



PostgreSQL - JAVA Interface

Advertisements

⬅ Previous Page

Next Page ➡

Installation

Before we start using PostgreSQL in our Java programs, we need to make sure that we have PostgreSQL JDBC and Java set up on the machine. You can check Java tutorial for Java installation on your machine. Now let us check how to set up PostgreSQL JDBC driver.

Download the latest version of *postgresql-(VERSION).jdbc.jar* from [postgresql-jdbc repository](#).

Add downloaded jar file *postgresql-(VERSION).jdbc.jar* in your class path, or you can use it along with `-classpath` option as explained below in the examples.

The following section assumes you have little knowledge about Java JDBC concepts. If you do not have, then it is suggested to spent half and hour with [JDBC Tutorial](#) to become comfortable with concepts explained below.

Connecting To Database

The following Java code shows how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.

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

public class PostgreSQLJDBC {
    public static void main(String args[]) {
        Connection c = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/testdb",
                    "postgres", "123");
        } catch (Exception e) {
            e.printStackTrace();
            System.err.println(e.getClass().getName()+": "+e.getMessage());
            System.exit(0);
        }
    }
}
```

```

    }
    System.out.println("Opened database successfully");
}
}

```

Before you compile and run above program, find **pg_hba.conf** file in your PostgreSQL installation directory and add the following line –

```

# IPv4 local connections:
host      all             all             127.0.0.1/32          md5

```

You can start/restart the postgres server in case it is not running using the following command –

```

[root@host]# service postgresql restart
Stopping postgresql service:          [ OK ]
Starting postgresql service:         [ OK ]

```

Now, let us compile and run the above program to connect with testdb. Here, we are using **postgres** as user ID and **123** as password to access the database. You can change this as per your database configuration and setup. We are also assuming current version of JDBC driver **postgresql-9.2-1002.jdbc3.jar** is available in the current path.

```

C:\JavaPostgresIntegration>javac PostgreSQLJDBC.java
C:\JavaPostgresIntegration>java -cp c:\tools\postgresql-9.2-1002.jdbc3.jar;C:\JavaPostgresInte
Open database successfully

```

Create a Table

The following Java program will be used to create a table in previously opened database. Make sure you do not have this table already in your target database.

```

import java.sql.*;

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

public class PostgreSQLJDBC {
    public static void main( String args[] ) {
        Connection c = null;
        Statement stmt = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/testdb",
                    "manisha", "123");
            System.out.println("Opened database successfully");

            stmt = c.createStatement();
            String sql = "CREATE TABLE COMPANY " +
                "(ID INT PRIMARY KEY     NOT NULL," +
                " NAME           TEXT      NOT NULL, " +
                " AGE              INT       NOT NULL, " +

```

```

        " ADDRESS          CHAR(50), " +
        " SALARY           REAL)";
stmt.executeUpdate(sql);
stmt.close();
c.close();
} catch ( Exception e ) {
    System.err.println( e.getClass().getName()+" : "+ e.getMessage() );
    System.exit(0);
}
System.out.println("Table created successfully");
}
}

```

When a program is compiled and executed, it will create the COMPANY table in **testdb** database and will display the following two lines –

```

Opened database successfully
Table created successfully

```

INSERT Operation

The following Java program shows how we can create records in our COMPANY table created in above example –

```

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

public class PostgreSQLJDBC {
    public static void main(String args[]) {
        Connection c = null;
        Statement stmt = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/testdb",
                    "manisha", "123");
            c.setAutoCommit(false);
            System.out.println("Opened database successfully");

            stmt = c.createStatement();
            String sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
                + "VALUES (1, 'Paul', 32, 'California', 20000.00 );";
            stmt.executeUpdate(sql);

            sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
                + "VALUES (2, 'Allen', 25, 'Texas', 15000.00 );";
            stmt.executeUpdate(sql);

            sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
                + "VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );";
            stmt.executeUpdate(sql);

            sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
                + "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );";
            stmt.executeUpdate(sql);

            stmt.close();
            c.commit();
            c.close();
        } catch (Exception e) {

```

```

        System.err.println( e.getClass().getName()+" : "+ e.getMessage() );
        System.exit(0);
    }
    System.out.println("Records created successfully");
}
}

```

When the above program is compiled and executed, it will create given records in COMPANY table and will display the following two lines –

```

Opened database successfully
Records created successfully

```

SELECT Operation

The following Java program shows how we can fetch and display records from our COMPANY table created in above example –

```

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

public class PostgreSQLJDBC {
    public static void main( String args[] ) {
        Connection c = null;
        Statement stmt = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/testdb",
                    "manisha", "123");
            c.setAutoCommit(false);
            System.out.println("Opened database successfully");

            stmt = c.createStatement();
            ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );
            while ( rs.next() ) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                int age = rs.getInt("age");
                String address = rs.getString("address");
                float salary = rs.getFloat("salary");
                System.out.println( "ID = " + id );
                System.out.println( "NAME = " + name );
                System.out.println( "AGE = " + age );
                System.out.println( "ADDRESS = " + address );
                System.out.println( "SALARY = " + salary );
                System.out.println();
            }
            rs.close();
            stmt.close();
            c.close();
        } catch ( Exception e ) {
            System.err.println( e.getClass().getName()+" : "+ e.getMessage() );
            System.exit(0);
        }
        System.out.println("Operation done successfully");
    }
}

```

When the program is compiled and executed, it will produce the following result –

```
Opened database successfully
```

```
ID = 1
```

```
NAME = Paul
```

```
AGE = 32
```

```
ADDRESS = California
```

```
SALARY = 20000.0
```

```
ID = 2
```

```
NAME = Allen
```

```
AGE = 25
```

```
ADDRESS = Texas
```

```
SALARY = 15000.0
```

```
ID = 3
```

```
NAME = Teddy
```

```
AGE = 23
```

```
ADDRESS = Norway
```

```
SALARY = 20000.0
```

```
ID = 4
```

```
NAME = Mark
```

```
AGE = 25
```

```
ADDRESS = Rich-Mond
```

```
SALARY = 65000.0
```

```
Operation done successfully
```

UPDATE Operation

The following Java code shows how we can use the UPDATE statement to update any record and then fetch and display updated records from our COMPANY table –

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

public class PostgreSQLJDBC {
    public static void main( String args[] ) {
        Connection c = null;
        Statement stmt = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/testdb",
                    "manisha", "123");
            c.setAutoCommit(false);
            System.out.println("Opened database successfully");
```

```
stmt = c.createStatement();
String sql = "UPDATE COMPANY set SALARY = 25000.00 where ID=1;";
stmt.executeUpdate(sql);
c.commit();

ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );
while ( rs.next() ) {
    int id = rs.getInt("id");
    String name = rs.getString("name");
    int age = rs.getInt("age");
    String address = rs.getString("address");
    float salary = rs.getFloat("salary");
    System.out.println( "ID = " + id );
    System.out.println( "NAME = " + name );
    System.out.println( "AGE = " + age );
    System.out.println( "ADDRESS = " + address );
    System.out.println( "SALARY = " + salary );
    System.out.println();
}
rs.close();
stmt.close();
c.close();
} catch ( Exception e ) {
    System.err.println( e.getClass().getName()+" : "+ e.getMessage() );
    System.exit(0);
}
System.out.println("Operation done successfully");
}
```

When the program is compiled and executed, it will produce the following result –

Opened database successfully

ID = 2

NAME = Allen

AGE = 25

ADDRESS = Texas

SALARY = 15000.0

ID = 3

NAME = Teddy

AGE = 23

ADDRESS = Norway

SALARY = 20000.0

ID = 4

NAME = Mark

AGE = 25

ADDRESS = Rich-Mond

SALARY = 65000.0

ID = 1

NAME = Paul

AGE = 32

ADDRESS = California

```
SALARY = 25000.0
```

```
Operation done successfully
```

DELETE Operation

The following Java code shows how we can use the DELETE statement to delete any record and then fetch and display remaining records from our COMPANY table –

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

public class PostgreSQLJDBC6 {
    public static void main( String args[] ) {
        Connection c = null;
        Statement stmt = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/testdb",
                    "manisha", "123");
            c.setAutoCommit(false);
            System.out.println("Opened database successfully");

            stmt = c.createStatement();
            String sql = "DELETE from COMPANY where ID = 2;";
            stmt.executeUpdate(sql);
            c.commit();

            ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );
            while ( rs.next() ) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                int age = rs.getInt("age");
                String address = rs.getString("address");
                float salary = rs.getFloat("salary");
                System.out.println( "ID = " + id );
                System.out.println( "NAME = " + name );
                System.out.println( "AGE = " + age );
                System.out.println( "ADDRESS = " + address );
                System.out.println( "SALARY = " + salary );
                System.out.println();
            }
            rs.close();
            stmt.close();
            c.close();
        } catch ( Exception e ) {
            System.err.println( e.getClass().getName()+" : "+ e.getMessage() );
            System.exit(0);
        }
        System.out.println("Operation done successfully");
    }
}
```

When the program is compiled and executed, it will produce the following result –

Opened database successfully

ID = 3

NAME = Teddy

AGE = 23

ADDRESS = Norway

SALARY = 20000.0

ID = 4

NAME = Mark

AGE = 25

ADDRESS = Rich-Mond

SALARY = 65000.0

ID = 1

NAME = Paul

AGE = 32

ADDRESS = California

SALARY = 25000.0

Operation done successfully



⬅ Previous Page

Next Page ➡

Advertisements

Ad



Get PDF



PDF-Convert.co

[VISIT SITE](#)



Tutorials Point (India) Pvt. Ltd.

YouTube 58K



[Write for us](#) [FAQ's](#) [Helping](#) [Contact](#)

Enter email for newsletter

go