

Praktikum 10

PEMROGRAMAN SOCKET

A. TUJUAN PEMBELAJARAN

1. Siswa memahami konsep aplikasi client server di jaringan.
2. Mahasiswa memahami konsep pemrograman socket dasar.
3. Mahasiswa mampu membangun program socket sederhana dg single thread

B. DASAR TEORI

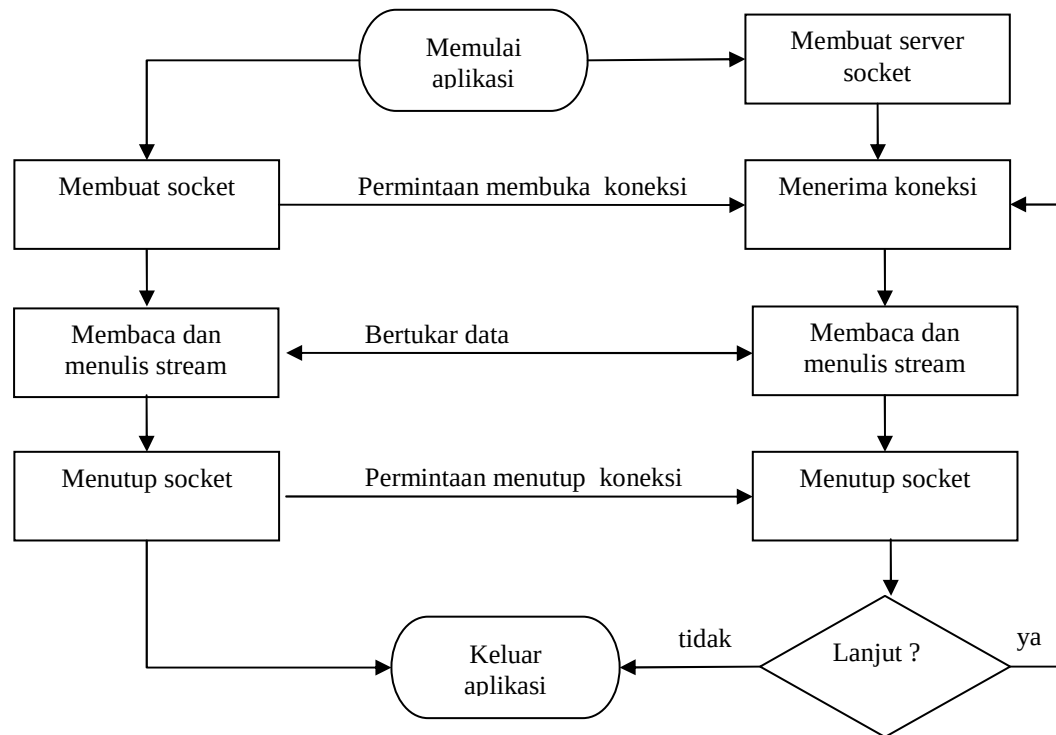
Model yang umum diterapkan dalam jaringan komputer adalah model client/server.

Konsepnya sederhana, sebuah aplikasi client melakukan permintaan untuk suatu layanan (service) informasi atau mengirim sebuah perintah ke suatu aplikasi server. Aplikasi server akan menerima permintaan dari client, kemudian memproses berdasarkan permintaan tersebut. Dari hasil pemrosesan yang sudah dilakukan, aplikasi server akan mengembalikan hasil pemrosesan tersebut ke aplikasi client.

Sehingga pada prinsipnya aplikasi server dalam status menunggu (*listen*) permintaan dari aplikasi client, sedangkan client mencoba membuat koneksi (*connect*) ke server.

Aplikasi client melakukan koneksi ke server melalui sebuah alamat *socket*. Alamat socket adalah kombinasi dari alamat IP dan nomor port. Contoh alamat socket adalah 192.168.1.30:80, dimana nomer 80 adalah nomer portnya. Jika alamat IP diibaratkan sebuah nomer telepon, maka nomer port adalah nomer ekstensinya.

Suatu proses yang hendak berkomunikasi dengan proses lain lewat mekanisme socket haruslah mengikatkan dirinya dengan salah satu port. Tahap proses ini disebut dengan *binding*.



Gambar 1

Gambar 10.1 menunjukkan bagan interkoneksi antara aplikasi server dan aplikasi client.

Model pemrograman client-server menggunakan dua macam koneksi pada layer 4/ layer transport OSI, yaitu connection oriented dan connectionless oriented. Untuk connection oriented menggunakan protokol TCP (Transmission Control Protocol). Sedangkan connectionless oriented menggunakan UDP (User Datagram Protocol).

Pada praktikum ini, kita akan mencoba membangun sebuah aplikasi client-server sederhana dengan menggunakan bahasa pemrograman Java. Pada J2SE telah disediakan paket java.net yang berisi kelas-kelas dan interface yang menyediakan API (Application Programming Interface) level rendah (Socket, ServerSocket, DatagramSocket) dan level tinggi (URL, URLConnection). Berikut adalah contoh dari kelas Socket berikut method

dan eksepsi error yang tersedia (Untuk lebih jelasnya anda bias membacanya di Java documentation) :

- `Socket(String host, int port, InetAddress localAddr, int localPort);`
 - membuat sebuah socket dan mengkoneksikannya ke port yang dituju pada alamat IP yang disebutkan pada parameter address atau nama host. Selain itu juga akan dilakukan bind socket ke alamat lokal dan port lokal. (Hal ini dilakukan jika koneksi antara client dan server membutuhkan nomor port yang sudah ditentukan.
- **Methods :**
 - `getInetAddress()` : untuk mendapatkan nama host yang dituju dan alamat IPnya
 - `getPort()` : untuk mendapatkan nomor remote host
 - `getLocalPort()`: untuk mendapatkan nomor port localhost
 - `getLocalAddress()`: untuk mendapatkan alamat local dimana socket digunakan
 - `getInputStream()`: mengembalikan objek input stream dari socket
 - `getOutputStream()`: mengembalikan objek output stream ke socket
 - `setSoTimeout(int timeout)` dan `getSoTimeout()` : Kedua method tersebut digunakan untuk memberi (set) dan mengambil (get) nilai opsi Socket untuk time out block (dalam milidetik) reading dari socket (`SO_TIMEOUT`). Jika dalam waktu timeout tidak mendapat suatu nilai maka, akan dilemparkan ke exception `java.net.SocketTimeoutException`. Nilai default timeoutnya adalah 0, yang berarti tanpa batas.
- **Exceptions :**
 - `SocketException` : Kelas ini merupakan kelas yang diturunkan dari kelas `IOException`. Kelas exception ini dipanggil atau dipicu ketika ada kegagalan dalam pemakaian socket, sebagai contoh adalah kegagalan dalam protokol TCP. Salah satu penyebabnya yang mungkin terjadi adalah ketika port yang akan digunakan sudah digunakan sebelumnya pada localhost. Penyebab yang lain adalah user tidak dapat melakukan bind ke port yang dituju. Misalnya saja, Anda ingin menggunakan port

80 untuk aplikasi Anda, namun ternyata pada komputer Anda tersebut sudah berjalan HTTP Server yang juga menggunakan port 80. Bila hal ini terjadi, maka JVM akan melemparkan kegagalan yang ada ke kelas exception `SocketException`.

- `BindException` : Exception ini akan dipanggil ketika ada port lokal yang akan digunakan sudah terpakai oleh yang lain, atau ada kegagalan dalam permintaan untuk menggunakan alamat.
- `ConnectException` : Exception ini akan dipanggil ketika sebuah koneksi ditolak oleh host yang dituju, oleh karena tidak ada proses yang siap menerima data pada port yang dituju.
- `NoRouteToHostException`: Koneksi yang akan dibangun tidak dapat dipenuhi oleh karena melebihi waktu timeout yang tersedia atau host yang dituju tidak dapat dicapai (unreachable).

C. Contoh Program

C.1. Program Aplikasi HTTP Server sederhana

```
1. /**
2.  * @author ferry
3.  */
4. ///A Simple Web Server (WebServer.java)
5.
6. import java.io.BufferedReader;
7. import java.io.InputStreamReader;
8. import java.io.PrintWriter;
9. import java.net.ServerSocket;
10. import java.net.Socket;
11.
12. public class webServer {
13.     /**
14.      * WebServer constructor.
15.     */
16.     protected void start() {
17.         ServerSocket s;
18.         int port = 8888;
19.         System.out.println("Webserver starting up on port " +port);
20.         System.out.println("(press ctrl-c to exit)");
21.         try {
22.             // tahap bind(),membuat socket
```

```
23.     s = new ServerSocket(port);
24. } catch (Exception e) {
25.     System.out.println("Error: " + e);
26.     return;
27. }
28.
29. System.out.println("Waiting for connection");
30. for (;;) {
31.     try {
32.         // tahap listen(), menunggu koneksi
33.         Socket remote = s.accept();
34.         // tahap accept()
35.         System.out.println("Connection, sending data.");
36.         BufferedReader in = new BufferedReader(new InputStreamReader(
37.             remote.getInputStream()));
38.         PrintWriter out = new PrintWriter(remote.getOutputStream());
39.         // membaca request
40.         String str = ".";
41.         while (!str.equals(""))
42.             str = in.readLine();
43.         // Mengirim response dan mengirim HTTP headers
44.         out.println("HTTP/1.0 200 OK");
45.         out.println("Content-Type: text/html");
46.         out.println("Server: Bot");
47.         // Batas pengiriman header ditandai dengan baris kosong
48.         out.println("");
49.         // Mengirim halaman HTML
50.         out.println("<H1>Selamat datang di web server percobaan</H1>");
51.         out.println("<blink>Selamat datang di web server percobaan</blink>");
52.         out.flush();
53.         remote.close();
54.     } catch (Exception e) {
55.         System.out.println("Error: " + e);
56.     }
57. }
58. }
59.
60. /**
61.  * Menjalankan server
62.  *
63.  * @param args
64.  *      Command line parameters are not used.
65.  */
66. public static void main(String args[]) {
67.     webServer ws = new webServer();
68.     ws.start();
```

```
69. }  
70. }
```

```
END
```

C.2. Program Aplikasi Chat Server dengan menggunakan TCP

```
1. import java.io.BufferedReader;  
2. import java.io.IOException;  
3. import java.io.InputStreamReader;  
4. import java.io.PrintWriter;  
5. import java.net.ServerSocket;  
6. import java.net.Socket;  
7.  
8. /**  
9.  *  
10. * @author ferry  
11. */  
12. public class TCPEchoServer {  
13.  
14.     private static ServerSocket servSock;  
15.     private static final int PORT = 1234;  
16.  
17.     public static void main(String args[]) {  
18.         System.out.println("Membuka socket.....\n");  
19.         try {  
20.             servSock = new ServerSocket(PORT);  
21.         } catch (IOException e) {  
22.             System.out.println("Gagal membuka port !!!");  
23.             System.exit(1);  
24.         }  
25.         do {  
26.             run();  
27.         } while (true);  
28.     }  
29.  
30.     private static void run() {  
31.         Socket link = null;  
32.         try {  
33.  
34.             link = servSock.accept();  
35.             BufferedReader in = new BufferedReader(new  
36.                 InputStreamReader(link.getInputStream()));  
37.             PrintWriter out = new PrintWriter(link.getOutputStream(), true);
```

```
37.     int numMessages = 0;
38.     String message = in.readLine();
39.     while (!message.equals("close")) {
40.
41.         System.out.println("Pesan diterima : [" + message.toString() + "] dalam
    " + message.length() + " bytes");
42.         //System.out.println("Message received");
43.         numMessages++;
44.         out.println("Isi Pesan " + numMessages + ":" + message);
45.         message = in.readLine();
46.
47.     }
48.     out.println(numMessages + " buah pesan telah diterima.");
49. } catch (IOException e) {
50. } finally {
51.     try {
52.         System.out.println("*****Menutup koneksi *****");
53.         link.close();
54.     } catch (IOException e) {
55.         System.out.println("Tidak dapat memustukan koneksi");
56.         System.exit(1);
57.     }
58. }
59. }
60. }
```

C.3. Program Aplikasi Chat Client dengan menggunakan TCP

```
1. import java.io.BufferedReader;
2. import java.io.IOException;
3. import java.io.InputStreamReader;
4. import java.io.PrintWriter;
5. import java.net.InetAddress;
6. import java.net.Socket;
7. import java.net.UnknownHostException;
8.
9. /**
10. *
11. * @author ferry
12. */
13. public class TCPEchoClient {
14.
15.     private static String strHost;
16.     private static InetAddress host;
17.     private static final int PORT = 1234;
18. }
```

```
19. public static void main(String args[]) {
20.     try {
21.         // host = InetAddress.getLocalHost();
22.         strHost = "10.252.44.177" ; // <- Masukan sesuai dengan tujuan
23.         host = InetAddress.getByName(strHost);
24.     } catch (UnknownHostException e) {
25.         System.out.println("Alamat tidak ditemukan");
26.         System.exit(1);
27.     }
28.     run();
29. }
30.
31. private static void run() {
32.     Socket link = null;
33.     try {
34.         link = new Socket(host, PORT);
35.         BufferedReader in = new BufferedReader(new
InputStreamReader(link.getInputStream()));
36.         PrintWriter out = new PrintWriter(link.getOutputStream(), true);
37.
38.         BufferedReader userEntry = new BufferedReader(new
InputStreamReader(System.in));
39.         String message, response;
40.         do {
41.             System.out.print("Masukkan pesan : ");
42.             message = userEntry.readLine();
43.             out.println(message);
44.             response = in.readLine();
45.             System.out.println("SERVER " + response);
46.         } while (!message.equals("close"));
47.     } catch (IOException e) {
48.         e.printStackTrace();
49.     } finally {
50.         try {
51.             System.out.println("Menutup Koneksi.");
52.             link.close();
53.         } catch (IOException e) {
54.             System.out.println("Tidak dapat memutuskan koneksi!");
55.             System.exit(1);
56.         }
57.     }
58. }
59. }
```

D. Tahap praktikum

1. Tulis kembali program C1 dalam editor yang anda sukai.
2. Jalankan program “netstat -ap TCP”. Catat alamat IP sumber, alamat IP tujuan dan nomor port yang sedang dalam keadaan menunggu (listen()).
3. Jalankan program C1 ! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
4. Jalankan web browser dan arahkan URL ke alamat IP computer anda ditambah dengan nomor port dari web server percobaan!
5. Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
6. Buat flowchart untuk menggambarkan interaksi program C1 dan browser anda!
7. Tulis kembali program C2,C3 dalam editor yang anda sukai.
8. Jalankan program Aplikasi Server C2! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
9. Jalankan program C3! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
10. Kirim 3 pesan yang berbeda. Pesan terakhir adalah “close”. Catat hasil pada aplikasi server dan aplikasi client.
11. Jalankan perintah pada no 11! Amati perubahan port dari aplikasi client yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2. Mengapa terjadi perubahan pada port aplikasi client?
12. Buat flowchart untuk menggambarkan interaksi program C2 dan program C3!

E. Tugas

1. Port dibagi menjadi tiga kategori menurut nomornya: well-known port, user port dan dynamic/private port. Jelaskan latar belakang pengkategorian tersebut!
2. Sebutkan dan jelaskan secara singkat beberapa contoh aplikasi client-server yang sering anda gunakan !
3. Jelaskan perbedaan proses komunikasi dengan menggunakan TCP dan UDP!
4. Buatlah flowchart program aplikasi chat Server dan aplikasi chat Client dengan menggunakan protocol UDP!

5. Implementasikan flowchart pada soal no 4 yang sudah anda buat ke dalam program !

DAFTAR PUSTAKA

1. <http://www.oracle.com/technetwork/java/javase/documentation/index.html>
2. URL <http://www.ustudy.in/node/7214>