



Pomorska Fundacja  
Inicjatyw Gospodarczych

# Relacyjne i nierelacyjne bazy danych - JDBC

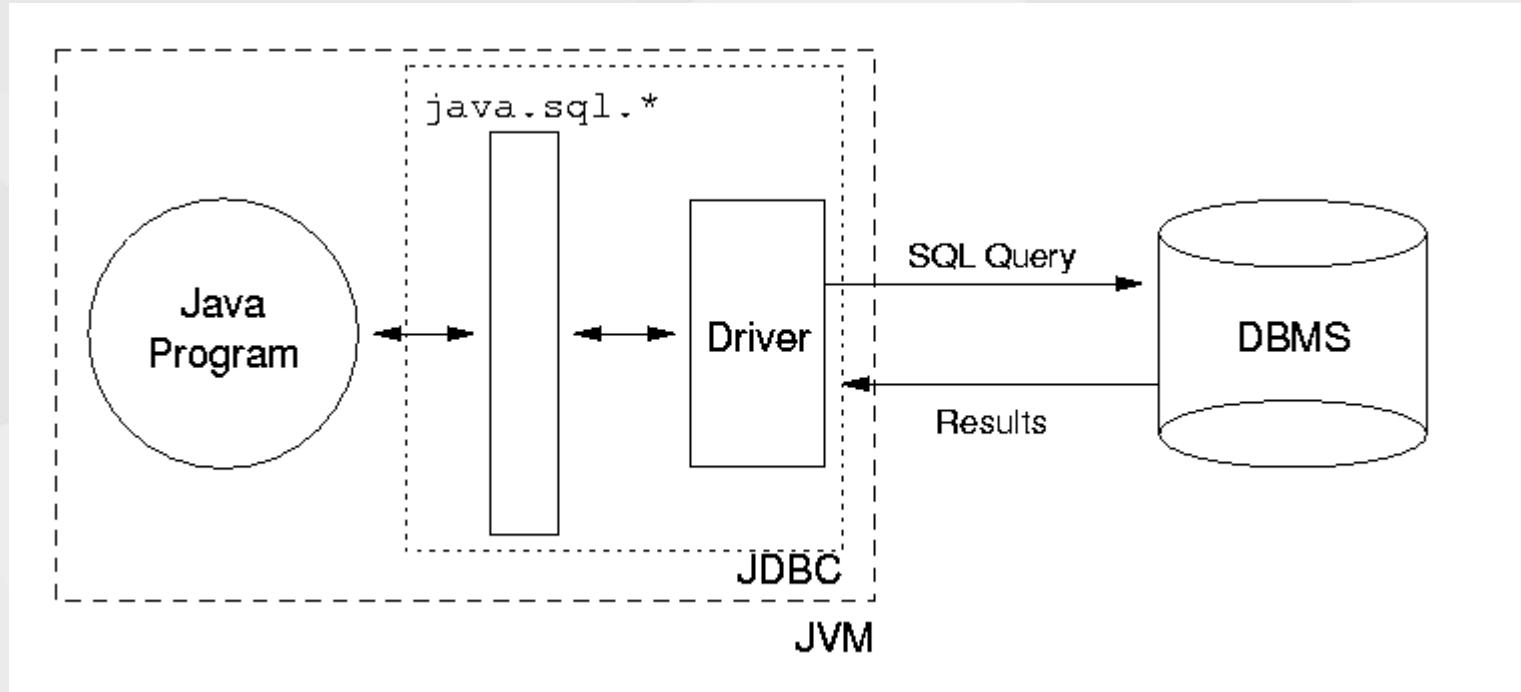
Michał Szymański

# JDBC – Wikipedia definicja

JDBC (ang. Java DataBase Connectivity - łącze do baz danych w języku Java) - interfejs programowania opracowany, umożliwiający niezależnym od platformy aplikacjom napisanym w języku Java porozumiewać się z bazami danych za pomocą języka SQL. Środowisko Java Platform, Standard Edition zawiera API JDBC, natomiast użytkownik musi uzyskać specjalny sterownik JDBC do swojej bazy danych.



# JDBC – Podstawy



# JDBC – Użycie

- 1) Import pakietu
- 2) Załadowanie drivera
- 3) Nawiązanie połączenia
- 4) Stworzenie „Statement”
- 5) Wywołania zapytania
- 6) Przetwarzanie odpowiedzi
- 7) Zamknięcie połączenia



# PRZYKŁAD

# Dodanie dependency Maven

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.6</version>
</dependency>
```

.



# JDBC – Driver

Wczytanie driver'a JDBC

```
Class.forName("com.mysql.jdbc.Driver");
```

Do pobrania ze strony producenta i do dodania do projektu:

<https://dev.mysql.com/downloads/connector/j/5.1.html>

Różne wersje JDBC - niewielkie różnice pomiędzy wersjami.

# JDBC – Nawiązanie połączenia

## Nawiązania połączenia

```
co= DriverManager.getConnection("jdbc:mysql://localhost/test", "root", "mysql");
```

URL dla Postgresa: *jdbc:postgresql://host:port/database*

Oracle: *jdbc:oracle:<drivertype>:<user>/<password>@<database>*



# JDBC – Wykonanie SQL

## Wywołanie zapytania SQL

```
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("SELECT VERSION()");
```

## Dostęp do rezultatów

```
if (rs.next()) {
    System.out.println(rs.getString(1));
}
```



# JDBC – Pobranie danych

```
pst = connection.prepareStatement("SELECT * FROM kontrola.straznik");
    rs = pst.executeQuery();

    while (rs.next()) {
        System.out.print(rs.getInt(1));
        System.out.print(": ");
        System.out.println(rs.getString(2));
    }
```



# JDBC – Wykonanie SQL

## Metody ResultSet:

- ....
- byte getByte(int columnIndex)
- Date getDate(int columnIndex)
- int getInt(int columnIndex)
- String getString(int columnIndex)
- ....

Uwaga na mapowanie typów DB na typy Javy

# JDBC – obsługa błędów i zwalnianie zasobów

```
    } catch (SQLException e) {  
  
        System.out.println("Blad polaczenia do bazy danych");  
        e.printStackTrace();  
        return;  
  
    } finally {  
        try {  
            if (rs != null) rs.close();  
            if (st != null) st.close();  
            if (connection != null) connection.close();  
        } catch (SQLException ex) {  
            System.out.println("Blad zamykania polaczenia");  
            ex.printStackTrace();  
            return;  
        }  
    }  
}
```



# JDBC – PreparedStatement

```
String stm = "INSERT INTO straznik(imie,nazwisko,pensja) VALUES(?, ?,?)";  
pst = connection.prepareStatement(stm);
```

```
    pst.setString(1, "Jan");  
    pst.setString(2, "Jeziorański");  
    pst.setDouble(3, 2300.32);  
    pst.executeUpdate();
```

# JDBC – Transakcje

```
connection.setAutoCommit(false);
stm = "INSERT INTO straznik(imie,nazwisko,pensja) VALUES(?, ?, ?)";
pst = connection.prepareStatement(stm);
pst.setString(1, "Jan");
pst.setString(2, "Jeziorański");
pst.setDouble(3, 2300.32);
pst.executeUpdate();
connection.commit();
} catch (SQLException ex) {
    if (con != null) {
        try {
            con.rollback();
        } catch (SQLException ex1) {
```

