas quanto a manipulação do banco de dados, como restrições de DELETE e UPDATE sem cláusulas WHERE.**
2. **Crie as classes DAO que irão manipular e resgatar os dados do banco de dados.**
   1. **Para executar querys que contenham atributos variáveis dentro das suas cláusulas utiliza-se o PreparedStatement:**

Connection conn = null;

PreparedStatement st = null;

try {

conn = DB.getConnection();

st = conn.prepareStatement(

"UPDATE seller "

+ "SET baseSalary = baseSalary + ?"

+ "WHERE "

+ "departmentId = ?"

);

st.setDouble(1, 200.0); //Inserindo atributos na query

st.setInt(2, 2);

int rowsAffected = st.executeUpdate();

}catch(SQLException e) {

e.printStackTrace();

}

* 1. **Para executar SELECT sem que nenhum atributo variável seja colocado dentro da query, utiliza-se Statement:**

Connection conn = null;

Statement st = null;

ResultSet rs = null;

try {

conn = DB.getConnection();

st= conn.createStatement();

rs = st.executeQuery("select \* from department");

while(rs.next()) {

System.out.println(rs.getInt("id") + ", " + rs.getString("nome"));

}

}catch(SQLException e){

e.printStackTrace();

}

* 1. **Para resgatar dados ao invés de enviar utiliza-se o ResultSet para obter-se um objeto com corpo de tabela:**

int rowsAffected = st.executeUpdate();

if(rowsAffected > 0) {

System.out.println("Done! Rows affected: " + rowsAffected);

ResultSet rs = st.getGeneratedKeys(); //Recolhe uma tabela com as primary Keys dos registros no banco de dados

while(rs.next()) {

System.out.println(rs.getInt(1)); //Resgata dados da coluna 1

}

}else {

System.out.println("Falhou. Nenhuma linha alterada!");

}

* 1. **Os códigos acima não demonstram isso, mas todas as variáveis PreparedStatement, Statement e ResultSet devem ser fechadas, inclusive a conexão com o banco.**

finally {

DB.closeResultSet(rs);

DB.closeStatement(st);

DB.closeConnection();

}