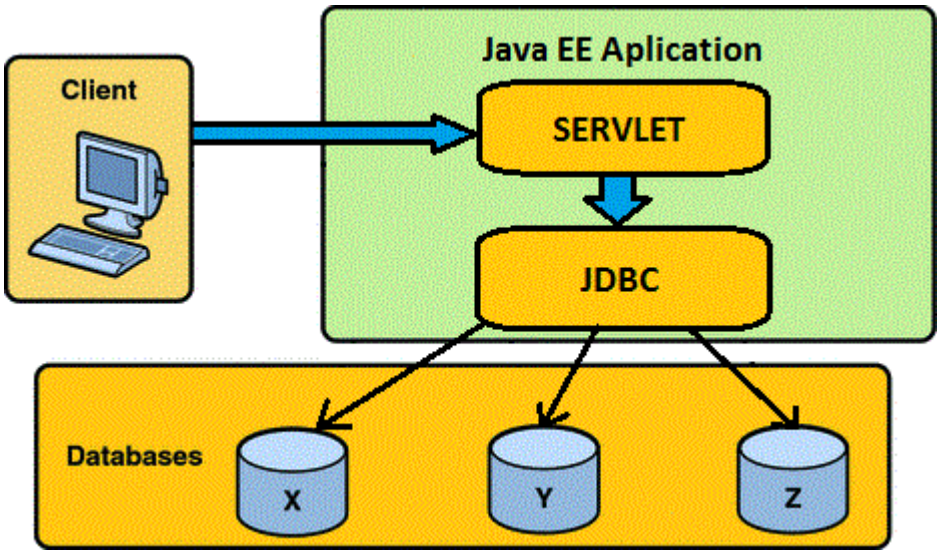
1. Sterownik JBDC

JDBC, czyli Java DataBase Conectivity - jest to interfejs pozwalający na ustanowienie połączenia do bazy danych z poziomu aplikacji napisanej w Javie. Biblioteki, których potrzebujemy, zawierają pakiety java.sql.* i są w taki sam sposób dostępny jak kolekcje w pakietach java.util.* .



Klientami mogą być : przeglądarki internetowe, telefony komórkowe, inne aplikacje webowe, telewizory a nawet routery (jak się je odpowiednio przerobi).

Servlet - komponent odpowiedzialny za komunikacje z klientem za pomocą protokołu http.

Bazami danych mogą być : serwer MySQL, SQLite, Oracle Database bądź inna relacyjna baza danych. Dla naszych potrzeb wystarczy MySQL, ze względu na prostotę instalacji i konfiguracji.

JDBC może pracować z różnymi bazami danych. Wystarczy podać odpowiedni sterownik obługujący dany typ bazy danych – dla MySQL będzie to sterownik **" com.mysql.cj.jdbc.Driver".**

2. Omawiana baza danych

3. Wyjaśnienie operacji z java.sql

1. Class.forName(Biblioteka.DRIVER); ***//załadowanie sterownika do systemu***

2. conn = DriverManager.getConnection(DB_URL "root", ""); ***//tworzenie połączenia z bazą danych***

   DB_URL – ***adres URL do bazy danych „biblioteka" znajdującej się na naszym serwerze (localhost)***

```
public static final String DRIVER = "com.mysql.cj.jdbc.Driver";

public static final String DB_URL
="jdbc:mysql://localhost/biblioteka?useUnicode=true&useJDBCCompliantTimezon
eShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";
```

3. Connection – połączenie z BD jest reprezentowane przez obiekt *Connection* (ustanowienie połączenia, zamknięcie połączenie etc.)

   stat = conn.createStatement(); ***//po ustanowienia połączenia należy utworzyć nowy obiekt klasy Statement (w tym przypadku należy traktować createStatement jako konstruktor)***

   //należy postąpić analogicznie, wtedy gdy chcemy utworzyć np. PreparedStatement

   PreparedStatement prepStmt = conn.prepareStatement("insert into czytelnicy values (NULL, ?, ?, ?);");

```
prepStmt.setString(1, tytul);
prepStmt.setString(2, autor);
prepStmt.execute();
```

//1 odnosi się do 1 znaku zapytania
//2 odnosi się do 2 znaku zapytania
//execute() – wykonania zapytania

4. `Statement – interfejs reprezentujący zapytanie SQL`

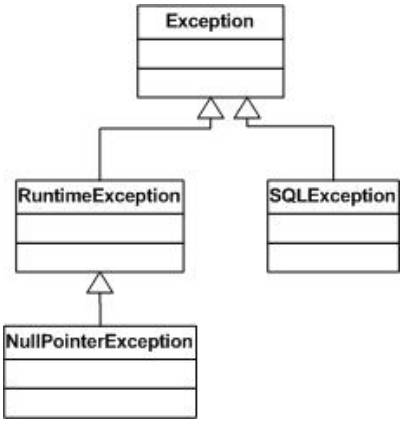
Typy Statement:

- Statement – zapytanie SQL bez parametrów
- PreparedStatement - zapytanie SQL mogące posiadać parametry
- CallableStatement – używane do wykonywania procedur, które mogą zawierać zarówno parametry wejściowe oraz wyjściowe

5. ResultSet
   "You execute `Statement` objects, and they generate `ResultSet` objects, which is a table of data representing a database ***result set****"*

   ResultSet result = stat.executeQuery("SELECT * FROM czytelnicy");

4. SQLException




```
public class SQLException
extends Exception
implements Iterable<Throwable>
```

An exception that provides information on a database access error or other errors.

Each `SQLException` provides several kinds of information:

- a string describing the error. This is used as the Java Exception message, available via the method `getMesasge`.
- a "SQLstate" string, which follows either the XOPEN SQLstate conventions or the SQL:2003 conventions. The values of the SQLState string are described in the appropriate spec. The `DatabaseMetaData` method `getSQLStateType` can be used to discover whether the driver returns the XOPEN type or the SQL:2003 type.
- an integer error code that is specific to each vendor. Normally this will be the actual error code returned by the underlying database.
- a chain to a next Exception. This can be used to provide additional error information.
- the causal relationship, if any for this `SQLException`.

`e.printStackTrace();` ***//dokładniejsze informacje o wyjątku – (co się stało i w którym miejscu w kodzie)***