Песочница

ОЖИДАЮТ ПРИГЛАШЕНИЕ ПОЛУЧИЛИ ПРИГЛАШЕНИЕ

pull 21 февраля 2012 в 23:30

Урок JDBC в примерах

Чулан*

Здраствуйте. Решил сделать необычный урок о JDBC. Данный урок написан ввиде класса с комментариями, которые подробно описывают методы и классы использованные для написания программы. В уроке написаны основные способы работы с БД используя JDBC.

JDBC – это стандарт взаимодействия приложиния с различными СУБД. JDBC основан на концепции драйверов, позволяющей получать соединение с БД по специальному url.

Из определения выходит, что нам нужен драйвер, его мы можем взять на сайте производителя СУБД. Скачиваем драйвер и потом импорти библиотеку в проект.

Далее будет один код и описание к нему, надеюсь основные принципы работы с JDBC будут понятны и вы сможете написать программу.

```
import java.sql.*;
import java.util.logging.*;

public class JDBCtest {

    public static void main(String[] args) {

        Connection connection = 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
            connection = DriverManager.getConnection(url, name, password);
            System.out.println("Соединение установлено");
            //Для использования SQL запросов существуют 3 типа объектов:
            //1.Statement: используется для простых случаев без параметров
            Statement statement = null;

            statement = connection.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 значеня
            preparedStatement = connection.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"));
            }

            preparedStatement = connection.prepareStatement(
                    "INSERT INTO users(username) values(?)");
            preparedStatement.setString(1, "user_name");
            //метод принимает значение без параметров
            //темже способом можно сделать и UPDATE
            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 = connection.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
            //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);
                }
            }
        }

    }
}
```

Метки: java,jdbc,примеры

ПОХОЖИЕ ПУБЛИКАЦИИ

Похожие публикации

15 ноября 2017 в 22:04

Непопулярное в JDBC

7 декабря 2017 в 18:01

Всегда ли нужен Builder в Java?

23 февраля 2018 в 15:33

Простые примеры отправки GET и POST запросов на Java с использованием библиотеки Rest-Assured

5 марта 2018 в 17:35

Как вручную установить Oracle Java на Debian или Ubuntu

20 марта 2018 в 13:04

Ускоряем время сборки и доставки java web приложения