

Praktikum 11

PEMROGRAMAN SOCKET LANJUTAN

A. TUJUAN PEMBELAJARAN

1. Siswa memahami konsep aplikasi client server di jaringan.
2. Siswa memahami konsep pemrograman socket lanjut.
3. Siswa mampu melakukan socket programming dengan multi thread

B. DASAR TEORI

Pada praktikum sebelumnya anda telah membuat program aplikasi client-server. Kelemahan dari program sebelumnya adalah tiap server hanya bisa melayani satu koneksi. Padahal dalam aplikasi sebenarnya setiap aplikasi server mempunyai kemampuan menerima koneksi lebih dari satu.

Pemrograman Java menawarkan dua buah solusi , yaitu dengan menggunakan thread dan menggunakan RMI (Remote Method Invocation). Pada praktikum ini, kita akan berfokus pada solusi dengan menggunakan multithread. Seperti yang sudah anda pelajari pada mata kuliah sistem operasi, pada proses single thread proses akan menjalankan bagian program secara terurut, dan memakai resources secara bergantian. Berbeda dengan multi thread, dimana thread saling berbagi bagian program dan resources dengan thread lain yang mengacu pada proses yang sama.

Pada praktikum ini kita akan mempelajari strategi penggunaan multithread pada pemrograman client-server. Pada sisi aplikasi server terdapat penambahan 1 buah class bernama clientHandler yang digunakan untuk melayani koneksi ke client. Main class dari aplikasi server memanggil thread yang berisi class tersebut apabila ada aplikasi client yang meminta koneksi.

C. Contoh Program

C.1. Program Utama(Main Program) Aplikasi Server dengan Multithread

```
1. /**
2.  * @author ferry
3.  */
4.
5. import java.io.IOException;
6. import java.net.ServerSocket;
7.
8. public class TCPEchoServerThread {
9.
10.     private static ServerSocket servSock;
11.     private static final int PORT = 12345;
12.
13.     public TCPEchoServerThread() {
14.     }
15.
16.     public void start() {
17.         try {
18.             servSock = new ServerSocket(PORT);
19.
20.             while (true) {
21.                 Thread clientThread = new Thread(new
clientHandler(servSock.accept()));
22.                 clientThread.start();
23.             }
24.         } catch (IOException e) {
25.             e.printStackTrace();
26.         } finally {
27.             try {
28.                 System.out.println("Menutup koneksi....");
29.                 servSock.close();
30.             } catch (IOException e) {
31.                 System.out.println("Tidak dapat memustukan koneksi");
32.                 e.printStackTrace();
33.                 System.exit(1);
34.             }
35.         }
36.     }
37.
38.     public static void main(String[] args) {
39.         TCPEchoServerThread es = new TCPEchoServerThread();
40.         System.out.println("Server telah berjalan di komputer ini pada port " +PORT);
41.         es.start();
42.     }
```

43. }

C.2. Program Aplikasi Server Untuk Melayani Koneksi Client

```
1. import java.io.BufferedReader;
2. import java.io.IOException;
3. import java.io.InputStreamReader;
4. import java.io.PrintWriter;
5. import java.net.*;
6.
7. class clientHandler implements Runnable {
8.
9.     private static int numConnections;
10.    private int connectionId = 0;
11.    Socket link;
12.
13.    public clientHandler(Socket s) {
14.        connectionId = numConnections++;
15.        System.out.println("Melayani koneksi ke-" + connectionId);
16.        link = s;
17.    }
18.
19.    public void run() {
20.        PrintWriter out = null;
21.        BufferedReader in = null;
22.        int numMessages = 0;
23.
24.        try {
25.            out = new PrintWriter(link.getOutputStream(), true);
26.            in = new BufferedReader(new
InputStreamReader(link.getInputStream()));
27.            String message=in.readLine();
28.            while (!message.equals("close")) {
29.                System.out.println("Pesan diterima : [" +message.toString() + "] dari
client " + connectionId +" dalam " + message.length() + " bytes");
30.                numMessages++;
31.                out.println("Isi Pesan " + numMessages + ":" + message);
32.                message = in.readLine();
33.            }
34.        } catch (Exception e) {
35.            e.printStackTrace();
36.        } finally {
37.            out.close();
38.            try {
39.                in.close();
40.                link.close();
```

```
41.      System.out.println("Menutup koneksi, #" + connectionId);
42.      } catch (IOException e) {
43.          e.printStackTrace();
44.      }
45.  }
46.  }
47. }
```

D. PERCOBAAN

1. Tulis kembali program C1,C2 dalam editor yang anda sukai.
2. Jalankan program Aplikasi Server C1!
3. Jalankan program “netstat -ap TCP”. Catat alamat IP sumber , alamat IP tujuan dan nomor port yang sedang dalam keadaan menunggu (listen()).
4. Jalankan program client! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 3.
5. Jalankan program sebanyak 2 client lagi! Kirim 3 pesan yang berbeda. Pesan terakhir adalah “close”. Catat hasil pada aplikasi server dan aplikasi client.
6. Buat flowchart untuk menggambarkan interaksi program C1,C2 dan aplikasi client!

E. LAPORAN RESMI

1. Buat program grup chat client-server sederhana dengan kebutuhan sebagai berikut :
 - a. Terdapat 1 buah aplikasi server sebagai perantara komunikasi antara aplikasi client (client lebih dari satu)
 - b. Aplikasi server :
 - i. Menampilkan dan mencatat client yang telah melakukan koneksi ke server.
 - ii. Melakukan broadcast pesan yang diterima ke semua client yang telah terkoneksi ke server dengan mencantumkan asal pengirim dan waktu pengiriman.

- c. Aplikasi client
 - i. Mengirimkan pesan ke server
 - ii. Menerima pesan broadcast dari server

DAFTAR PUSTAKA

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