Κατανεμημένα Συστήματα

2ο Φυλλάδιο εργαστηρίου (09/03/2017)

Java Sockets

Η πιο διαδεδομένη αρχιτεκτονική ανάπτυξης δικτυακών εφαρμογών είναι η αρχιτεκτονική του πελάτη - εξυπηρετητή (client - server). Ο εξυπηρετητής είναι μια διεργασία, η οποία εκτελείται σε έναν υπολογιστή και αναμένει να συνδεθεί σε αυτήν κάποιο πρόγραμμα - πελάτης, για να του παράσχει υπηρεσίες.

Η Java μας δίνει την δυνατότητα της ανταλλαγής δεδομένων μεταξύ προγραμμάτων που τρέχουν σε διαφορετικούς υπολογιστές. Αυτό επιτυγχάνεται με την δημιουργία sockets μεταξύ των δύο προγραμμάτων σε μια αρχιτεκτονική πελάτη-εξυπηρετητή.

Ως Socket ορίζεται το ένα άκρο, από έναν επικοινωνιακό δίαυλο διπλής κατεύθυνσης, μεταξύ δυο προγραμμάτων που εκτελούνται στο δίκτυο. Περιλαμβάνει το πρωτόκολλο, την διεύθυνση και τον αριθμό θύρας του άκρου.

Κάθε πρόγραμμα διαβάζει από και γράφει σε ένα socket, με χρήση Input και OutputStreams. Έτσι η διαδικασία αυτή δε διαφέρει σε τίποτα με την εγγραφή και ανάγνωση αρχείων. Μπορούμε να δημιουργήσουμε δύο είδη sockets ανάλογα με το πρωτόκολλο μεταφοράς που θα χρησιμοποιήσουμε :

- Στην πρώτη περίπτωση βασίζεται στο TCP (Transmission Control Protocol) και είναι μια υπηρεσία προσανατολισμένη στην σύνδεση (connection-oriented service). Μπορούμε να το θεωρήσουμε ανάλογο της τηλεφωνικής υπηρεσίας, στην οποία, μετά την εγκαθίδρυση μιας σύνδεσης μεταξύ δυο συνομιλητών, αυτή χρησιμοποιείται μέχρι το πέρας της συζητήσεως τους.

- Στην δεύτερη περίπτωση βασίζεται στο UDP (User Datagram Protocol) και είναι μια υπηρεσία χωρίς σύνδεση (connectionless service). Το ανάλογο, σε αυτήν την περίπτωση, είναι το ταχυδρομείο: μπορούμε να στείλουμε πολλά πακέτα στον ίδιο παραλήπτη, αλλά δεν είναι σίγουρο ότι όλα θα ακολουθήσουν την ίδια διαδρομή (σύνδεση) για να φτάσουν στον προορισμό τους.

Μια ακόμη σημαντική διαφορά, μεταξύ των παραπάνω δύο ειδών, είναι ότι τα TCP sockets εξασφαλίζουν μια αξιόπιστη μεταφορά της πληροφορίας, ότι αποστέλλεται από το ένα άκρο είναι σίγουρο ότι θα φτάσει στο άλλο. Στο UDP socket όμως δεν συμβαίνει αυτό. Είναι στην ευθύνη του αποστολέα να ελέγξει ότι αυτό που έστειλε, το έλαβε τελικά ο παραλήπτης και δεν χάθηκε στον δρόμο. Επιπλέον, η σύνδεση με TCP socket απαιτεί την ανταλλαγή τριών "πακέτων χειραψίας" (handshake packets) και είναι πιο χρονοβόρα στην αρχικοποίηση της από την αντίστοιχη με UDP datagrams. Οι προηγούμενες δύο διαφορές καθορίζουν τελικά και την χρήση των δύο αυτών ειδών.

Στην ορολογία της Java, ο όρος *Socket* χρησιμοποιείται για τα TCP sockets στην ονοματολογία των κλάσεων και των μεθόδων, ενώ για την δήλωση των UDP sockets, χρησιμοποιείται ο όρος *Datagram*.

Ο πελάτης (Client)

Στον κώδικα του πελάτη θα πρέπει να δημιουργήσουμε μία σύνδεση προς τον εξυπηρετητή. Για να το επιτύχουμε αυτό χρειαζόμαστε να γνωρίζουμε την **διεύθυνση (IP address)** του και την **πόρτα (port)** που περιμένει την σύνδεση. Η διεύθυνση μπορεί να εκφραστεί σαν όνομα πεδίου (domain name) είτε σαν IP διεύθυνση. Αυτές οι δύο τιμές δίνονται ως παράμετροι στον δημιουργό της κλάσης ***java.net.Socket***. Αφού τρέξουμε τον δημιουργό της κλάσης, επιτυγχάνεται η σύνδεση μεταξύ των δύο προγραμμάτων.

Ανταλλαγή μηνυμάτων

Στην συνέχεια για να ανταλλάξουμε μηνύματα μεταξύ των δύο προγραμμάτων θα πρέπει να πάρουμε το *InputStream* και *OutpuStream* της σύνδεσης με τις μεθόδους `InputStream` **`getInputStream`**`()` και `OutputStream` **`getOutputStream`**`()` για να λάβουμε και να στείλουμε αντίστοιχα. Οπότε τώρα χρησιμοποιώντας τις μεθόδους που ορίζονται στις κλάσεις *InputStream* και *OutpuStream* μπορούμε να ανταλλάξουμε δεδομένα. Τέλος για να κλείσουμε την σύνδεση εκτελούμε την μέθοδο `void` **`close`**`()` της κλάσης *java.net.Socket.*

Χρήσιμες Πληροφορίες

Επιπλέον στην κλάση *java.net.Socket* ορίζονται διάφορες μέθοδοι που μας δίνουν χρήσιμες πληροφορίες για την σύνδεση. Κάποιες από αυτές είναι:

```
InetAddress getLocalAddress()
InetAddress getInetAddress()
```

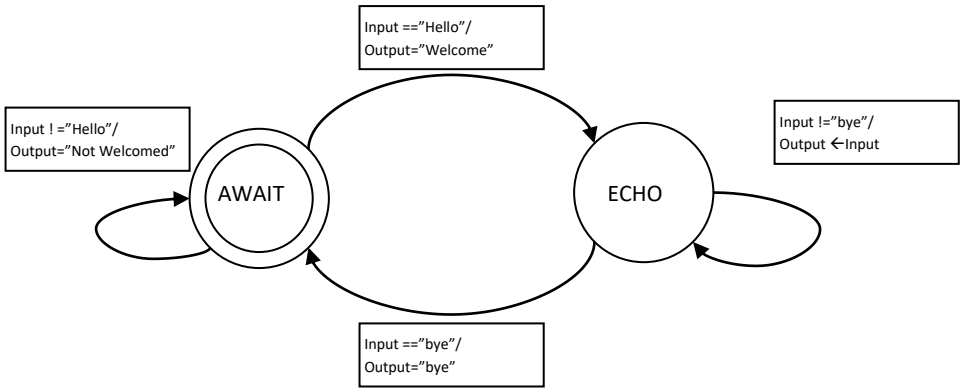
οι οποίες επιστρέφουν την τοπική και την απομακρυσμένη διεύθυνση της σύνδεσης.

```
int getLocalPort()
int getPort()
```

οι οποίες επιστρέφουν την τοπική και την απομακρυσμένη πόρτα της σύνδεσης.

1η Εργαστηριακή Άσκηση – Επικοινωνία Client-Server / Echo Εφαρμογή

Καλείστε να δημιουργήσετε έναν πελάτη που θα συνδέεται σε έναν εξυπηρετητή (θα ενημερωθείτε κατά τη διάρκεια του εργαστηρίου για την ip διεύθυνση του) και ο οποίος δέχεται συνδέσεις στην θύρα 5555. Ο εξυπηρετητής υλοποιεί το παρακάτω πρωτόκολλο.



Ενδεικτική έξοδο της εφαρμογής μπορεί να είναι η ακόλουθη:

1η εκτέλεση:
Sending Messages to the Server...
Connecting to /83.212.114.133 and port 5555
Local Address :/195.251.163.201 Port :9442
Write what the client will send: Hello
The server says: Welcome
Write what the client will send: hi
The server says: hi
Write what the client will send: How are you?
The server says: How are you?
Write what the client will send: bye
The server says: bye
Connection Closing...

2η εκτέλεση:
Sending Messages to the Server...
Connecting to /83.212.114.133 and port 5555
Local Address :/195.251.163.201 Port :9429
Write what the client will send: Hi
The server says: Not Welcomed
Connection Refused!!!

Ενδεικτική Λύση:

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

public class EchoClient {

    public static void main(String[] args)  {

      try{

        Socket sock = new Socket("83.212.114.133", 5555);

        BufferedReader instream = new BufferedReader(new InputStreamReader
(sock.getInputStream()));
        BufferedWriter outstream = new BufferedWriter(new
OutputStreamWriter(sock.getOutputStream()));
```

```
        System.out.println("Sending Messages to the Server...");
        System.out.println("Connecting to "+ sock.getInetAddress()+ " and
port "+sock.getPort());
        System.out.println("Local Address :"+sock.getLocalAddress()+" Port
:"+sock.getLocalPort());


        String strin, strout;

       Scanner in = new Scanner(System.in);
       do{
           System.out.print("Write what the client will send: ");
           strout = in.nextLine();
           outstream.write(strout+"\n");
           outstream.flush();
           strin = instream.readLine();
           System.out.println("The server says: "+strin);
       }
       while (!strin.equals("bye"));

        instream.close();
        outstream.close();
        sock.close();
        System.out.println("Connection Closing...");
        }
      catch (IOException ex){
          System.out.println("Connection Refused!!!");

        }
    }
  }
```

2η Εργαστηριακή Άσκηση – Εκτέλεση αριθμητικών πράξεων σε μοντέλο πελάτη-εξυπηρετητή

Καλείστε να δημιουργήσετε έναν πελάτη ο οποίος θα συνδέεται σε έναν εξυπηρετητή (θα ενημερωθείτε κατά τη διάρκεια του εργαστηρίου για την ip διεύθυνση του) και ο οποίος δέχεται συνδέσεις στην θύρα 5556. Η εφαρμογή του πελάτη διαβάζει από τον χρήστη 2 ακέραιους αριθμούς και έναν αριθμητικό τελεστή (+, -, * και /). Τα δεδομένα αυτά τα στέλνει στον εξυπηρετητή ο οποίος αναλαμβάνει να εκτελέσει την αριθμητική πράξη και να επιστρέψει το τελικό αποτέλεσμα στον πελάτη. Ο εξυπηρετητής επιστρέφει και ένα μήνυμα εάν ήταν δυνατή η όχι η εκτέλεση της πράξης.

Ενδεικτική Λύση:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.util.Scanner;

public class Calc_Client {
      public static void main(String[] args)  {
```

```
      try{

        Socket sock = new Socket("localhost", 5556);

        DataInputStream instream = new DataInputStream
(sock.getInputStream());
        DataOutputStream outstream = new
DataOutputStream(sock.getOutputStream());

        System.out.println("Sending Messages to the Server...");
        System.out.println("Connecting to "+ sock.getInetAddress()+ " and
port "+sock.getPort());
        System.out.println("Local Address :"+sock.getLocalAddress()+" Port
:"+sock.getLocalPort());


       Scanner in = new Scanner(System.in);

       System.out.print("Write the first number: ");
       int number1 = in.nextInt();
       outstream.writeInt(number1);

       System.out.print("Write the second number: ");
       int number2 = in.nextInt();
       outstream.writeInt(number2);

       System.out.print("Write the operator: ");
       String operator = in.next();
       outstream.writeUTF(operator);

       double result = instream.readDouble();
       String message = instream.readUTF();
       if (result == -1)
             System.out.println(message);
       else
             System.out.println("The server says: "+ result);

        instream.close();
        outstream.close();
        sock.close();
        System.out.println("Connection Closing...");
        }
      catch (Exception ex){
          System.out.println("Connection Refused!!!");

        }
    }
}
```