(디지털컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

-23일차 학습 및 질문 노트-

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – Kyeonghwan Lee(이경환)
airtrade7@naver.com

```java
public class SocketServerTest {
    // 누군가 접속하면 접속 시간을 알려주는 서비스를 제공함
    public static void main(String[] args) {
        // 포트의 역할: 서비스 번호
        // 결국 우리가 어떤 서비스에 접근하기 위해서는 무엇을 알아야 한다 ? 아이피와 포트
        int port = Integer.parseInt("33333");

        try {
            // 소켓이란 ?
            // 전기 분야에서 소켓에 전원 코드를 연결하면 전기 제품들이 구동 가능한것과 마찬가지로
            // 프로그래밍 분야에서 소켓이란 다른 컴퓨터와 내 컴퓨터를 연결하는 통로 역할을 한다.
            // 그러니까 통신을 수행할 수 있도록 내 소켓을 만들었음
            ServerSocket servSock = new ServerSocket(port);
            System.out.println("Server: Listening - " + port);

            while (true) {
                // accept() 부분에서 서버는 Blocking(블록킹) 연산을 수행하고 있음
                // 니가 준비될때까지 난 계속 기다린다. (문 두드리면서)

                // Blocking의 반대 개념도 있지 않을까 ?
                // Non-Blocking 이라고 하며 비동기 처리와 관계가 깊음
                // (여기에 있는 sock는 접속한 사용자 소켓임)
                Socket sock = servSock.accept();
                // 접속이 완료되었으면 접속한 클라이언트의 IP를 확인한다.
                System.out.println("[" + sock.getInetAddress() + "] client connected");

                // 클라이언트를 향해 출력할 객체를 생성함(송신)
                // 클라이언트(입력) <---------------------- 서버(출력)
                // 클라이언트(출력) ----------------------> 서버(입력)
                OutputStream out = sock.getOutputStream();
                // PrintWriter에 송신용 객체를 배치함으로써
                // writer.println 으로 구동시키는 것이 전송되게 만들었음
                PrintWriter writer = new PrintWriter(out, true);
                // 현재 시간 정보가 클라이언트에게 전송됨
                writer.println(new Date().toString());

                // 클라이언트로 부터 입력받을 객체를 생성함(수신)
                InputStream in = sock.getInputStream();
                // InputStream을 사용해서 들어오는 객체는 반드시 아래와 같이 읽어야 합니다.
                // InputStreamReader(): InputStream 읽기
                // BufferedReader(): 데이터가 많이 들어오거나 빈번하게 지속적으로 들어올 수 있어
                //                   버퍼를 가진 상태에서 읽기를 지원하기 위함
                BufferedReader reader = new BufferedReader(new InputStreamReader(in));
                System.out.println("msg: " + reader.readLine());
            }
        } catch (IOException e) {
            System.out.println("Server Exception: " + e.getMessage());
            e.printStackTrace();
```

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

public class SocketClientTest {
    // 접속 시간에 대한 정보를 획득하고자 하는 서비스 이용자
    public static void main(String[] args) {
        // 사설망이라 컴퓨터 털릴일 없으니 걱정 no!
        String hostname = "192.168.30.141";
        int port = 33333;

        for (int i = 0; i < 10; i++) {
            try {
                // 클라이언트 자신의 소켓을 생성한다.
                // 생성할 때 나는 서버의 ip 주소(hostname)에 서비스(port)에 접속하고 싶어!
                // 라고 요청하면서 소켓을 만든다.
                Socket sock = new Socket(hostname, port);

                // 서버에게 전송하기 위한 객체를 준비함
                OutputStream out = sock.getOutputStream();
                // 이 내용을 서버에게 송신함
                String str = "Hello Network Programming!!!";
                out.write(str.getBytes());

                // 서버에서 날아온 수신 정보
                InputStream in = sock.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(in));

                String time = reader.readLine();
                System.out.println(time);
            } catch (UnknownHostException e) {
                System.out.println("Server Not Found: " + e.getMessage());
            } catch (IOException e) {
                System.out.println("I/O Error: " + e.getMessage());
            }
        }
    }
}
```

클라이언트/서버

- 컴퓨터간의 관계를 역할로 구분하는 개념
  - ✓ 서버(server) : 서비스를 제공하는 컴퓨터(service provider)
  - ✓ 클라이언트(client) : 서비스를 사용하는 컴퓨터(service user)

- 서버는 크게 서버기반 모델과 P2P 모델로 나누어짐

서버기반 모델

- 안정적인 서비스의 제공이 가능
- 공유 데이터의 관리와 보안이 용이
- 서버구축비용과 관리비용 발생

P2P 모델

- 서버구축 및 운용비용 절감
- 자원의 활용율 극대화 가능
- 자원의 관리가 어려움
- 보안이 취약

소켓 통신

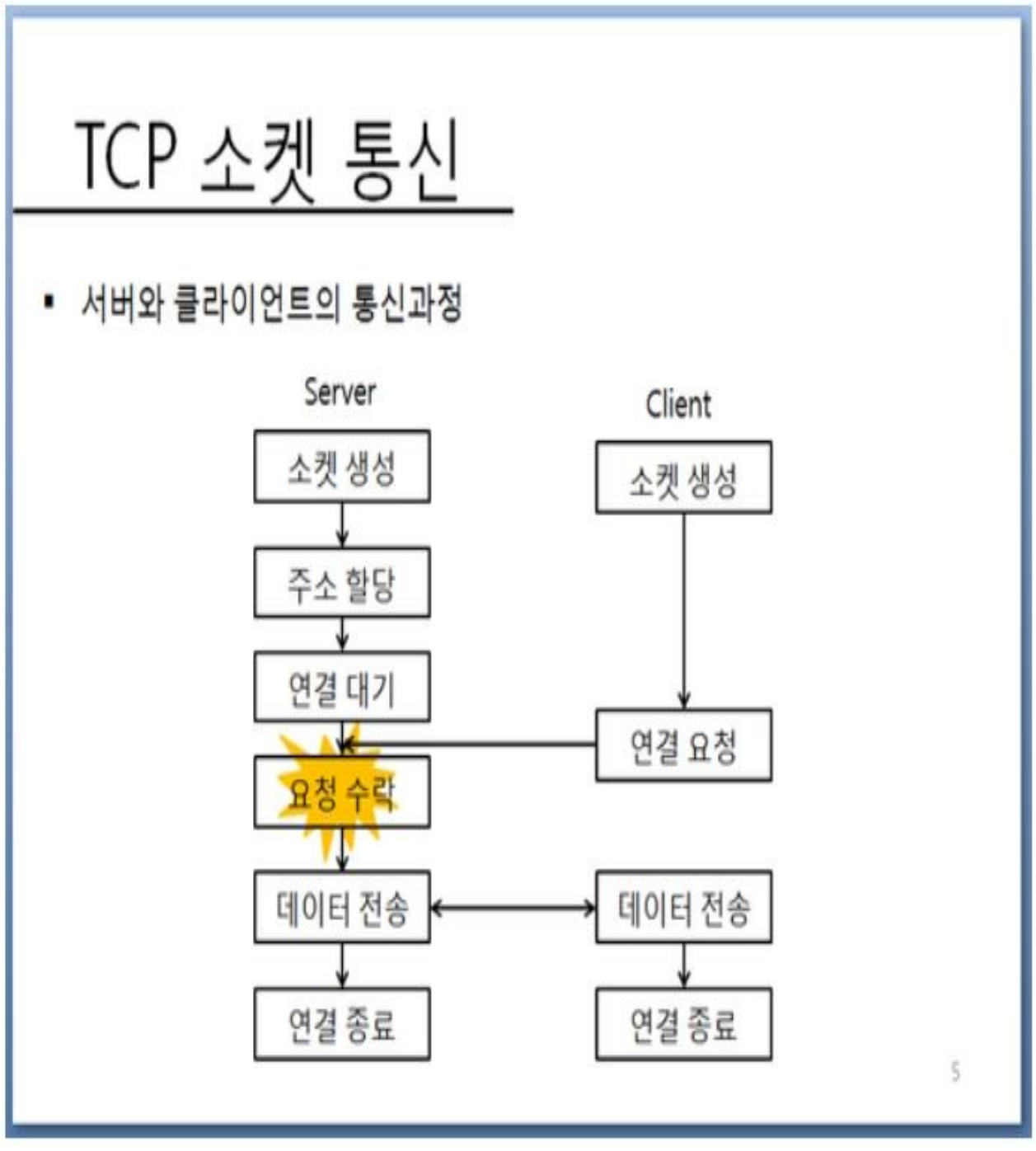
- 사용자가 네트워크에 접근할 수 있는 인터페이스를 제공
- 소켓이란 프로세스간의 통신에 사용되는 양쪽 끝단(endpoint)을 의미
- 자바에서는 java.net패키지를 통해 소켓 프로그래밍 지원
- 사용되는 프로토콜에 따라 다른 종류의 소켓을 제공

TCP

- 연결기반
- 데이터의 경계를 구분안함
- 신뢰성 있는 데이터 전송
- 데이터의 전송순서 보장
- 데이터의 수신여부를 확인

UDP

- 비연결기반
- 데이터의 경계를 구분
- 신뢰성 없는 데이터 전송
- 데이터의 전송순서가 바뀔 수 있음
- 데이터의 수신여부를 확인안함
- 패킷을 관리해주어야 함



소켓 통신의 통신 과정

1. 서버 프로그램에서는 서버 소켓을 사용해서 서버 컴퓨터의 특정 포트에서 클라이언트의 연결 요청을 처리할 준비를 한다

2. 클라이언트 프로그램은 접속할 서버의 IP주소와 포트 정보를 가지고 소켓을 생성해서 서버에 연결을 요청한다.

3. 서버 소켓은 클라이언트의 연결 요청을 받으면 서버에 새로운 소켓을 생성해서 클라이언트의 소켓과 연결되도록 한다.

4. 이제 클라이언트의 소켓과 새로 생성된 서버의 소켓은 서버 소켓과 관계없이 일대일 통신을 한다