BAB 10
Jaringan

Java memperbolehkan Anda dalam mempermudah pengembangan aplikasi yang mengerjakan berbagai pekerjaan melalui jaringan. Ini adalah suatu cita-cita pembuatan Java yang menjadi salah satu kekuatan Java sejak dapat dibuat untuk ditampilkan melalui internet. Sebelum mempelajari tentang jaringan dalam Java. Pertama-tama Anda akan diperkenalkan kepada beberapa konsep dasar jaringan.

Pada akhir pembahasan, diharapkan pembaca dapat :
1.Mengerti konsep dasar jaringan
- IP address
- protokol
- ports
- paradigma client/server
- socket

2.Membuat aplikasi menggunakan package jaringan Java
- *ServerSocket*
- *Socket*
- *MulticastSocket*
- *DatagramPacket*

10.1 Konsep Dasar Jaringan

Jika sebelumnya Anda telah mengetahui, bahwa internet adalah jaringan global dengan berbagai jenis komputer yang berbeda yang tersambung dalam berbagai cara. Walaupun terdapat perbedaan dalam software dan hardware yang tersambung bersama-sama, hal tersebut sangatlah bagus bahwa internet masih dapat berfungsi. Hal ini memungkinkan karena standar komunikasi memiliki ketetapan dan juga keselarasan. Standar ini menjamin kesesuaian dan kekuatan komunikasi diantara luasnya sistem pada internet. Mari kita pelajari beberapa standar yang berlaku.

10.1.1 IP Address
Pada setiap komputer yang tersambung dengan internet memiliki alamat IP yang unik. Alamat IP secara logika hampir sama dengan alamat pengiriman surat tradisional dimana memiliki arti bahwa alamat yang bersifat unik tersebut mewakili dari keterangan sebuah object. Alamat tersebut diwakilkan dalam 32-bit nomor yang digunakan sebagai pengenal yang bersifat unik dari setiap komputer yang tersambung dengan internet. *192.1.1.1* adalah contoh dari sebuah alamat IP. Mereka juga bisa ditulis dengan bentuk simbol seperti *docs.rinet.ru.*

10.1.2 Protokol
Karena terdapat jenis komunikasi yang berbeda-beda yang mungkin terjadi pada internet, di sana harus terdapat suatu jumlah yang sama untuk mekanisme penangangan komunikasi. Setiap jenis komunikasi membutuhkan protokol yang spesifik dan unik.

Protokol mengatur peraturan dan standar yang menetapkan jenis komunikasi internet yang khusus. Hal tersebut menjelaskan format data yang dikirim lewat internet, seiring dengan bagaimana dan kapan itu dikirim.

Konsep dari protokol tentunya tidak terlalu asing untuk kita. Mengingat sudah beberapa kali Anda telah menggunakan jenis percakapan ini :

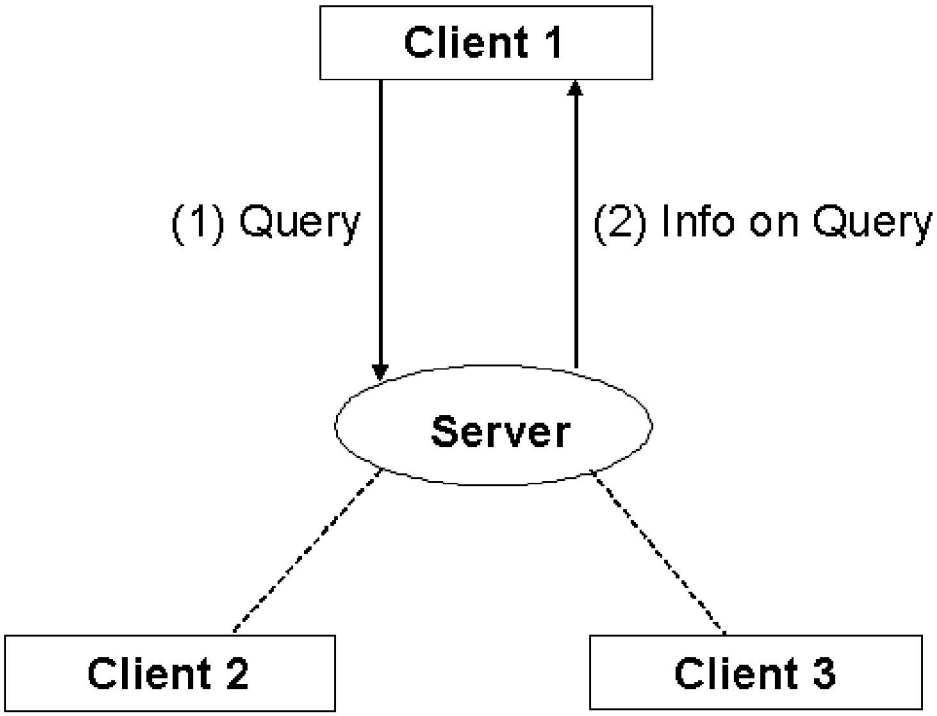
"Hallo."
"Hallo. Selamat siang. Bolehkah saya berbicara dengan Joan?"
"Okay, mohon tunggu sebentar."
"terima kasih."
...
Ini adalah protokol sosial yang digunakan ketika dalam pembicaraan melalui telepon. Jenis protokol tipe ini memberikan kita kepercayaan untuk mengetahui apa yang harus dilakukan dalam situasi tersebut. Mari kita lihat beberapa protokol penting yang digunakan pada internet. Hypertext Transfer Protocol (HTTP) adalah salah satu protokol yang sering digunakan. Digunakan untuk mentransfer dokumen HTML pada Web. Kemudian, ada juga File Transfer Protocol (FTP) dimana lebih umum dibandingkan dengan HTTP dan memperbolehkan Anda untuk mentransfer file biner lewat intenet. Kedua protokol tersebut memiliki peraturan masing-masing dan standar dalam pengiriman data. Java juga mendukung kedua protokol tersebut.

10.1.3 Port
Sekarang, protokol hanya bisa dipertimbangkan jika digunakan dalam konteks suatu jasa. Sebagai contoh, protokol HTTP digunakan ketika Anda menyediakan isi Web melalui layanan HTTP. Setiap komputer pada internet dapat menyediakan berbagai jenis layanan melalui berbagai jenis protokol yang mendukung. Masalahnya, bagaimanapun juga, kita harus mengetahui jenis layanan sebelum sebuah informasi dapat ditransfer. Untuk itulah port digunakan.

Port adalah 16-bit nomor dimana mengenal setiap layanan yang ditawarkan oleh server jaringan. Untuk menggunakan layanan khusus dan oleh karena itu, jalur komunikasi yang melewati protokol tertentu, Anda perlu menyambungkan pada port yang sesuai. Port dihubungkan dengan nomor dan beberapa nomor bersifat spesifik yang berhubungan dengan jenis layanan khusus. Port dengan layanan pekerjaan tertentu disebut port standar. Sebagai contoh, layanan FTP terletak pada port 21 sedangkan layanan HTTP terletak pada port 80. Jika Anda ingin menggunakan file transfer FTP, Anda perlu terhubung dengan port 21 pada komputer Anda. Sekarang, semua standar layanan tertentu diberikan nilai port dibawah 1024. port dengan nilai diatas 1024 disediakan untuk komunikasi custom. Jika terdapat kasus dimana port dengan nilai diatas 1024 telah digunakan oleh beberapa komunikasi custom, Anda harus mencari nilai lain yang tidak digunakan.

10.1.4 Paradigma client/server
Paradigma client/server adalah dasar untuk framework jaringan Java. Tentunya, penetapan ini terdiri dari dua elemen besar, yaitu client dan server. Client adalah mesin yang membutuhkan beberapa jenis informasi sedangkan server adalah mesin yang menyimpan informasi dan menunggu untuk menyampaikannya pada client.

Paradigma ini menjelaskan sebuah skenario sederhana. Tentunya, client terhubung dengan sever dan meminta informasi. Kemudian server mengingat permintaan dan mengembalikan informasi yang tersedia kepada client.



Gambar1.1.4: Model Client/Server

10.1.5 sockets

Konsep umum jaringan yang terakhir sebelum kita membahas lebih dalam tentang Java networking adalah dengan memperhatikan *sockets.* Kebanyakan pemrograman Java network menggunakan jenis khusus dari komunikasi jaringan yang diketahui sebagai *sockets.*

Socket adalah software abstrak untuk media input atau output komunikasi. Socket digunakan oleh Java untuk mengatasi komunikasi pada jaringan level rendah. Jalur komunikasi ini memungkinkan untuk mentransfer data melalui port khusus. Singkatnya, socket adalah point terakhir untuk komunikasi antara dua mesin.

10.2 The Java Networking Package

Package dari *java.net* menyediakan banyak class yang berguna untuk pengembangan aplikasi jaringan. Untuk daftar lengkap dari class jaringan dan interface, dapat dilihat pada dokumentasi API. Pembelajaran akan difokuskan pada empat class yaitu : class ServerSocket, Socket, MulticastSocket, dan DatagramPacket.

10.2.1 Class ServerSocket dan Socket

Class ServerSocket menyediakan fungsi-fungsi dasar dari sebuah server. Tabel berikut menjelaskan dua dari empat constructor pada class ServerSocket:

Constructor ServerSocket
`ServerSocket(int port)`
Ketika sebuah server menetapkan suatu port tertentu, sebuah port 0 menugaskan sebuah server kepada port bebas manapun. Panjang antrian maksimum untuk koneksi yang akan datang diatur sebanyak 50 sebagai defaultnya.
`ServerSocket(int port, int backlog)`
Ketika sebuah server menetapkan suatu port tertentu, panjang antrian maksimum untuk koneksi yang akan datang berdasarkan pada parameter backlog.

Tabel 1.2.1a: Constructor ServerSocket

Berikut ini adalah beberapa method class pada ServerSocket :

Method ServerSocket
`public Socket accept()`
Menyebabkan server untuk menunggu dan mendengarkan dari koneksi client, kemudian menerimanya.
`public void close()`
Menutup socket server. Client tidak dapat lagi terhubung ke server hingga dibuka kembali
`public int getLocalPort()`
Mengembalikan port dimana socket juga membatasi
`public boolean isClosed()`
Mendeteksi apakah socket tertutup atau belum

Tabel 1.2.1b: Method ServerSocket

Contoh yang berhasil melakukan implementasi sebuah server sederhana, dimana sebuah informasi sederhana dikirim oleh client dapat dilihat pada listing program berikut ini :

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

public class EchoingServer {
   public static void main(String [] args) {
      ServerSocket server = null;
      Socket client;

      try {
         server = new ServerSocket(1234);
```

```
                //1234 nomor port yang belum digunakan
            } catch (IOException ie) {
                System.out.println("Cannot open socket.");
                System.exit(1);
            }

            while(true) {
                try {
                    client = server.accept();
                    OutputStream clientOut = client.getOutputStream();
                    PrintWriter pw = new PrintWriter(clientOut, true);
                    InputStream clientIn = client.getInputStream();
                    BufferedReader br = new BufferedReader(new
                                        InputStreamReader(clientIn));
                    pw.println(br.readLine());
                } catch (IOException ie) {
                }
            }
        }
    }
```

Ketika class ServerSocket mengimplementasikan server socket, Class Socket mengimplementasikan socket client. Class *Socket* memiliki delapan constructor, dua diantaranya siap dipanggil. Langsung saja kita lihat dua constructor tersebut.

Constructor Socket
`Socket(String host, int port)`
Membuat sebuah socket client dimana dihubungkan dengan diberikan nomor port pada host tertentu.
`Socket(InetAddress address, int port)`
Membuat sebuah socket client dimana dihubungkan dengan diberikannya nomor port pada alamat IP tertentu.

Tabel 1.2.1c: Constructor Socket

Berikut adalah beberapa dari method class pada Socket :

Method Socket
`public void close()`
Menutup socket client
`public InputStream getInputStream()`
Menerima kembali input stream yang berhubungan dengan socket ini.
`public OutputStream getOutputStream()`
Menerima kembali output stream yang berhubungan dengan socket ini.
`public InetAddress getInetAddress()`

Method Socket
Mengembalikan alamat IP kepada socket ini pada saat masih terhubung.
`public int getPort()`
Mengembalikan remote port kepada socket ini pada saat masih terhubung.
`public boolean isClosed()`
Mendeteksi apakah socket telah tertutup atau tidak

Tabel 1.2.1d: Method Socket

Contoh yang berhasil melakukan implementasi sebuah client sederhana, dimana mengirim data kepada server dapat dilihat pada listing program dibawah ini :

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

       public class MyClient {
          public static void main(String args[]) {
             try {
                //Socket client = new Socket("133.0.0.1", 1234);
                Socket client = new Socket(InetAddress.getLocalHost(),
                                           1234);
                InputStream clientIn = client.getInputStream();
                OutputStream clientOut = client.getOutputStream();
                PrintWriter pw = new PrintWriter(clientOut, true);
                BufferedReader br = new BufferedReader(new
                                        InputStreamReader(clientIn));
                BufferedReader stdIn = new BufferedReader(new
                                        InputStreamReader(System.in));
                System.out.println("Type a message for the server: ");
                pw.println(stdIn.readLine());
                System.out.println("Server message: ");
                System.out.println(br.readLine());
                pw.close();
                br.close();
                client.close();
             } catch (ConnectException ce) {
                System.out.println("Cannot connect to the server.");
             } catch (IOException ie) {
                System.out.println("I/O Error.");
             }
          }
       }
```

10.2.2 Class MulticastSocket dan DatagramPacket

Class MulticastSocket sangat berguna untuk aplikasi yang mengimplementasikan komunikasi secara berkelompok. Alamat IP untuk kelompok multicast berkisar antara 224.0.0.0 hingga 239.255.255.255. Meskipun begitu, alamat 224.0.0.0 telah dipesan dan seharusnya tidak digunakan. Class ini memiliki tiga constructor tetapi yang akan dibahas hanya salah satu dari ketiga constructor ini.

Constructor MulticastSocket
`MulticastSocket(int port)`
Membuat multicast socket dibatasi dengan pemberian nomor port

Tabel 1.2.2a: Constructor MulticastSocket

Tabel berikutnya memberikan penjelasan beberapa method *MulticastSocket*.

Method MulticastSocket
`public void joinGroup(InetAddress mcastaddr)`
Bergabung dengan kelompok multicast pada alamat tertentu
`public void leaveGroup(InetAddress mcastaddr)`
Meninggalkan kelompok multicast pada alamat tertentu
`public void send(DatagramPacket p)`
Metode turunan dari class DatagramSocket. Mengirim *p* dari socket ini.

Tabel 1.2.2b:Method MulticastSocket

Sebelum seseorang dapat mengirim pesan kepada suatu kelompok, pertama kali yang harus dilakukan oleh orang tersebut adalah harus menjadi anggota dari multicast kelompok dengan menggunakan method *joinGroup*. Sekarang seorang anggota dapat mengirim pesan melalui method *send*. Jika Anda telah selesai berbicara dengan kelompok, Anda dapat menggunakan method *leaveGroup* untuk melepaskan keanggotaan Anda.

Sebelum melihat contoh dalam menggunakan class multicastSocket, pertama-tama mari kita lihat pada class DatagramPacket. Perhatikan bahwa dalam method *send* dari class multiSocket, dibutuhkan parameter yaitu object DatagramPacket. Sehingga, kita harus mengerti object jenis ini sebelum menggunakan method *send*.

Class *DatagramPacket* digunakan untuk mengirim data melalui protokol connectionless seperti multicast. Masalah yang ditimbulkan bahwa pengiriman packet tidak terjamin. Mari kita perhatikan dua dari enam constructor.

Constructor DatagramPacket
`DatagramPacket(byte[] buf, int length)`
Constructor dari datagramPacket untuk menerima paket dengan panjang *length.* Seharusnya kurang dari atau sama dengan ukuran dari buffer *buf.*
`DatagramPacket(byte[] buf, int length, InetAddress address, int port)`
Constructor dari datagramPacket untuk mengirim paket dengan panjang *length* dengan nomor port tertentu dan host tertentu.

Tabel 1.2.2c: Constructor DatagramPacket

Berikut adalah beberapa method dari class DatagramPacket.

Method-method DatagramPacket
`public byte[] getData()`
Mengembalikan buffer dimana data telah disimpan
`public InetAddress getAddress()`
Mengembalikan alamat IP mesin dimana paket yang dikirim atau yang diterima
`public int getLength()`
Mengembalikan panjang data yang dikirim atau diterima
`public int getPort()`
Mengembalikan nomor port pada remote host dimana paket yang dikirim atau yang diterima

Table 1.2.2d: Method DatagramPacket

Contoh multicast kita juga mengandung dua class, server dan client. Server menerima pesan dari client dan mencetak pesan tersebut.

Berikut adalah class server

```
import java.net.*;

public class ChatServer {
   public static void main(String args[]) throws Exception {
      MulticastSocket server = new MulticastSocket(1234);
      InetAddress group = InetAddress.getByName("234.5.6.7");
      //getByName – Mengembalikan alamat IP yang diberikan oleh Host
      server.joinGroup(group);
      boolean infinite = true;
      /* Server terus-menerus menerima data dan mencetaknya*/
      while(infinite) {
         byte buf[] = new byte[1024];
         DatagramPacket data = new DatagramPacket(buf,
                                                  buf.length);
         server.receive(data);
         String msg = new String(data.getData()).trim();
         System.out.println(msg);
      }
      server.close();
   }
}
```

Berikut adalah class client

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

public class ChatClient {
   public static void main(String args[]) throws Exception {
      MulticastSocket chat = new MulticastSocket(1234);
```

```
            InetAddress group = InetAddress.getByName("234.5.6.7");
            chat.joinGroup(group);
            String msg = "";
            System.out.println("Type a message for the server:");
            BufferedReader br = new BufferedReader(new
                                    InputStreamReader(System.in));
            msg = br.readLine();
            DatagramPacket data = new DatagramPacket(msg.getBytes(),
                        0, msg.length(), group, 1234);
            chat.send(data);
            chat.close();
        }
    }
```

10.3 Latihan

10.3.1 Trivia Server

Buatlah sebuah server yang berisi satu set pertanyaan yang mudah. Secara sederhana, akan ada sekitar 5-10 pertanyaan.

Client yang terhubung ke server mengirim sebuah permintaan untuk sebuah pertanyaan atau jawaban sebuah pertanyaan, Client mengirim pesan “permintaan”. Untuk jawaban dari sebuah pertanyaan, client mengirim pesan “jawaban”. Ketika menerima pesan “permintaan”, secara acak server akan memilih satu pertanyaan dari koleksi yang ada. Dia mengirimkan pertanyaan yang dipilih sesuai dengan nomor yang bersangkutan kepada client.

Ketika server menerima pesan “jawaban” dari client, dia menginformasikan user bahwa user perlu mengirimkan jawaban sesuai dengan nomor pertanyaan kepada server. Jawaban itu harus dalam format <no pertanyaan>#<jawaban Anda>.

Berikut adalah contoh skenario :
Client: "permintaan"
Server: "3#Siapa pembuat Java?"
Client: "jawaban"
Server: "Berikan jawabanmu dengan format: <nomor pertanyaanr>#<jawaban Anda>"
Client: "3#James Gosling"
Server: Kerja yang bagus!
...