**Использование Statement, PreparedStatement, CallableStatement (классы в объекты которых заворачиваются SQL запросы)**

import java.sql.\*;

import java.util.logging.\*;

public class JDBCtest {

public static void main(String[] args) {

**Блок подключения должен делаться через DataSource и JNDI, а не через DriverManager:**

**(**

Connection conn = null;

//URL к базе состоит из протокола:подпротокола://[хоста]:[порта\_СУБД]/[БД] и других\_сведений

String url = "jdbc:postgresql://127.0.0.1:5432/test";

//Имя пользователя БД

String name = "user";

//Пароль

String password = "123456";

try {

//Загружаем драйвер

Class.forName("org.postgresql.Driver");

System.out.println("Драйвер подключен");

//Создаём соединение

conn = DriverManager.getConnection(url, name, password);

System.out.println("Соединение установлено");

**)**

**//Дальше для использования SQL запросов существуют 3 типа объектов:**

**//1.Statement: используется для простых случаев без параметров**

Statement statement = null;

statement = conn.createStatement();

//Выполним запрос

ResultSet result1 = statement.executeQuery( "SELECT \* FROM users where id >2 and id <10");

//result это указатель на первую строку с выборки

//чтобы вывести данные мы будем использовать

//метод next() , с помощью которого переходим к следующему элементу

System.out.println("Выводим statement");

while (result1.next()) {

System.out.println("Номер в выборке #" + result1.getRow()

+ "\t Номер в базе #" + result1.getInt("id")

+ "\t" + result1.getString("username"));

}

// Вставить запись

statement.executeUpdate( "INSERT INTO users(username) values('name')");

//Обновить запись

statement.executeUpdate( "UPDATE users SET username = 'admin' where id = 1");

**//2.PreparedStatement: предварительно компилирует запросы,**

**//которые могут содержать входные параметры**

PreparedStatement preparedStatement = null;

// место вставки нашего запроса теперь **не** в ***executeQuery()***, **а** в ***prepareStatement ()***

preparedStatement = conn.prepareStatement( "SELECT \* FROM users where id > ? and id < ?");

//Устанавливаем в нужную позицию значения определённого типа

preparedStatement.setInt(1, 2);

preparedStatement.setInt(2, 10);

//выполняем запрос

ResultSet result2 = preparedStatement.executeQuery();

System.out.println("Выводим PreparedStatement");

while (result2.next()) {

System.out.println("Номер в выборке #" + result2.getRow() + "\t Номер в базе #" + result2.getInt("id")

+ "\t" + result2.getString("username"));

}

// Вставить запись

preparedStatement = conn.prepareStatement( "INSERT INTO users(username) values(?)");

preparedStatement.setString(1, "user\_name");

//тем же способом можно сделать и UPDATE

//но метод принимает значение без параметров

preparedStatement.executeUpdate();

**//3.CallableStatement: используется для вызова хранимых функций,**

// которые могут содержать входные и выходные параметры

CallableStatement callableStatement = null;

//Вызываем функцию myFunc (хранится в БД)

callableStatement = conn.prepareCall( " { call myfunc(?,?) } ");

//Задаём входные параметры

callableStatement.setString(1, "Dima");

callableStatement.setString(2, "Alex");

//Выполняем запрос

ResultSet result3 = callableStatement.executeQuery();

//Если CallableStatement возвращает несколько объектов ResultSet,

//то нужно выводить данные в цикле с помощью метода next() *(как до этого)*

//у меня функция возвращает один объект

result3.next();

System.out.println(result3.getString("MESSAGE"));

//если функция вставляет или обновляет, то используется метод executeUpdate()

} catch (Exception ex) {

//выводим наиболее значимые сообщения

Logger.getLogger(JDBCtest.class.getName()).log(Level.SEVERE, null, ex);

} finally {

if (connection != null) {

try {

connection.close();

} catch (SQLException ex) {

Logger.getLogger(JDBCtest.class.getName()).log(Level.SEVERE, null, ex);

} } } } }