

2021.06.09 Java

〈Network 추가 설명〉

《 server

```
import ...  
// 서비스는 ?? >> 내가 하기 힘들니까 돈 줄테니까 너가 해줘  
// 서버 >> 서비스 제공자  
// 클라이언트 >> 서비스 이용자  
public class _1st_SocketServerEdu {  
    public static void main(String[] args) {  
        int port = Integer.parseInt(s: "33333");  
        try{  
            ServerSocket servSock = new ServerSocket(port);  
            // 통신을 수행할 수 있도록 내 socket을 만듦.  
            // SOCKET >>  
            // 전기 분야에서의 SOCKET에 전원 코드를 연결하면 전기 제품을 구동가능한 것과 마찬가지로  
            // 프로그래밍 분야에서 SOCKET이란 다른 컴퓨터와 내 컴퓨터를 연결하는 통로 역할을 함.  
            System.out.println("Server: Listening - " + port);  
            while(true){  
                // accept() 부분에서 서버는 Blocking(블록킹) 연산을 수행하고 있음.  
                // client의 접속이 없을 때는 승인 등의 일을 blocking 하고 있다는 의미.  
                // client가 준비될때까지 계속 기다린다.(문 두드리면서 / spinlock이란 비슷한 개념)  
                // >> client가 socket을 요청하기 전까지 계속.  
  
                Socket sock = servSock.accept();  
                // >> client가 접속하면 그 후부터 다음 code 시행.  
                // >> 위의 sock은 접속한 사용자 socket임.  
                // sock.getInetAddress() > 접속한 client의 ip 확인  
                System.out.println "[" + sock.getInetAddress() + "] client connected");  
            }  
        }  
    }  
}
```

```
//client(입력) <----- server(출력)
//client(출력) -----> server(입력)
```

송신

```
OutputStream out = sock.getOutputStream(); // client를 향해 출력할 객체를 생성성
PrintWriter writer = new PrintWriter(out, autoFlush: true);
// PrintWriter에 송신용 객체(out)을 배치함으로
// writer.println으로 구동시키는 것을 전송되게 만듦.

writer.println(new Date().toString());
// 누군가가 접속하면 접속 시간을 알려주는 서비스를 제공
// 이 시점에 client에게 시간 정보가 전송됨.
```

수신

```
InputStream in = sock.getInputStream(); // client로부터 입력받을 객체를 생성
BufferedReader reader = new BufferedReader(new InputStreamReader(in));
// InputStream을 사용해서 들어오는 객체는 반드시 위의 bufferedReader 코드 형식으로 읽어야 한다.
// InputStreamReader(): InputStream 읽기
// BufferedReader(): 데이터가 많이 들어오거나 빈번하게 지속적으로 들어올 수 있어
// 버퍼를 가진 상태에서 읽기를 지원하기 위함
//InputStream의 경우에도 blocking 연산을 수행하고 있음
// 즉 입력이 들어올 때까지 기다린다.

System.out.println("msg: " + reader.readLine());
```

```
// 위에 기재한 blocking과 반대되는 개념을
// non-blocking이라고 하며 비동기 처리와 관계가 깊다.
// 그러나 c나 c++을 쓰거나
// accept하는 thread를 따로 두면되는데 지금은 너무 어려움
// accept용 thread / data 수신용 thread / data 송신용 thread 이런식으로
// 따로 만들어 처리하는 방식이 node.js
// 그냥 기본적인 개념의 흐름만 알아둘 것..
```

```
} catch (IOException e){
    System.out.println("Server Exception: " + e.getMessage());
    e.printStackTrace();
}
```

« client

```
import ...
public class _1st_SocketClientEdu {
    // 접속 시간에 대한 정보를 획득하고자 하는 서비스 이용자
    public static void main(String[] args) {

        String hostname = "192.1"; //gitbash에서 ipconfig -all
        int port = 33333;
        // 포트의 역할: 서비스 번호 (PID랑 같은거 아님 주의)
        // 결국 어떤 서비스에 접근하기 위해서는 IP와 포트를 알아야 함.
        for (int i = 0; i < 10; i++) {
            try {
                // server와 마찬가지로 client쪽에도 통신을 수행할 수 있도록 내 socket을 만듦.
                // client쪽에는 접속하고 싶은 server의 ip(hostname)와 서비스(port)를 인자로 하는 socket만듦.
                // 이 socket이 생성되는 시점은 server에서 accept()했을 때 생김.
                Socket sock = new Socket(hostname, port);

                // 서버에게 전송하기 위한 객체 'out'
                OutputStream out = sock.getOutputStream();
                String str = "Hello Network Programming";
                out.write(str.getBytes());

                //서버에서 송신한 정보 읽는 로직
                InputStream in = sock.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(in));
                String time = reader.readLine();
                System.out.println(time);

            } catch (UnknownHostException e) { //ip주소 잘못 적었을 때의 catch
                System.out.println("Server Not Found: " + e.getMessage());
            } catch (IOException e) { // 동작하다가 error가 났을 때의 catch
                System.out.println("I/O Error: " + e.getMessage());
            }
        }
    }
}
```

```
_1st_SocketServerEdu x _1st_SocketClientEdu x
"C:\Program Files\Java\jdk-16\bin\java.exe"
Server: Listening - 33333
[/192.168.1.100] client connected
.
```

Q. 왜 SERVER쪽에 CLIENT가 송신한 "Hello Network Programming"이 출력이 안
될까요??

< 기타 개념 check >

- 내가 다음을 검색하면 먼저 dacom라우터 쪽으로 가서 중재받고(최단거리 알려줌) daum라우터로 가는 흐름.

```
> daum.net
Non-authoritative answer:
Server: ns.dacom.co.kr
Address: 164.124.101.2

Name: daum.net
Addresses: 203.133.167.16
           203.133.167.81
           211.231.99.80
           211.231.99.17
```

- throws의 개념 :
예외가 발생할 경우 내가 처리하지 않고 자신보다
상위에 있는 다른 method나 class에 책임을 전하는
목적의 code.

```
public class _2nd_BankLockTest {
    public static void main(String[] args) throws InterruptedException {
        Counter counter = new Counter();
        System.out.println("First count: " + counter.getCount());
        Thread adder = new Thread(new Worker(counter, true, 1000));
        adder.start();
    }
}
```