Bab 10
Jaringan

Java mengijinkan anda untuk mempermudah mengembangkan aplikasi yang mengerjakan berbagai pekerjaan melalui jaringan. Ini adalah suatu cita-cita pembuatan Java yang menjadi salah satu kekuatan Java sejak dibuat dengan internet di dalam cita-cita. Sebelum mempelajari tentang jaringan dalam Java. Pertama-tama anda akan diperkenalkan kepada beberapa konsep dasar jaringan.

Setelah menyelesaikan bab ini, anda diharapkan dapat :
1.Mengerti konsep dasar jaringan
- IP addres
- protokol
- ports
- client/server
- socket

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

10.1 Konsep dasar jaringan

Jika sebelumnya anda suda mengetahui, internet adalah jaringan global dengan berbagai jenis komputer yang berbeda yang tersambung dalam berbagai jalan. Walaupun terdapat perbedaan dalam software dan hardware yang disambung bersama, hal tersebut sangatlah bagus dimana internet masih 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 melihat beberapa standar yang berlaku.

10.1.1 IP Adress
Pada setiap komputer yang terkoneksi dengan internet memiliki alamat IP 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 objek. 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
Sejak terdapat jenis komunikasi yang berbeda-beda yang mungkin terjadi pada internet,disana harus pula ada suatu jumlah yang sama untuk mekanisme penangangan mereka . Setiap jenis komunikasi membutuhkan protokol yang spesifik dan unik.

Protokol mengatur peraturan dan standar dimana 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 :

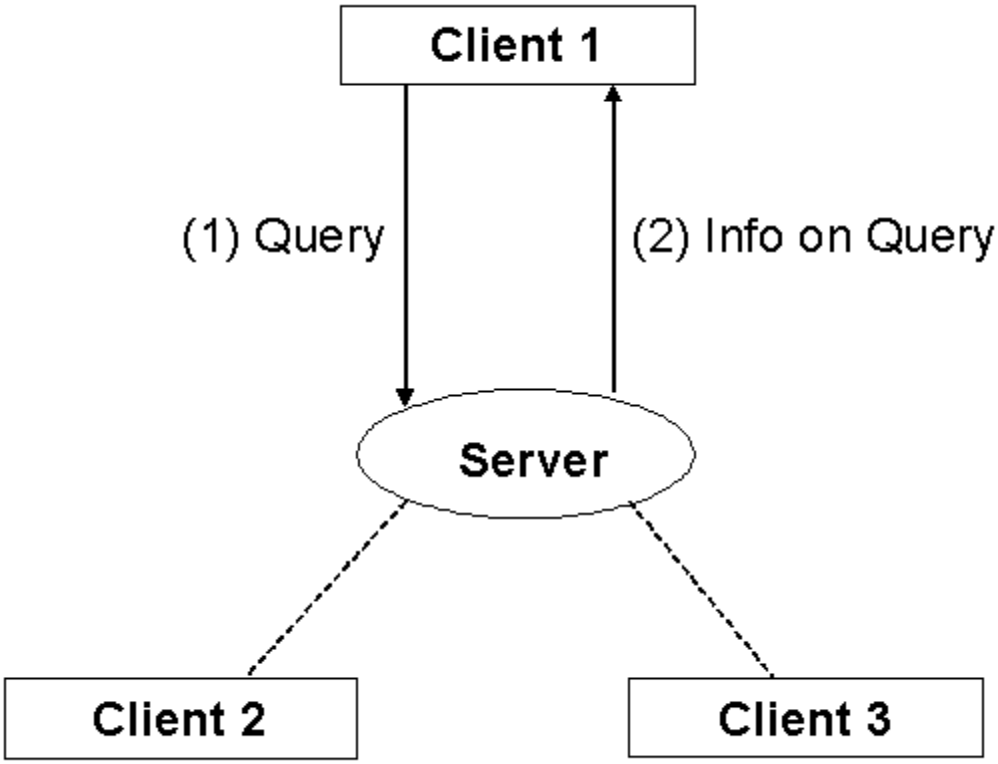
"Hallo."
"Hallo. Selamat siang. Bolehkah saya berbicara dengan Joan?"
"Okay, mohon tunggu sebentar."
"terima kasih."
...
Ini adalah protokol sosial yang digunakan ketika dalam pembicaraan melalui telepon. Jenis protokol tipe ini memberikan kita kepercayaan untuk mengetahui apa yang harus dilakukan dalam situasi tersebut. Mari kita lihat beberapa protokol penting yang digunakan pada internet. Tanpa banyak kata, 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 mengijinkan anda untuk mentransfer file biner pada intenet. Kedua protokol tersebut memiliki peraturan masing-masing dan standar dalam pengiriman data. Java juga dapat mendukung kedua protokol tersebut.

10.1.3 Port
Sekarang, protokol hanya bisa dipertimbangkan manakala 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, 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 untuk 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 lainnya yang tidak digunakan.

10.1.4 Paradigma client/server
Paradigma client/server adalah dasar untuk Java networking framework. 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: Client/Server model

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 terkahir 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 merujuk ke dokumentasi API. Kita akan fokus pada empat class yaitu : ServerSocket, Socket, MulticastSocket, dan DatagramPacket class.

10.2.1 ServerSocket and the Socket class

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

Konstruktor ServerSocket
`ServerSocket(int port)`
Ketika sebuah server menetapkan suatu port tertentu. Sebuah port 0 menugaskan sebuah server kepada port bebas manapun. Panjang antrian maksimun 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: Konstruktor ServerSocket

Berikut adalah beberapa dari class method :

Metode ServerSocket
`public Socket accept()`
Menyebabkan server untuk menunggu dan mendengarkan dari koneksi client, lalu 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: Metode ServerSocket

Contoh yang berhasil melakukan implementasi sebuah server sederhana, dimana sebuah informasi sederhana dikirim oleh client.

```
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 is an unused port number
                } 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 ServerSocket class mengimplementasikan server socket, Socket class mengimplementasikan client socket. Socket class memiliki delapan konstruktor, dua diantaranya siap dipanggil. Langsung saja kita lihat dua konstruktor tersebut.

Konstruktor 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 diberikan nomor port pada alamat IP tertentu.

Tabel 1.2.1c: Konstruktor Socket

Berikut adalah beberapa dari class method :

Metode 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()`

Metode 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: Metode Socket

Contoh yang berhasil melakukan implementasi sebuah client sederhana, dimana mengirim data kepada server.

```
      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 MulticastSocket dan DatagramPacket class

class MulticastSocket sangat berguna untuk aplikasi yang mengimplementasikan komunikasi secara berkelompok. Alamat IP untuk kelompok multicast berkisar diantara 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 konstruktor tetapi kita akan membahas satu dari ketiga konstruktor ini.

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

Tabel 1.2.2a: Konstruktor MulticastSocket

Tabel berikutnya memberikan penjelasan beberapa metode MulticastSocket.

Metode 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:Metode MulticastSocket

Sebelum seorang dapat mengirim pesan kepada kelompok, pertama kali seorang tersebut seharusnya menjadi anggota dari multicast kelompok dengan menggunakan metode joinGroup. Sekarang seorang anggota dapat mengirim pesan melalui metode send. Jika anda telah selesai berbicara dengan kelompok, anda dapat menggunakan metode leavekelompok untuk melepaskan keanggotaan anda.

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

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

Konstruktor DatagramPacket
`DatagramPacket(byte[] buf, int length)`
Konstruktor 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)`
Konstruktor dari datagramPacket untuk mengirim paket dengan panjang *length* dengan nomor port tertentu dan host tertentu.

Tabel 1.2.2c:Konstruktor DatagramPacket

Berikut adalah beberapa metode menarik dari class DatagramPacket.

Metode 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: Metode 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 mencetak mereka */
      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!
...