

UTILIZANDO NETBEANS PARA CONECTAR VIA JDBC COM POSTGRES

O que é JDBC?

Bom apesar deste assunto ser básico, no mundo iniciante do qual faço parte, talvez seja necessário. JDBC é na verdade uma API única para acesso a banco de dados a SUN a construiu trabalhando em conjunto com líderes mundiais no setor de banco de dados, dentre as premissas básicas e principais estariam:

- A JDBC deveria ser uma API em nível de SQL;
- A JDBC deveria tirar proveito da experiência das APIs de banco de dados pré existentes;
- A JDBC deveria ser simples;

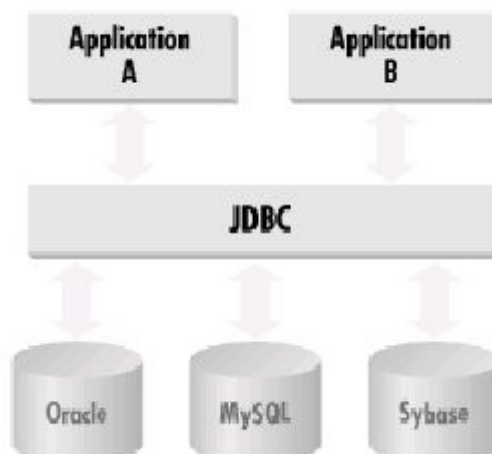
Uma API em nível de SQL significa que a JDBC permite que você escreva instruções SQL, incorporadas ao seu código, permitindo este tráfego entre o mundo do banco de dados e o mundo java, traduzindo dados em objetos e os erros de conexão ocorridos seja exceções(*Exceptions*) enfim a proposta do JDBC é a de ser a interface do programador java com o banco de dados. Com isso proliferou-se as APIs de acesso a banco de dados, provocando um verdadeiro caos, a SUN bem que tentou uma API universal para todos os bancos de dados como a ODBC(*Open DataBase Connectivity*) mas infelizmente ela não é muito bem compatível com o mundo java em que é visada a portabilidade, sendo que essa API é escrita em C e somente "estável" no ambiente windows. A ponte JDBC-ODBC é um ótimo recurso para quem deseja aprender banco de dados, o que acontece é que você estará restrito ao Microsoft Access inutilizando assim todos os benefícios da linguagem java, pois seu sistema só irá rodar onde exista Microsoft Windows com ODBC nativo, creio que não é bem esta sua intenção.

A idéia da JDBC é ser simples e ao mesmo tempo o mais simples possível para desenvolvedores como nós, o critério que a SUN utilizou foi "perguntar se a aplicação de banco de dados lê bem", pois é ela definiu que as tarefas simples e corriqueiras poderiam ser facilmente resolvidas pela interface, ao passo que tarefas muito mais complexas aí sim utilizariam API do banco de dados específicos para essas tarefas.

A estrutura da JDBC

A JDBC atinge seus objetivos através de um conjunto de interfaces java fornecida pelo fabricante do banco de dados, o conjunto das classes que implementam mecanismos para esse banco de dados é denominado *driver* JDBC, que tem por finalidade ocultar aspectos específicos dos bancos de dados, fazendo com que o desenvolvedor fique mais voltado para o código e deixe de se preocupar as peculiaridades de banco de dados. Abaixo podemos ver uma breve visualização da estrutura JDBC:

Figure 3.1. The JDBC architecture



Como fica claro na imagem acima, a API jdbc, é basicamente a intermediária entre sua aplicação e os bancos de dados existentes, estas são normalmente fornecidas pelos **fabricantes** de banco de dados. Bom no nosso caso nós utilizaremos o driver jdbc fornecido pelo **postgresql** no meu caso como estou usando a versão j2se 1.5 vou baixar então o JDBC3 que é o especificado para esta versão, e de acordo com a versão do meu postgres então será o **pg74.215.jdbc3.jar**.

Antes de mais nada vamos ver como adicionar o driver jdbc ao NetBeans 4.0, IDE na qual estou utilizando.

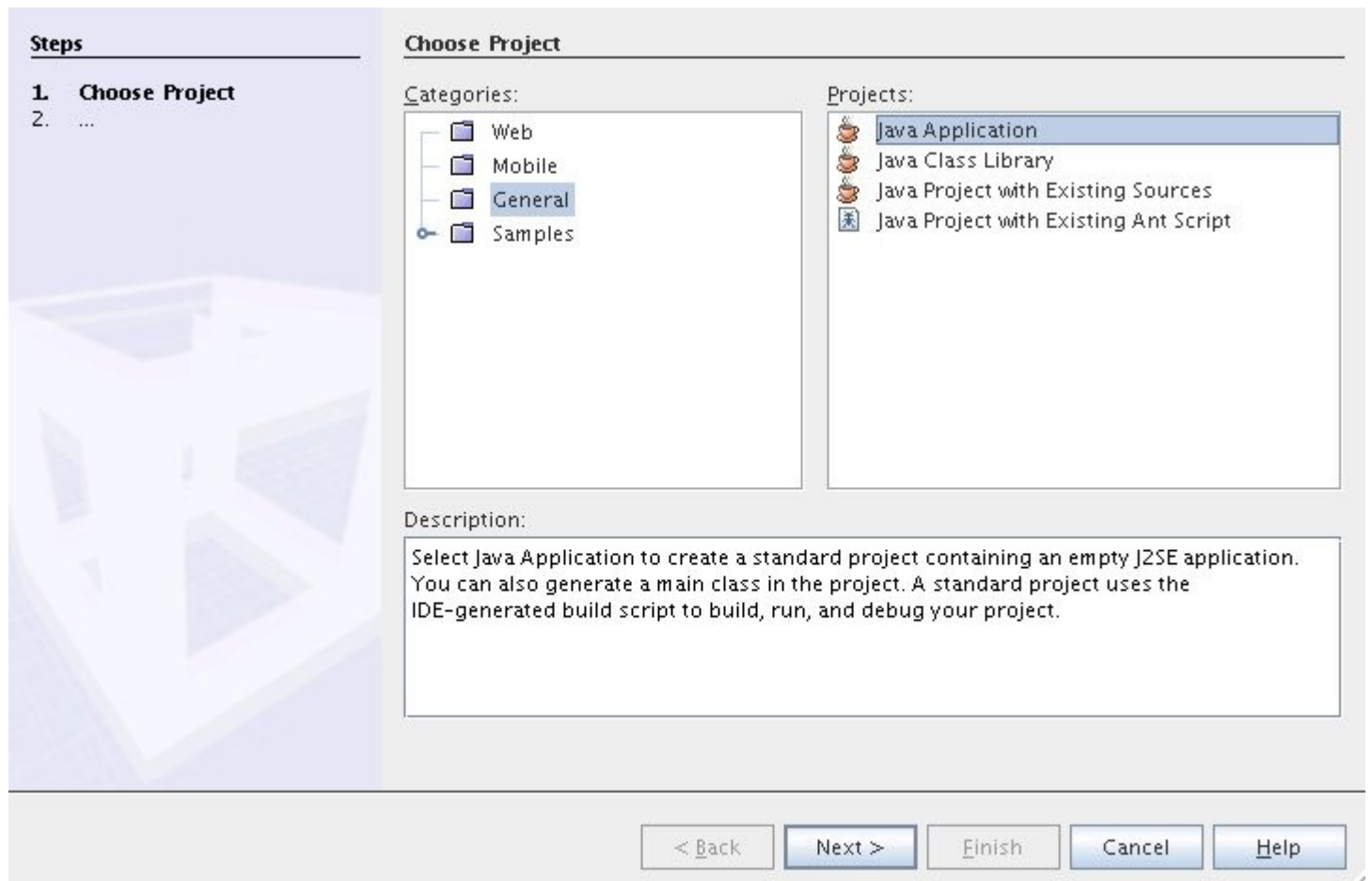
1º Passo

Vamos criar um novo projeto abrindo a IDE e clicando no segundo desenho da esquerda para a direita:




2º Passo

Agora iremos selecionar que tipo de projeto é o nosso e logo após clicar em next:



3º Passo

Agora vamos dar nome ao nosso projeto e clicar em finish



Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

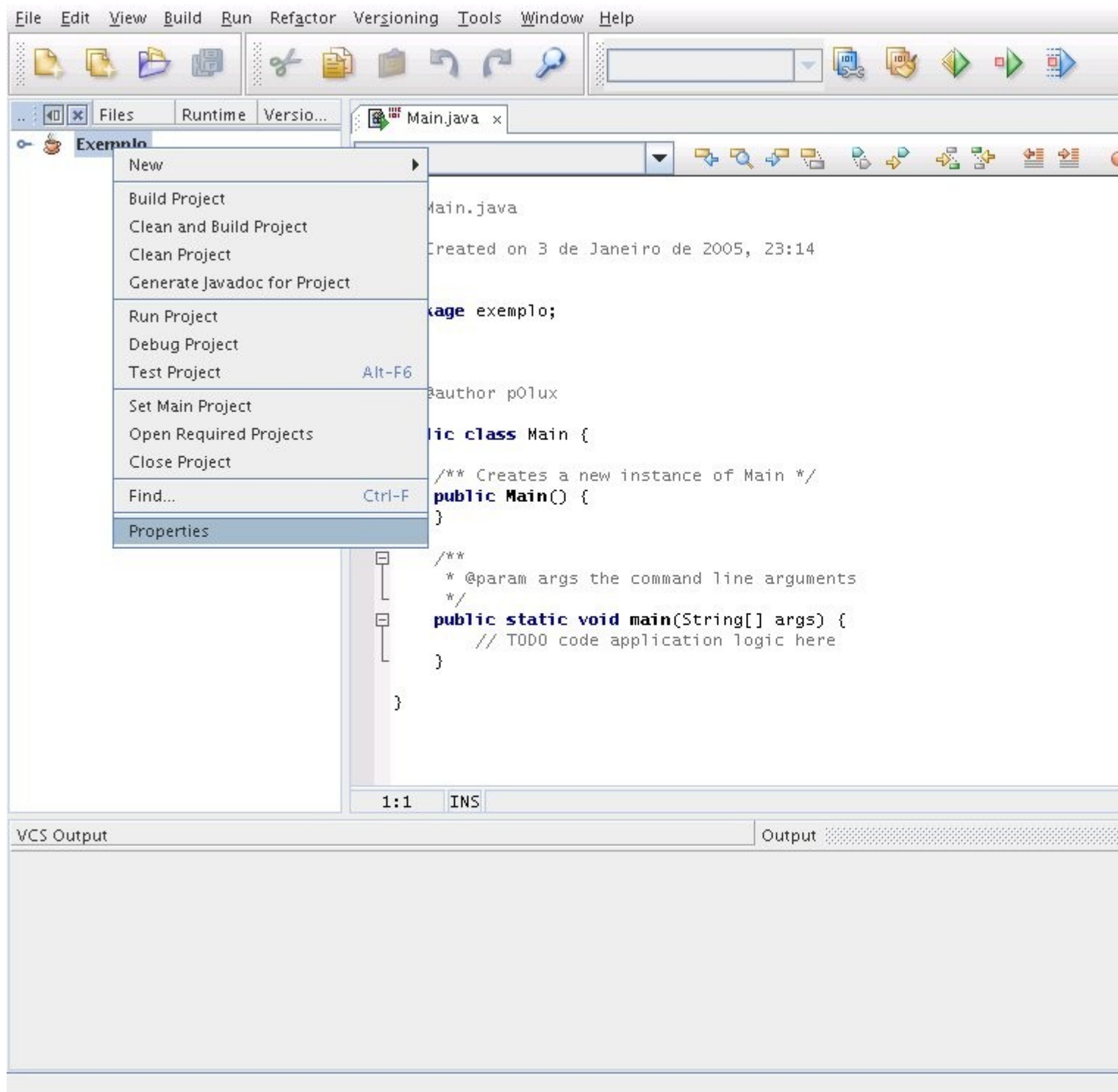
Project Location:

Project Folder:

☒ Set as Main Project

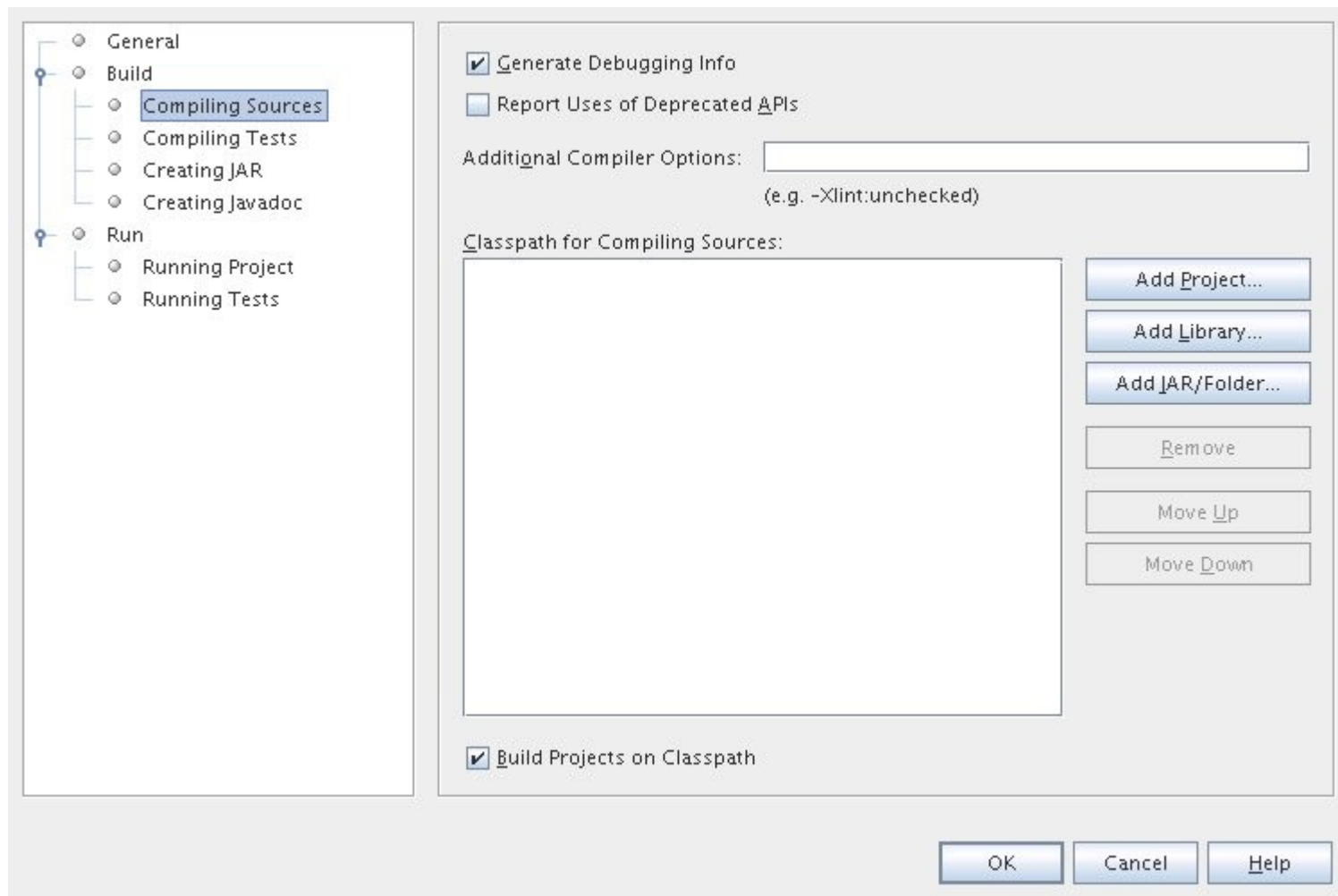
☒ Create Main Class:

4° Agora iremos setar o classpath do nosso netbeans para que ele possa trabalhar com o driver jdbc do postgres, primeiramente iremos clicar com o direito do mouse no projeto, e ir em properties:



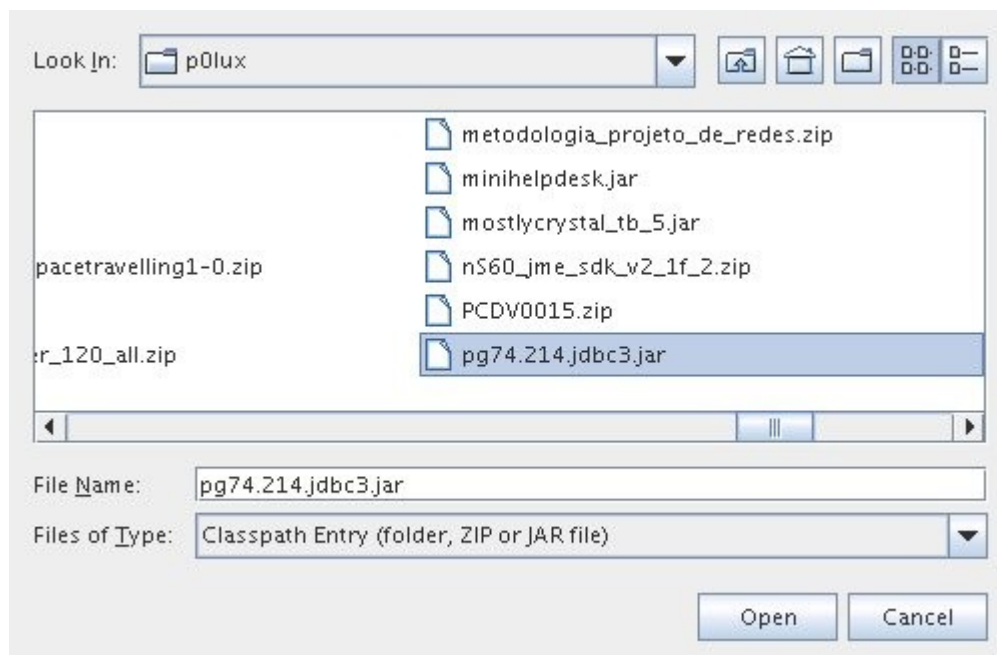
5º Passo

Agora iremos até a sessão *compiling sources* para que possamos adicionar nosso driver jdbc, clicaremos no botão *Add JAR/Folder* para localizarmos o driver:



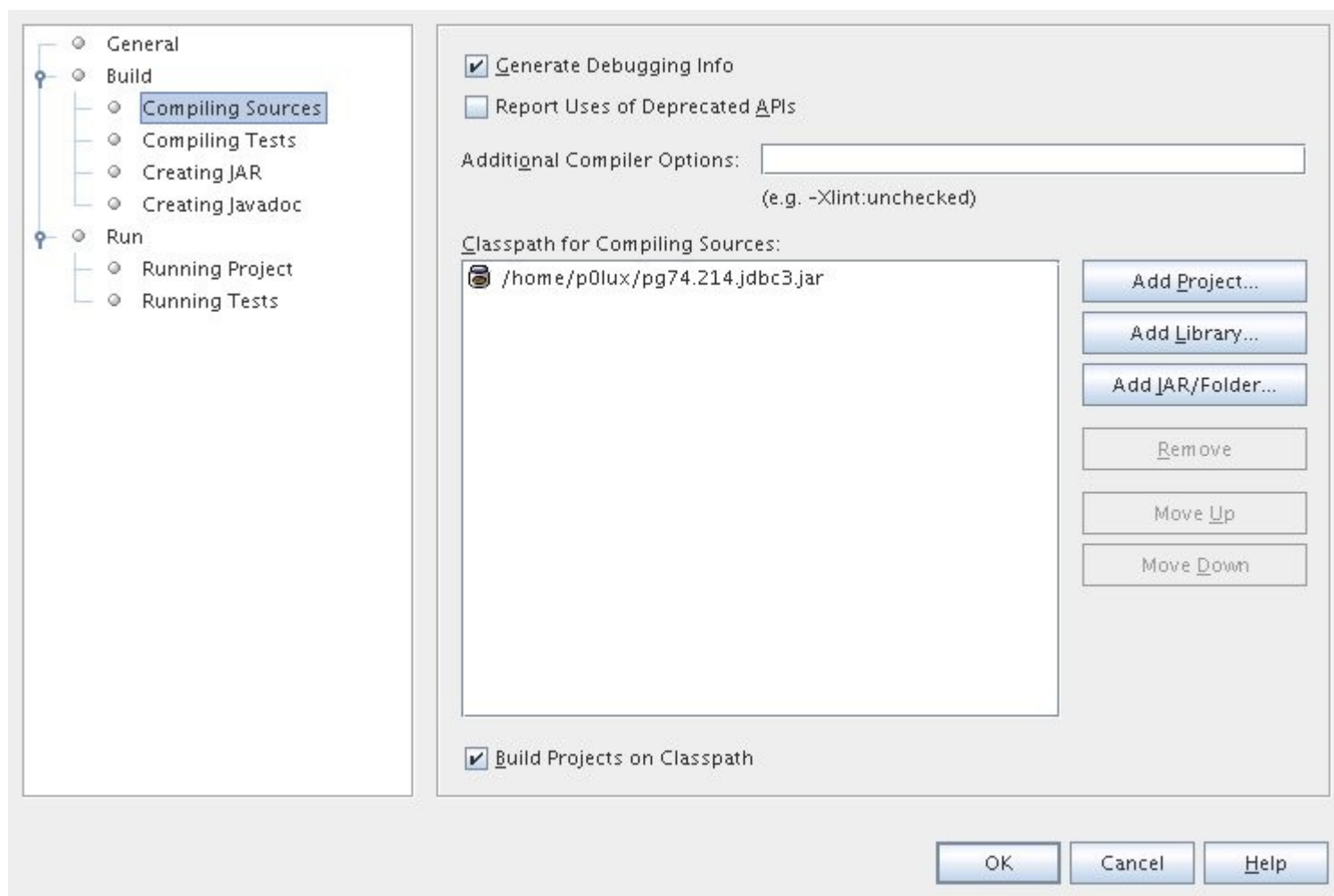
6º Passo

Agora estamos seleccionando nosso driver jdbc e posteriormente daremos um ok:



7º Passo

Por fim iremos clicar em *finish* pois nosso driver jdbc já se encontra adicionado ao classpath do netbeans:



Configurada nossa IDE para criação de nossas classes vamos conhecer algumas classes que fazem parte da api *java.sql* e que nos será útil para fazer um teste inicial de conexão ao banco de dados, são elas: *Connection* e *DriverManager*.

java.sql.Connection

Sua função é fazer a conexão lógica com o banco de dados, nos possibilitando enviar instruções sql ao banco de dados, controlar o aborto ou registro de conexões existentes.

java.sql.DriverManager

Responsável por selecionar o driver apropriado para a conexão ao banco de dados através da url que lhe for passada, recebendo como parâmetros o endereço, senha e usuário.

Primeiro exemplo, efetuando conexão jdbc:

Código:

```
//Classe de conexao ao banco de dados
package temp;

import java.sql.Connection;
import java.sql.DriverManager;

public class Conexao {

    private Connection con;
    private String driver;
    private String user;
    private String pass;
    private String endereco;

    public Conexao() {
```

```

try {

driver = "org.postgresql.Driver";
user = "postgres";
pass = "12345678";
endereco = "jdbc:postgresql://localhost/test";

Class.forName(driver);
con = DriverManager.getConnection(endereco, user, pass);

} catch (Exception e) {

System.out.println("Nao foi possivel efetuar a conexao!");
e.printStackTrace();

}
}

public static void main (String args[]) {

new Conexao();

}
}

```

Então vamos a uma breve análise do código acima, primeiramente importamos classes que nos seriam necessárias e já descritas acima, logo depois é declarada a classe *Conexao* como pública e inicializadas as variáveis de classe, logo após é criado um construtor que atribui valores a essas variáveis, em seguida *Class.forName(driver)* é encarregado de registrar o Driver.

No nosso segundo exemplo iremos iniciar o código para efetuar o cadastro na base de dados, quero ressaltar que este código explora o cadastro no banco de dados da forma mais primitiva possível assim como os outros códigos que serão exemplificados, o efeito deles é apenas didático:

Código:

```

//Classe para cadastro

package temp;

import java.sql.Statement;
import java.sql.Connection;
import java.sql.DriverManager;

public class Cadastro {

String driver, user, pass, endereco, chamado;
Connection con = null;

public Cadastro() {

try {

driver = "org.postgresql.Driver";
user = "postgres";
pass = "12345678";
endereco = "jdbc:postgresql://localhost/test";
chamado = "José Pereira", '2', '07/12/2004', '11:23', '14:56', 'Problemas ao conectar', 'Foram apresentados problemas no modem do client';

Class.forName(driver);

```

```

con = DriverManager.getConnection(endereco, user, pass);

Statement statement = con.createStatement();

statement.executeUpdate("INSERT INTO minihelpdesk (" +
"solicitante, num_chamado, data_solicitacao, hora_entrada, hora_saida, assunto, descricao)" +
"VALUES(" + chamado + ");");

con.close();

}

catch (Exception e) {

System.err.println("Problemas apresentados na operacao de cadastro");
e.printStackTrace();
}
}

public static void main(String args[]) {

new Cadastro();
}

}

```

Primeiramente importamos as classes necessárias do pacote java.sql são elas: *Statement*, *Connection*, *DriverManager*.

java.sql.Statement

Classe que nos permite executar as instruções sql e nos retornar os resultados que essas instruções produz, é ela que receberá nosso objeto da classe *Connection*. Não iremos explorar as outras classes restantes pois a explicação destas já foi explorada no texto acima.

Agora iremos a nossa classe de consulta, neste exemplo iremos fazer uma simples consulta ao cadastro que efetuamos.

Código:

```

import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.DriverManager;

public class Consulta {

private String driver = "org.postgresql.Driver";
private String user = "postgres";
private String endereco = "jdbc:postgresql://localhost/teste";
private String passwd = "12345678";

private Connection con;
private ResultSet rs;

private int contador=1;

public Consulta() {

try {

Class.forName(driver);

con = DriverManager.getConnection(endereco, user, passwd);

Statement statement = con.createStatement();

```



```
rs = statement.executeQuery("SELECT * FROM minihelpdesk WHERE num_chamado=1");

while(rs.next()) {

for (int i = 1; i < 8; i++) {

System.out.println(rs.getString(i));
}
}

} catch (Exception e) {

e.printStackTrace();
}

}

public static void main(String[] args) {

new Consulta();
}
}
```

java.sql.ResultSet

Classe responsável por armazenar as consultas feitas ao banco de dados pela referência *statement*.

Bom espero ter ajudado, este texto visa apenas dar uma visão muito básica de jdbc.

** Texto e imagens baseados no livro "Programação para banco e dados JDBC & Java - O'reilly"*

** Dúvidas e comentários devem ser postados diretamente no **fórum** para que possamos compartilhar conhecimento.*