Bazy danych programowanie (JDBC, Hibernate)

JDBC

JDBC (ang. Java DataBase Connectivity - łącze do baz danych w języku Java) – interfejs programowania opracowany w 1996 r. przez Sun Microsystems, umożliwiający niezależnym od platformy aplikacjom napisanym w języku Java porozumiewanie się z bazami danych za pomocą języka SQL.

JDBC

API standaryzuje:

- sposób nawiązywania połączenia z bazą danych
- podejście do inicjowania zapytań
- metodę tworzenia sparametryzowanych zapytań

JDBC

API standaryzuje:

- strukturę danych wyniku zapytania (tabela)
- określanie liczby kolumn
- wyszukiwanie metadanych itp.

- API nie standaryzuje składni SQL
- JDBC nie jest osadzonym SQL
- klasy JDBC znajdują się w pakiecie java.sql

JDBC

Interfejs API JDBC składa się z czterech podstawowych części:

- JDBC Driver
- Connection
- Statement
- Result Set

JDBC

Główne funkcjonalności JDBC:

- zapytania do bazy danych (odczyt danych z niej)
- zapytanie o metadane bazy danych
- aktualizacja danych.
- wspieranie transakcji

JDBC API

- **java.sql.DriverManager** - obsługuje ładowanie sterowników i zapewnia obsługę tworzenia nowych połączeń z bazą danych

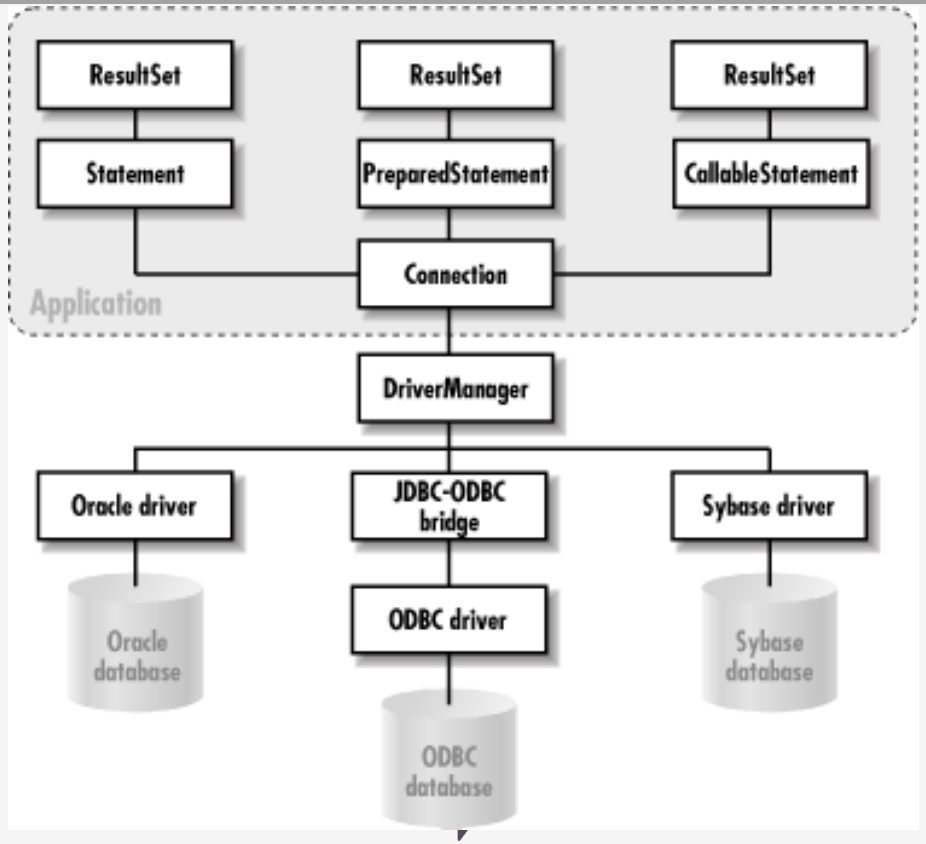
- **java.sql.Connection** - reprezentuje połączenie określoną bazą danych

JDBC API

- **java.sql.Statement -** działa jako kontener do wykonywania instrukcji SQL w danym połączeniu

- **java.sql.ResultSet -** kontroluje dostęp do wyników wiersza danego zapytania (Statement)

JDBC API


ResultSet
ResultSet
ResultSet
Statement
PreparedStatement
CallableStatement
Connection
Application
DriverManager
Oracle driver
JDBC-ODBC bridge
Sybase driver
Oracle database
ODBC driver
Sybase database
ODBC database

JDBC API

1. Connect
2. Query
3. Process results
4. Close

JDBC API - Connect

➔ rejestracja sterownika

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

JDBC API - Connect

➔ utworzenie połączenia

```
Connection conn =
   DriverManager.getConnection(
   "jdbc:mysql://localhost:3306/hr?" +
   "user=pmazur&password=pmazur");
```

JDBC API - Query

➔ przygotowanie zapytania

Obiekt *Statement* wysyła zapytanie SQL do DBMS. Stwórz obiekt *Statement*, a następnie go uruchom, wywołując odpowiednią metodę execute dla instrukcji SQL, którą chcesz wysłać. Dla instrukcji *SELECT*, metoda do użycia to *executeQuery*. W przypadku instrukcji, które tworzą lub modyfikują tabele, należy użyć metody *executeUpdate*.

JDBC API - Query

Obiekt *Statement* wysyła zapytanie SQL do DBMS. Stwórz obiekt *Statement*, a następnie go uruchom, wywołując odpowiednią metodę execute dla instrukcji SQL, którą chcesz wysłać. Dla instrukcji *SELECT*, metoda do użycia to *executeQuery*. W przypadku instrukcji, które tworzą lub modyfikują tabele, należy użyć metody *executeUpdate*.

JDBC API - Query

Aby utworzyć obiekt *Statement* używamy aktywnego połączenia. W poniższym przykładzie używamy naszego obiektu *Connection* do tworzenia obiektu *Statement*:

Statement s = conn.createStatement ();

JDBC API - Query

W tym momencie *statement* istnieje, ale nie ma instrukcji SQL do przekazania do DBMS.
Musimy dostarczyć to do metody, której używamy do wykonania s. Na przykład w poniższym fragmencie kodu dostarczamy *executeQuery* z instrukcją SQL z powyższego przykładu:

```
ResultSet resultSet =
    s.executeQuery ("SELECT * FROM employees");
```

JDBC API - Result

JDBC zwraca wyniki w obiekcie *ResultSet*, więc musimy zadeklarować instancję klasy *ResultSet*, aby zatrzymać nasze wyniki.

*String query = "SELECT * FROM departments";*

ResultSet ResultSet = s.executeQuery(query);

JDBC API - Close

➔ zamykanie połączenia

resultSet.close();
statement.close();
connection.close();