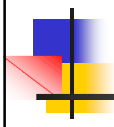


# Programowanie w języku Java

Wykład 4: Programowanie rozproszone: TCP/IP, URL.

# Programowanie sieciowe w Javie

- proste programowanie sieciowe (java.net)
  - na poziomie UDP
  - na poziomie IP
  - na poziomie URL
- JDBC (java.sql)
  - obsługa baz danych
- RMI
  - zdalne wywoływanie metod
- CORBA
  - komunikacja z aplikacjami napisanymi w innych językach

# Komunikacja na poziomie UDP

```
socket = new DatagramSocket(4445);

byte[ ] buf = new byte[256];
DatagramPacket packet = new DatagramPacket(buf, buf.length);
socket.receive(packet);
```

```
DatagramSocket socket = new DatagramSocket();
byte[] buf = new byte[256];
InetAddress address = InetAddress.getByName("adres.pl");
DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 4445);
socket.send(packet);
```



# Programowanie na poziomie IP

```
InetAddress addr = InetAddress.getByName(null);
InetAddress addr = InetAddress.getByName("127.0.0.1");
InetAddress addr = InetAddress.getByName("localhost");
```

```
Serwer:
ServerSocket s = new
                 ServerSocket(PORT);
Socket socket = s.accept();
BufferedReader in =
  new BufferedReader(
   new InputStreamReader(
     socket.getInputStream()));
PrintWriter out =
 new PrintWriter(
  new BufferedWriter(
   new OutputStreamWriter(
    socket.getOutputStream())),
                   true);
...
socket.close();
s.close();
```

```
Klient:
Socket socket = new Socket(addr,PORT);
BufferedReader in =
 new BufferedReader(
  new InputStreamReader(
    socket.getInputStream()));
PrintWriter out =
 new PrintWriter(
   new BufferedWriter(
     new OutputStreamWriter(
     socket.getOutputStream())),true);
....
socket.close();
```

# Serwer wielowątkowy (1)

```
import java.io.*;
import java.net.*;

class ServeOneJabber extends Thread {
  private Socket socket;
  private BufferedReader in;
  private PrintWriter out;
  public ServeOneJabber(Socket s)
      throws IOException {
    socket = s;
    in =  new BufferedReader(
        new InputStreamReader(
          socket.getInputStream()));
   out =   new PrintWriter(
        new BufferedWriter(
          new OutputStreamWriter(
            socket.getOutputStream())), true);
    start(); // Calls run()
  }
```

```
public void run() {
    try {
      while (true) {
        String str = in.readLine();
        if (str.equals("END")) break;
        System.out.println("Echoing: " + str);
        out.println(str);
      }
      System.out.println("closing...");
    } catch(IOException e) {
      System.err.println("IO Exception");
    } finally {
      try {
        socket.close();
      } catch(IOException e) {
        System.err.println("Socket not closed");
      }
    }
  }
}
```



# Serwer wielowątkowy (2)

```
public class MultiJabberServer {
  static final int PORT = 8080;
  public static void main(String[] args)
      throws IOException {
    ServerSocket s = new ServerSocket(PORT);
    System.out.println("Server Started");
    try {
      while(true) {
        Socket socket = s.accept();
        try {
          new ServeOneJabber(socket);
        } catch(IOException e) {
            socket.close();
        }
      }
    } finally {
      s.close();
    }
  }
}
```

# Komunikacja SSL

```
import java.io.*;
import javax.net.ssl.*;
. . .
int port = availablePortNumber;
SSLServerSocket s;
try {
   SSLServerSocketFactory sslSrvFact =
      (SSLServerSocketFactory)
     SSLServerSocketFactory.getDefault();
  s=(SSLServerSocket)sslSrvFact.
            createServerSocket(port);
  SSLSocket c = (SSLSocket)s.accept();
  OutputStream out = c.getOutputStream();
  InputStream in = c.getInputStream();
   // Send messages to the client through
   // the OutputStream
   // Receive messages from the client
   // through the InputStream
}

catch (IOException e) {}
```

```
import java.io.*;
import javax.net.ssl.*;
. . .
int port = availablePortNumber;
String host = "hostname";
try {
   SSLSocketFactory sslFact =  (SSLSocketFactory)
                SSLSocketFactory.getDefault();
   SSLSocket s = (SSLSocket)
                sslFact.createSocket(host, port);
   OutputStream out = s.getOutputStream();
   InputStream in = s.getInputStream();
    // Send messages to the server through
    // the OutputStream
    // Receive messages from the server
    // through the InputStream
}
catch (IOException e) {
}
```



# Programowanie na poziomie URL (1)

```
URL u = new URL(getDocumentBase(),"strona.html");
....
getAppletContext().showDocument(u);
```

```
URL(String protocol, String host, String file)

URL(String spec)
```

```
try {
 URL myURL = new URL(. . .) ;
}
 catch (MalformedURLException e) {
 . . . // exception handler code here
. . .
}
```

# Przykład 1

```
import java.net.*;
import java.io.*;

public class URLReader {
    public static void main(String[] args) throws Exception {
            URL yahoo = new URL("http://www.pk.edu.pl");
            BufferedReader in = new BufferedReader(
                                        new InputStreamReader(
                                        yahoo.openStream()));

            String inputLine;

            while ((inputLine = in.readLine()) != null)
                System.out.println(inputLine);

            in.close();
    }
}
```



# Przykład 2

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
public class ShowHTML extends JApplet {
  JButton send = new JButton("Go");
  JLabel l = new JLabel();
  public void init() {
    Container cp = getContentPane();   cp.setLayout(new FlowLayout());
    send.addActionListener(new Al());    cp.add(send);   cp.add(l);
  }
  class Al implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
      try {
        URL u = new URL(getDocumentBase(),  "FetcherFrame.html");
        getAppletContext().showDocument(u);
      } catch(Exception e) {   l.setText(e.toString());   }
    }
  }
}
```

# Nawiązanie połączenia URL

```
URL pk = new URL("http://www.pk.edu.pl/");
URLConnection pkConnection = pk.openConnection();
pkConnection.connect();
```

```
BufferedReader in = new BufferedReader( new InputStreamReader(
                                    pkConnection.getInputStream()));
```

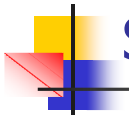
```
connection.setDoOutput(true);
OutputStreamWriter out = new OutputStreamWriter(
                                connection.getOutputStream());

String string = URLEncoder.encode("Text do wysłania", "UTF-8");
```



# JDBC- Java DataBase Connectivity

- Pakiet: java.sql
- Sterownik bazy danych
  np. dla bazy MySql: „com.mysql.jdbc.Driver"
- Adres URL bazy danych:
  jdbc:podprotokół:adres_bazy



# Schemat obsługi bazy danych:

1. Odnalezienie sterownika JDBC:
   **Class.forName( "com.mysql.jdbc.Driver ");**
2. Konfigurowanie bazy danych
3. Nawiązanie połączenia z bazą danych:
   **DriverManager.getConnection(dbUrl, user, password);**
4. Wysyłanie zapytań SQL:
   **createStatement(), executeQuery()**
5. Zamkniecie połączenia z bazą
   **close()**



# Przykład

```
import java.sql.*;

public class Lookup {
  public static void main(String[] args)
  throws SQLException, ClassNotFoundException {
    String dbUrl = "jdbc:odbc:people";
    String user = "";
    String password = "";
    Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver");
    Connection c = DriverManager.getConnection(dbUrl, user, password);
    Statement s = c.createStatement();
    ResultSet r = s.executeQuery("SELECT FIRST, LAST, EMAIL " +
       "FROM people.csv people " + "WHERE " + "(LAST='" + args[0] + "') " +
       " AND (EMAIL Is Not Null) " + "ORDER BY FIRST");
    while(r.next()) {
      System.out.println( r.getString("Last") + ", " + r.getString("fIRST")
       + ": " + r.getString("EMAIL") );
    }
    s.close();
  }
}
```

# Koniec