

JAVA I BAZY DANYCH

ZAGADNIENIA:

- wprowadzenie;
- JDBC;
- komunikacja z bazą danych;
- HSQLDB.

MATERIAŁY:

<http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>



BAZY DANYCH - SQL

Tabela				
id	Imie	Nazwisko	Wiek	numer
1	Barbara	Tokarska	47	FSX458721
2	Tomasz	Czyżewski	24	HGJ874398
3	Jerzy	Pietras	35	VTF937836
4	Agata	Kałuża	17	JKD259077

```
SELECT Imie,Nazwisko FROM Tabela WHERE id = 3;  
UPDATE Tabela SET Imie='Marek' WHERE id=4;  
DELETE FROM Tabela WHERE Imie='Tomasz';  
INSERT INTO Tabela (id, Imie) VALUES (5, 'Dorota');
```

JDBC

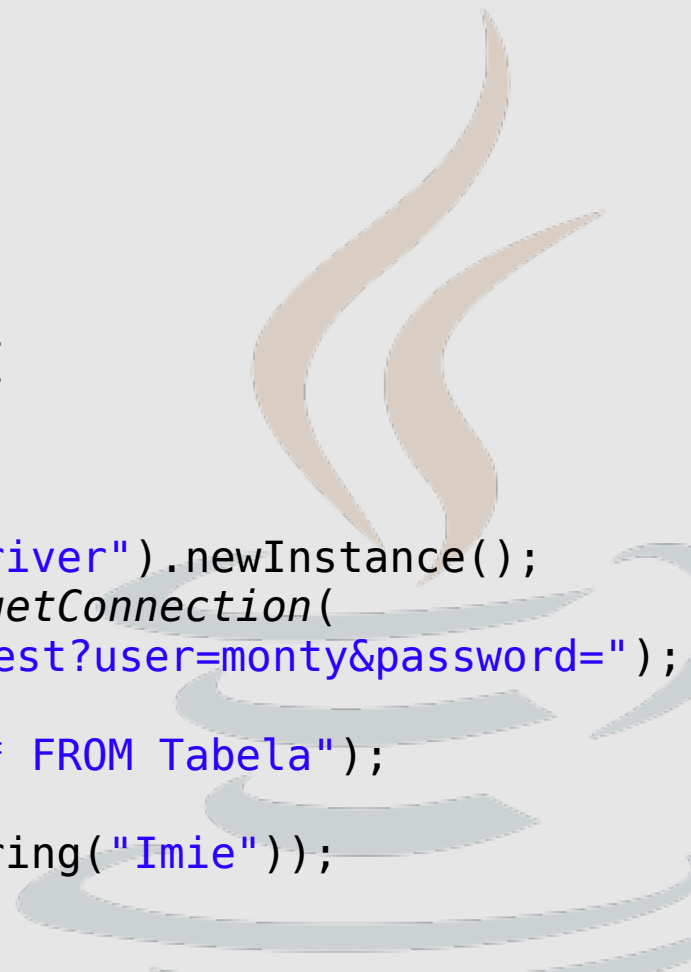
Java Database Connectivity (JDBC) to specyfikacja określająca zbiór klas i interfejsów napisanych w Javie, które mogą być wykorzystane przez programistów tworzących oprogramowanie korzystające z baz danych. Implementacja JDBC jest dostarczany przez producentów baz danych.

Jedną z ważniejszych zalet takiego rozwiązania jest ukrycie przed programistą kwestii technicznych dotyczących komunikacji z bazą danych. Dzięki temu ten sam program napisany w Javie może współpracować z różnymi systemami baz danych (np. Oracle, Sybase, IBM DB2). Wystarczy podmienić odpowiednie biblioteki implementujące JDBC.

PRZYKŁAD

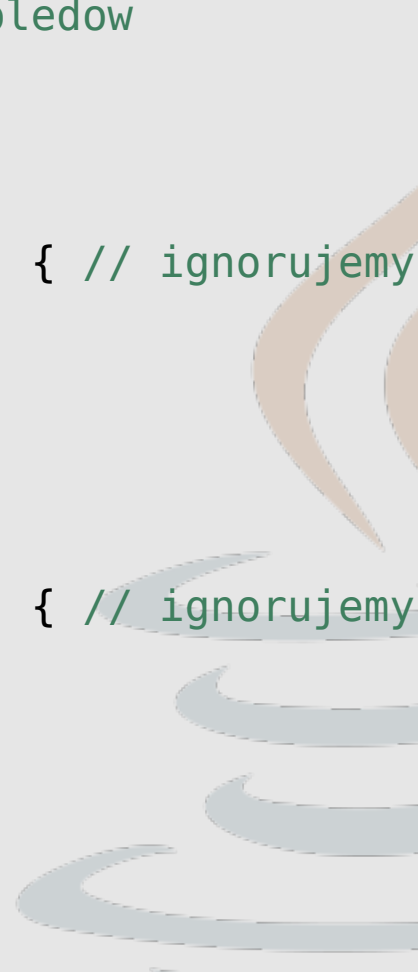
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class TestDriver {
    public static void main(String[] args) {
        Statement stmt = null;
        ResultSet rs = null;
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost/test?user=monty&password=");
            stmt = con.createStatement();
            rs = stmt.executeQuery("SELECT * FROM Tabela");
            while (rs.next()) {
                System.out.println(rs.getString("Imie"));
            }
        }
    }
}
```



PRZYKŁAD

```
catch (Exception ex) { // obsługa błędów
} finally { // zwalnianie zasobów
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException sqlEx) { // ignorujemy
        }
        rs = null;
    }
    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException sqlEx) { // ignorujemy
        }
        stmt = null;
    }
    // analogicznie con.close();
}
}
}
```



NAWIĄZANIE POŁĄCZENIA

Istnieją dwie metody nawiązania połączenia z bazą danych:

- za pomocą klasy **DriverManager**:

```
String url = "jdbc:odbc:bazadanych";
```

```
Connection con = DriverManager.getConnection(url, "login", "hasło");
```

- za pomocą klasy **DataSource** i usługi JNDI:

```
Context ctx = new InitialContext();
```

```
DataSource ds = (DataSource)ctx.lookup("jdbc/MojaDB");
```

```
Connection con = ds.getConnection("myLogin", "myPassword");
```

DriverManager

DriverManager jest tradycyjną warstwą zarządzającą JDBC pomiędzy użytkownikiem a sterownikiem. Aktualną listę dostępnych sterowników można uzyskać za pomocą metody: Enumeration

DriverManager.getDrivers().

Aby załadować dodatkowy sterownik należy skorzystać z metody:

Class.forName(String) podając jako argument klasę ze sterownikiem np:

```
Class.forName("jdbc.odbc.JdbcOdbcDriver"); // protokół jdbc:odbc
```

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

DataSource

DataSource reprezentuje źródło danych. Zawiera informacje identyfikujące i opisujące dane. Obiekt **DataSource** współpracuje z technologią *Java Naming and Directory Interface* (JNDI), jest tworzony i zarządzany niezależnie od używającej go aplikacji.

Korzystanie ze źródła danych zarejestrowanego w JNDI zapewnia:

- brak bezpośredniego odwołania do sterownika przez aplikację,
- umożliwia implementację grupowania połączeń (pooling) oraz rozproszonych transakcji.

Te cechy sprawiają, że korzystanie z klasy **DataSource** jest zalecaną metodą tworzenia połączenia z bazą danych, szczególnie w przypadku dużych aplikacji rozproszonych.

PRZESYŁANIE ZAPYTAŃ

Do przesyłania zapytań do bazy danych służą obiekty klasy:

- **Statement** – typowe pytania (bezparametrowe),
- **PreparedStatement** – prekompilowane pytania zawierające parametry wejściowe,
- **CallableStatement** – procedury zapisane w bazie danych.

Obiekt **Statement** tworzy się w ramach nawiązanego wcześniej połączenia:

```
Connection con = DriverManager.getConnection(url, login, pass);  
Statement stmt = con.createStatement();  
stmt.executeUpdate("INSERT INTO table(name, price) VALUE 'ser', 2.0");
```

PRZESYŁANIE ZAPYTAŃ

Do przesyłania zapytań służą metody:

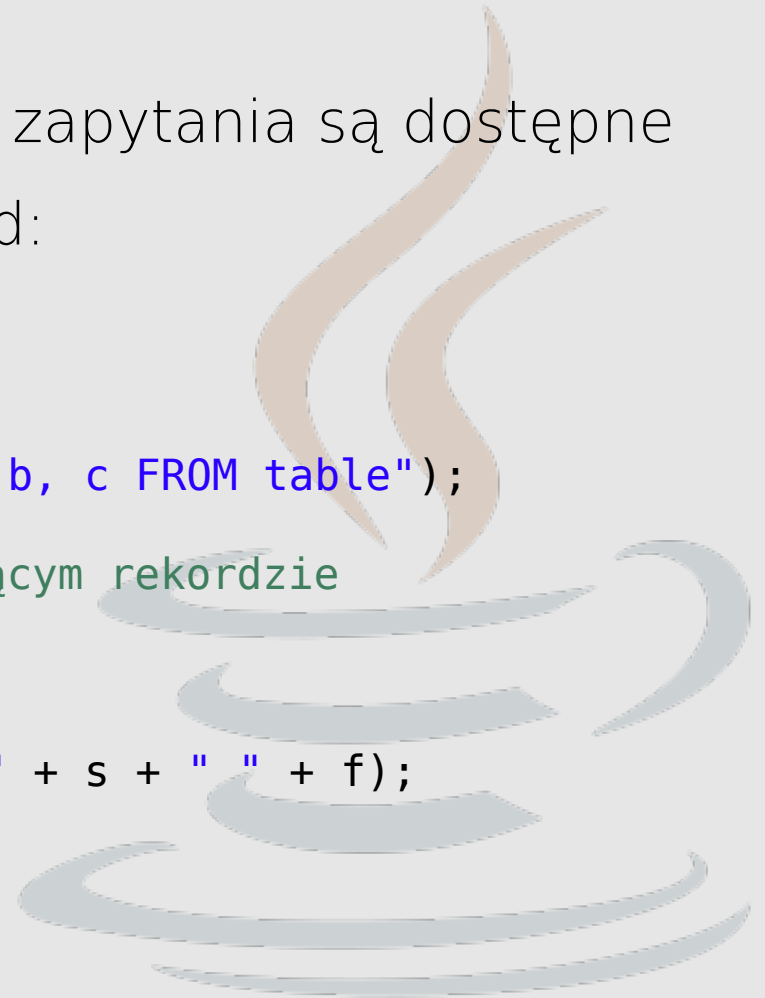
- **executeQuery** – pytania zwracające dane: SELECT,
- **executeUpdate** – pytania zmieniające dane: INSERT, UPDATE, CREATE TABLE,
- **execute** – dowolne zapytania. Dzięki tej instrukcji można wykonać sekwencje pytań, przekazać i odebrać dodatkowe parametry.

W ramach jednego obiektu Statement można wykonać sekwencyjnie kilka zapytań. Po zakończeniu używania obiektu zaleca się wywołanie metody **close()**.

ODBIERANIE WYNIKÓW

Wyniki zwrócone w wyniku wykonania zapytania są dostępne poprzez obiekt typu **ResultSet**. Przykład:

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM table");
while (rs.next()) {
    // odebranie i wypisanie wyników w bieżącym rekordzie
    int i = rs.getInt("a");
    String s = rs.getString("b");
    float f = rs.getFloat("c");
    System.out.println("Rekord = " + i + " " + s + " " + f);
}
```



HSQLDB

HSQLDB to system zarządzania relacyjnymi bazami danych w całości napisany w Javie (open source).

Strona domowa projektu: <http://www.hsqldb.org>.

Niektóre własności:

- obsługa SQL'a,
- obsługa transakcji (COMMIT, ROLLBACK, SAVEPOINT), zdalnych procedur i funkcji (mogą być pisane w Javie), triggerów itp.
- możliwość dołączania do programów i appletów. Działanie w trybie read-only,
- rozmiar tekstowych i binarnych danych ograniczony przez rozmiar pamięci.

HSQDLDB - SERWER

- HyperSQL Server – tryb preferowany. Klasa `org.hsqldb.server.Server`,
- HyperSQL WebServer – używany, jeśli serwer może używać tylko protokołu HTTP lub HTTPS. W przeciwnym razie niezalecany. Klasa `org.hsqldb.server.WebServer`,
- HyperSQL Servlet – używa tego samego protokołu co WebServer. Wymaga osobnego kontenera serwletów (np. Tomcat).
- *Stand-alone* – “serwer” uruchamiany w ramach procesu, w którym nawiązujemy połączenie z bazą.

Wszystkie tryby pracy serwera umożliwiają korzystanie z JDBC.

HSQLDB - POŁĄCZENIE

```
try {
    Class.forName("org.hsqldb.jdbc.JDBCDriver" );
} catch (Exception e) {
    System.err.println("ERROR: failed to load HSQLDB JDBC driver.");
    e.printStackTrace();
    return;
}

Connection c =
DriverManager.getConnection("jdbc:hsqldb:sql://localhost/xdb", "SA", "");

Connection c =
DriverManager.getConnection("jdbc:hsqldb:http://localhost/xdb", "SA", "");

Connection c =
DriverManager.getConnection("jdbc:hsqldb:file:/opt/db/testdb;shutdown=true", "SA", "");
```



HSQLDB *stand-alone*

W trybie *stand-alone* „serwer” bazy danych działa w ramach tej samej wirtualnej maszyny Javy, co korzystający z niego program „kliencki”.

Przykład uruchomienia bazy:

```
Connection c = DriverManager.getConnection(  
    "jdbc:hsqldb:file:/opt/db/testdb", "sa", "");
```

Niewielkie bazy danych mogą być uruchamiane do pracy w pamięci operacyjnej komputera:

```
Connection c = DriverManager.getConnection(  
    "jdbc:hsqldb:mem:testdb", "sa", "");
```

W obecnej wersji HSQLDB istnieje możliwość jednoczesnego używania wielu „serwerów” baz danych działających w trybie *stand-alone*.

PODSUMOWANIE

Podstawowe klasy służące do interakcji z systemami bazodanowymi są udostępniane za pośrednictwem obiektów: **Connection**, **Statement** i **ResultSet**. Wszystkie te klasy należą do pakietu **java.sql**. Klasy rozszerzające funkcjonalność JDBC (np. **DataSource**) znajdują się w pakiecie **javax.sql**.

Korzystanie z interfejsu JDBC umożliwia jednolity sposób dostępu do różnych systemów bazodanowych. Jako przykład przedstawiono bazę HSQLDB. Jej ważną zaletą jest możliwość działania w ramach jednej Wirtualnej Maszyny Javy wraz z programem klienckim.

DZIĘKUJĘ ZA UWAGĘ