# PAO - Laborator 11

*2019-05-13*

# Content

- Sockets
  - UDP
  - TCP
- JDBC
  - MySQL, H2 …
  - Connection
  - Statement, PrepareStatement
  - ResultSet, ResultSetMetaData

# Sockets

## What Is a Socket?

A *socket* is an abstraction through which an application may send and receive data, in much the same way as an open file handle allows an application to read and write data to stable storage. A socket allows an application to plug in to the network and communicate with other applications that are plugged in to the same network. Information written to the socket by an application on one machine can be read by an application on a different machine and vice versa.

# Sockets

- OSI

| | |
|---|---|
| Layer 7 | Application |
| Layer 6 | Presentation |
| Layer 5 | Session |
| Layer 4 | Transport |
| Layer 3 | Network |
| Layer 2 | Data Link |
| Layer 1 | Physical |

# Sockets

- TCP/IP

| | |
|---|---|
| Layer 5 | Application |
| Layer 4 | Transport |
| Layer 3 | Network |
| Layer 2 | Data Link |
| Layer 1 | Physical |

# Sockets

- TCP/IP

# TCP protocol

# UDP protocol

**UDP Header**

| *Offsets* | Octet | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | 0 | Source port | | | | | | | | | | | | | | | | Destination port | | | | | | | | | | | | | | | |
| 4 | 32 | Length | | | | | | | | | | | | | | | | Checksum | | | | | | | | | | | | | | | |

- Arrow in a wind

# UDP and TCP message

- Create a server socket and bind it to a specific port number
- Listen for a connection from the client and accept it. This results in a client socket is created for the connection.
- Read data from the client via an InputStream obtained from the client socket.
- Send data to the client via the client socket's OutputStream.
- Close the connection with the client.

# UDP and TCP message

```
//          4. Send data to the client via the client socket's
OutputStream.
SimpleDateFormat format =
    new SimpleDateFormat("EEE, d MMM yyyy HH:mm:Ss z");


String msg = "<html><body>OK message</body></html>\n\n";
String response = "HTTP/1.0 200 OK\n"
        + "Server: HTTP server/0.1\n"
        + "Date: "+format.format(new Date())+"\n"
        + "Content-type: text/html; charset=UTF-8\n"
        + "Content-Length: " + msg.length() + "\n\n"
        + msg;
```

# JDBC

A relational database is a database that is organised into tables, which consist of rows and columns.

There are two main ways to access a relational database from Java:

- Java Database Connectivity Language (JDBC): Accesses data as rows and columns.
- Java Persistence API (JPA): Accesses data through Java objects using a concept called object-relational mapping (ORM). The idea is that you don't have to write as much code, and you get your data in Java objects.
- A relational database is accessed through Structured Query Language (SQL).
- In addition to relational databases, there is another type of database called a NoSQL database. This is for databases that store their data in a format other than tables.

# JDBC

JDBC interfaces

- **Driver**: Knows how to get a connection to the database
- **Connection**: Knows how to communicate with the database
- **Statement**: Knows how to run the SQL
- **ResultSet**: Knows what was returned by a SELECT query

All database classes are in the package java.sql.

# JDBC

- ResultSet types

By default, a ResultSet is in **TYPE_FORWARD_ONLY** mode. It allows you to go through the data once in the order in which it was retrieved.

Two other modes that you can request when creating a Statement are:

TYPE_SCROLL_INSENSITIVE and TYPE_SCROLL_SENSITIVE.

With **TYPE_SCROLL_INSENSITIVE**, you have a static view of what the ResultSet looked like when you did the query. If the data changed in the table, you will see it as it was when you did the query.

With **TYPE_SCROLL_SENSITIVE**, you would see the latest data when scrolling through the ResultSet.

# JDBC

- ResultSet types

| ResultSet Type | Can Go Backward | See latest Data from Database Table | Supported by Most Drivers |
|---|---|---|---|
| TYPE_FORWARD_ONLY | No | No | Yes |
| TYPE_SCROLL_INSENSITIVE | Yes | No | Yes |
| TYPE_SCROLL_SENSITIVE | Yes | Yes | No |

# JDBC

- ResultSet concurrent mode

| ResultSet Type | Can Read Data | Can Update Data | Supported by All Drivers |
|---|---|---|---|
| CONCUR_READ_ONLY | Yes | Yes | No |
| CONCUR_UPDATABLE | Yes | No | Yes |

# JDBC

- Statement execution

| Method | Return Type | What is Returned for SELECT | What is returned for DELETE/ INSERT/ UPDATE |
|---|---|---|---|
| stmt.execute() | boolean | TRUE | FALSE |
| stmt.executeQuery() | ResultSet | The rows and columns returned | n/a |
| stmt.executeUpdate() | int | n/a | Number of rows affected |

# JDBC

- Statement get data, primitives

| Method Name | Return Type | Example Database Type |
|---|---|---|
| getBoolean | boolean | BOOLEAN |
| getDate | java.sql.Date | DATE |
| getDouble | double | DOUBLE |
| getInt | int | INTEGER |
| getLong | long | BIGINT |
| getObject | Object | Any type |
| getString | String | CHAR, VARCHAR |
| getTime | java.sql.Time | TIME |
| getTimeStamp | java.sql.TimeStamp | TIMESTAMP |

# JDBC

- Date, time behaviour

| JDBC Type | Java 8 Type | Contains |
|---|---|---|
| java.sql.Date | java.time.LocalDate | Date only |
| java.sql.Time | java.time.LocalTime | Time only |
| java.sqlTimeStamp | java.time.LocalDateTime | Both date and time |

- LocalDate d = resultSet.getDate(i).toLocalDate();

# JDBC

- ResultSet positioning

| Method | Description | Requires Scrollable `ResultSet` |
|---|---|---|
| `boolean absolute(int rowNum)` | Move cursor to the specified row number | Yes |
| `void afterLast()` | Move cursor to a location immediately after the last row | Yes |
| `void beforeFirst()` | Move cursor to a location immediately before the first row | Yes |
| `boolean first()` | Move cursor to the first row | Yes |
| `boolean last()` | Move cursor to the last row | Yes |
| `boolean next()` | Move cursor one row forward | No |
| `boolean previous()` | Move cursor one row backward | Yes |
| `boolean relative(int rowNum)` | Move cursor forward or backward the specified number of rows | Yes |

# JDBC

- SQL exception

```
System.out.println("state: " + e.getSQLState());
System.out.println("error code: " + e.getErrorCode());
System.out.println("next exception: " + e.getNextException());
```

# JDBC

- Transaction
- connection.setSavepoint
- connection.releaseSavepoint
- connection.rollback
- connection.setAutoCommit - true sau false

# JDBC

- DataSource

Pool of connections

```
MysqlDataSource dataSource = new MysqlDataSource();
dataSource.setDatabaseName("store");
dataSource.setUser("root");
dataSource.setPassword("fmi2019");
dataSource.setServerName("localhost");
dataSource.setServerTimezone("UTC");
```

# JDBC

- XA drivers - distributed db

# JDBC

- H2
- http://www.h2database.com/html/download.html

```
import java.sql.*;

public class Test {

    public static void main(String[] a)

        throws Exception {

      Connection conn = DriverManager.

        getConnection("jdbc:h2:~/test", "sa", "");

      // add application code here

      conn.close();

    }

}
```

# TASKS - 1

- Create a server tcp multi client. Use an ExecutorService
- The response can have task name a date as html content.

# TASKS - 2

- Update the services from homework project to save data on a db. Implement at minimum 4 entities with a CRUD.

# TASKS - 3 -homework

- Download tomcat, any version from 7 or after.
- https://tomcat.apache.org/download-90.cgi
- Netbeans, https://netbeans.apache.org

# TASKS - 3 - homework

- Create an empty project like below