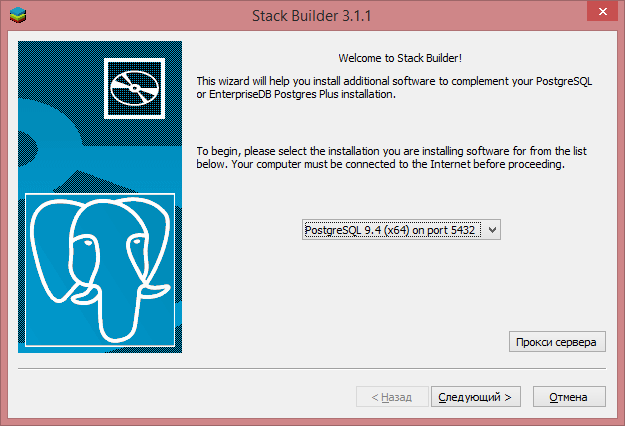
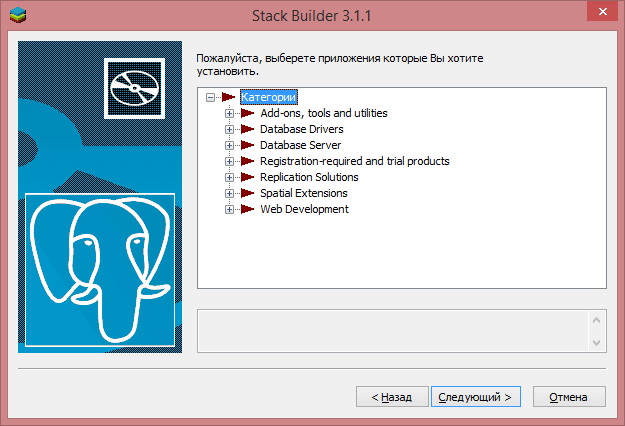
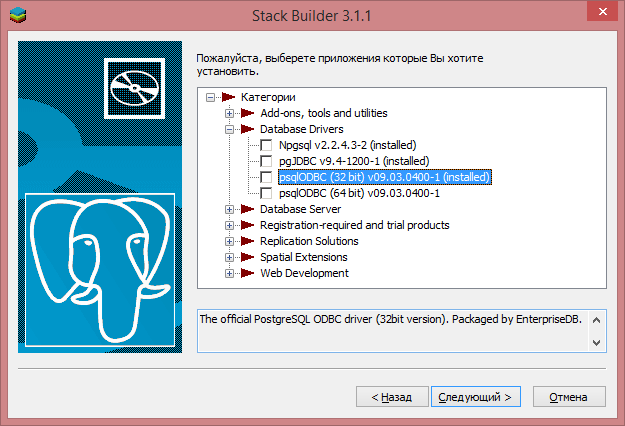
1. Установка драйвера JDBC для PostgreSQL
2. Запустить Application Stack Builder из меню пуск (папка PostgreSQL 9.4)



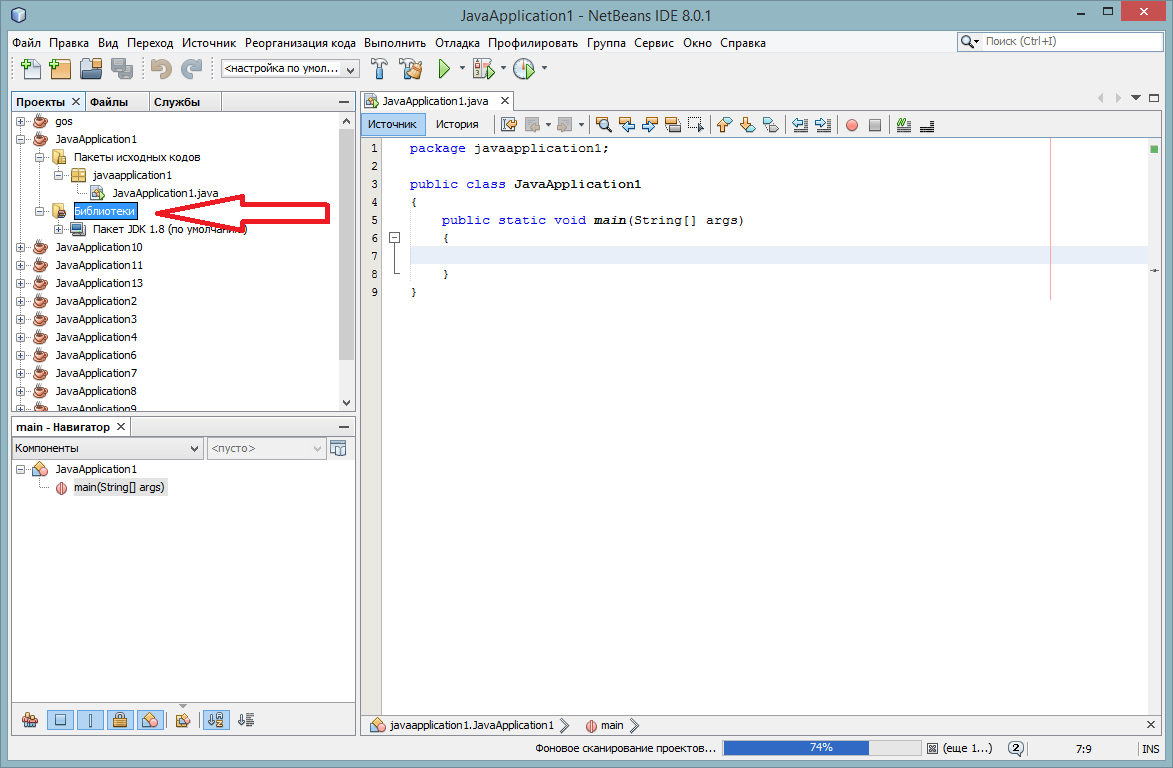
1. В появившемся окне выбрать нужный север и нажать кнопку далее.



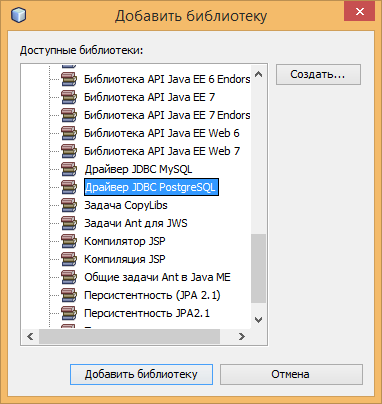
1. Открыть закладку Database Drivers



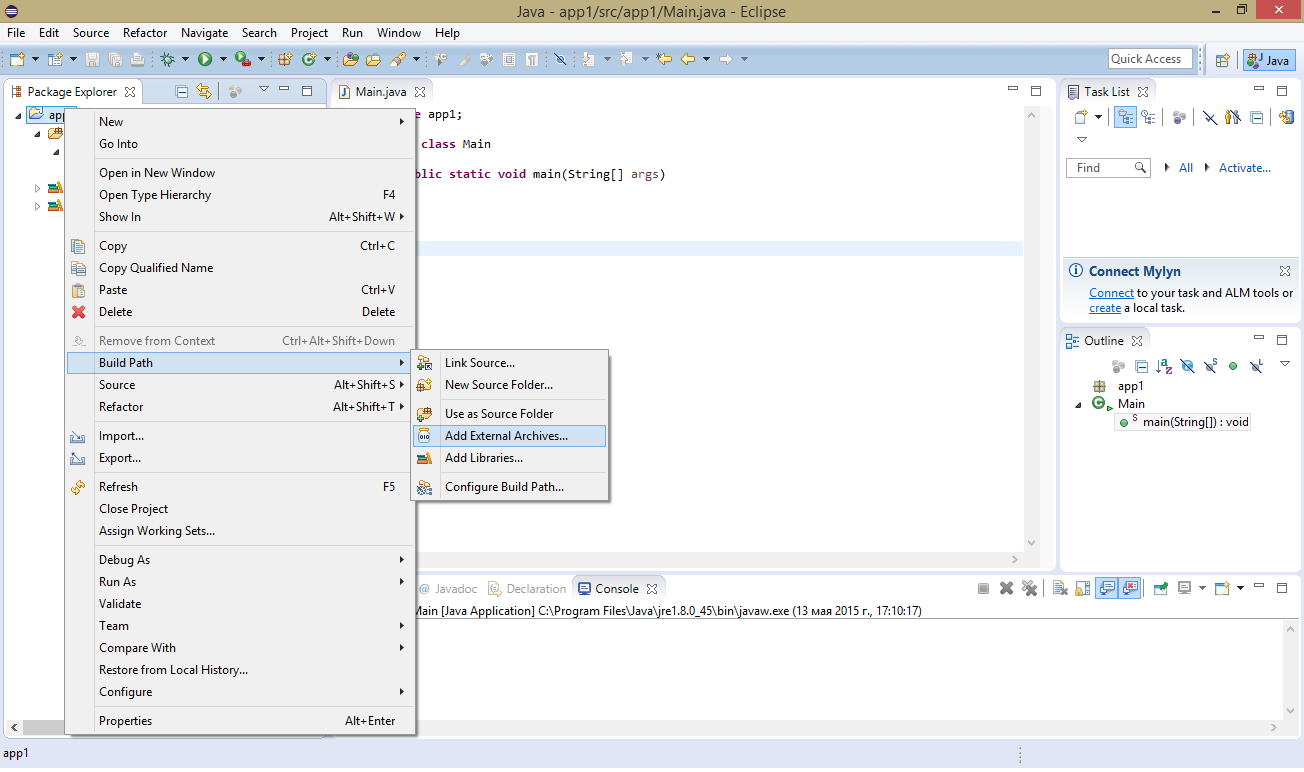
1. Список драйверов:
2. Npgsql – драйвер для .Net
3. pgJDBC – драйвер для Java (нам нужен этот)
4. psglODBC (32 bit) – ODBC драйвер (для Delphi, C++ Builder)
5. psglODBC (64 bit) – ODBC драйвер (для Delphi, C++ Builder)
6. Выбираем нужные драйвера и нажимаем далее до конца установки.
7. Подключение библиотеки драйвера к проекту
   1. NetBeans
      1. В окне проектов открываем наш проект, выбираем библиотеки, нажимаем правую кнопку мыши и выбираем пункт меню «Добавить библиотеку»



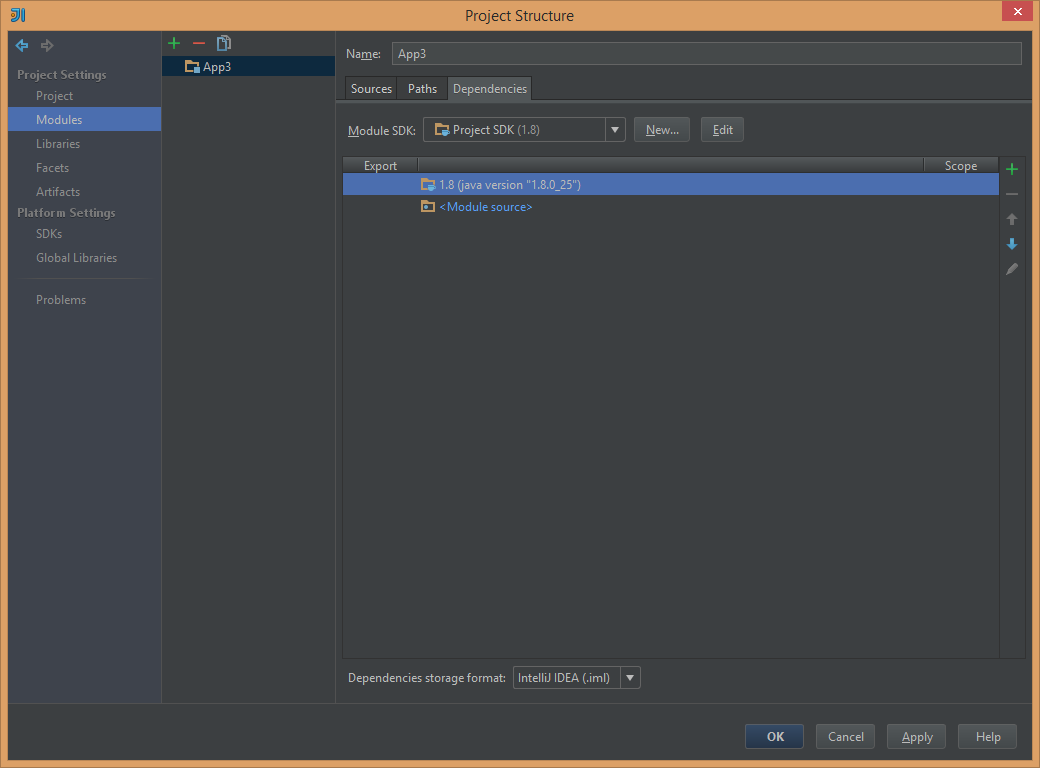
* + 1. В появившемся окне выбираем Драйвер JDBC PostgreSQL и нажимаем «Добавить библиотеку»



* 1. Eclipse
     1. Скачать библиотеку: <https://jdbc.postgresql.org/download/postgresql-9.4-1201.jdbc4.jar>
     2. В «Package Explorer» выбрать проект, нажать на нём правую кнопку мыши, выбрать закладку «Build Path» и в ней нажать «Add External Archives…»



* + 1. Выбрать файл postgresql-9.4-1201.jdbc4.jar
  1. Intellij IDEA
     1. Скачать библиотеку: <https://jdbc.postgresql.org/download/postgresql-9.4-1201.jdbc4.jar>
     2. Открыть свойства модуля, открыть закладку «Dependencies»



* + 1. Нажать на зелёный плюс справа, выбрать пункт меню «2 Library» 🡪«Java»
    2. Выбрать файл postgresql-9.4-1201.jdbc4.jar
    3. В появившемся окне нажать ОК

1. Подключение к базе

В всех выше перечисленных средах подключение выполняется одинаково:

import java.sql.\*;  
  
public class Main  
{  
 public static void main(String[] args)  
 {  
 try  
 {  
 String user = "postgres";  
 String pass = "admin";  
 Class.*forName*("org.postgresql.Driver");  
 Connection dbh = DriverManager.*getConnection*("jdbc:postgresql://127.0.0.1/test", user, pass);  
 } catch (ClassNotFoundException | SQLException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

Строка

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

Загружает драйвер и регистрирует его для автоматического использования с JDBC.  
Для соединения с базой данных используем класс Connection:

Connection dbh = DriverManager.getConnection(url, user, passwd);

Здесь url - это строка, по которой JDBC определяет куда, где и чем устанавливать соединение. Она имеет следующий формат:

jdbc:postgresql:база\_данных

jdbc:postgresql://сервер/база\_данных

jdbc:postgresql://сервер:порт/база\_данных

Где сервер – это адрес сервера, где расположена СУБД, порт – это tcp-порт, на котором будет устанавливаться соединение и база\_данных – это имя базы данных, которую мы хотим использовать.

Установив соединение, мы можем выполнять sql-запросы к нашей базе данных.

Statement st = dbh.createStatement();  
String sql = "select \* from table1";  
ResultSet rs = st.executeQuery(sql);  
while (rs.next())  
{  
 System.*out*.println(rs.getString("name"));  
}  
rs.close();  
st.close();

Для того, чтобы посылать запросы к базе данных, необходимо создать экземпляр класса Statement st. Метод executeQuery этого класса отправляет переданный ему запрос к базе данных и в качестве ответа возвращает результат в виде класса ResultSet.

Перед тем как получать данные результата запроса, которые хранятся в нашем случае в переменной rs, необходимо выполнить метод next(). После выполнения этого метода, вы получаете очередную запись виртуально SQL-таблицы ответа на запрос. Метод next() возвращает true, если на запрос были получены данные или еще есть данные в стеке ответа. В противном случаете next() возвращает false.

Метод getString() возвращает строковую переменную. В качестве аргумента мы передаем название колонки. После использования результатов переменную rs надо закрыть методом close(), чтобы потом можно было повторно ее использовать для других запросов.

В спецификации JDBC вы можете использовать ResultSet, только один раз последовательно прочитав значения. Это означает, что, взяв однажды значение, вы не можете получить его снова. Для этого лучше использовать дополнительную переменную.

Метод executeQuery() удобен лишь в том случае, когда мы ожидаем ответа от базы данных. Специально для запросов, на которые ответ не обязателен в JDBC был введен еще один метод класса Statement. Это – executeUpdate(). К запросам такого типа относится UPDATE, CREATE, INSERT и пр.

Класс PreparedStatement позволяет создавать SQL запросы и подставлять в них значения:

PreparedStatement ps = dbh.prepareStatement("insert into table1 (\"name\") values (?)");

После чего методом setString() заменяем знаки '?' на значения:

String b = "myname";  
ps.setString(1, b);

И после чего выполняем метод executeUpdate():

ps.executeUpdate();

import java.sql.\*;  
public class Main  
{  
 public static void main(String[] args)  
 {  
 try  
 {  
 String user = "postgres";  
 String pass = "admin";  
 Class.*forName*("org.postgresql.Driver");  
 Connection dbh = DriverManager.*getConnection*("jdbc:postgresql://127.0.0.1/test", user, pass);  
  
 PreparedStatement ps = dbh.prepareStatement("insert into table1 (\"name\") values (?)");  
 String b = "myname";  
 ps.setString(1, b);  
 ps.executeUpdate();  
  
 Statement st = dbh.createStatement();  
 String sql = "select \* from table1";  
 ResultSet rs = st.executeQuery(sql);  
 while (rs.next())  
 {  
 System.*out*.println(rs.getString("name"));  
 }  
 rs.close();  
 st.close();  
  
  
 } catch (ClassNotFoundException | SQLException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

1. Вывод данных в таблицу(JTable)

import java.sql.\*;  
import java.util.Vector;  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
  
public class Main  
{  
 public static DefaultTableModel buildTableModel(ResultSet rs) throws SQLException  
 {  
 ResultSetMetaData metaData = rs.getMetaData();  
 Vector<String> columnNames = new Vector<String>();  
 int columnCount = metaData.getColumnCount();  
 for (int column = 1; column <= columnCount; column++)  
 columnNames.add(metaData.getColumnName(column));  
 Vector<Vector<Object>> data = new Vector<Vector<Object>>();  
 while (rs.next())  
 {  
 Vector<Object> vector = new Vector<Object>();  
 for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++)  
 vector.add(rs.getObject(columnIndex));  
 data.add(vector);  
 }  
 return new DefaultTableModel(data, columnNames);  
 }  
  
 public static void main(String[] args)  
 {  
 JFrame f = new JFrame();  
 f.setSize(800, 600);  
 f.setVisible(true);  
 f.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);  
 JTable t;  
 try  
 {  
 String user = "postgres";  
 String pass = "admin";  
 Class.*forName*("org.postgresql.Driver");  
 Connection dbh = DriverManager.*getConnection*("jdbc:postgresql://127.0.0.1/test", user, pass);  
  
 Statement st = dbh.createStatement();  
 String sql = "select \* from table1";  
 ResultSet rs = st.executeQuery(sql);  
 t = new JTable(*buildTableModel*(rs));  
 rs.close();  
 st.close();  
 t.setSize(700, 500);  
 t.setLocation(0, 0);  
 f.add(t);  
 f.repaint();  
 } catch (ClassNotFoundException | SQLException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}