디지털컨버전스 기반 UI/UX Front 전문 개발자 양성과정

A조 박재민

2021-01-19(SocketClient / Server)복습

*IP주소

-IP주소에는 사설과 공인 IP로 구성 되어 있다.

1)사설 IP

-192.168 로 시작하는 주소를 가짐

-특정한 지역에 있는 로컬망을 위한 주소

(로컬망이란? = 외부로 나갈 순 없지만 내부에서 통신 활용이 가능)

2)공인 IP

-공인 IP가 있어야 인터넷 접속이 가능하다.

근데 집이나 학원은 사설 IP인데 어떻게 접속하나요? —> 공유기랑, 스위치 혹은 라우터로 공인 IP로 변환하여 인터넷 접속 가능

공유기 - NAT 프로토콜 사용하여 사설 IP를 공인IP로 변환 / 스위치는 라우터에게만 토스만하는 것? 라우터는 토스 받은 IP주소를 보고 내가 어디를 가야하는 지 알 수 있다는 말의 정확한 이해가 필요하다.

2021-01-19(SocketClient / Server)복습

*소켓(Socket)

-통신에 필요한 정보를 담고 있는 객체. 이 객체를 이용하여 통신을 수행

-소켓 생성에 필요한 값

1)인터넷 주소(IP Address, 도메인주소)

ex)Socket socket = new Socket(“localhost”,8888);

localhost=도메인 주소

2) 포트번호(통신하는 프로그램을 구분하기 위한 식별 번호

8888 = 포트번호

*클라이언트(Client) : 접속을 요청하는 쪽

*서버(Server) : 접속을 받아 들이는 쪽

*포트번호

-OS가 구분할 수 있도록 통신프로그램에 부여되는 식별번호

-클라이언트 쪽 프로그램은 OS가 포트번호를 자동으로 부여한다.

-서버 쪽은 프로그램 스스로가 지정한다.

단, 사용중인 포트 번호를 중복 사용 할 수 없다.
기존에 이용되는 포트번호도 피해야한다.

ex) `ssh: 22, http: 80, ftp: 20, 21, telnet: 23 https: 443`

2021-01-19(SocketClient / Server)복습

*대기열

-클라이언트가 서버에 접속을 요청 했을 때, 그 클라이언트의 연결 정보를 저장한 목록이다.

-대기열이 꽉 차 있을 경우, 클라이언트의 연결 요청을 거절한다.

*소켓의 종류

1)서버 소켓

-서버측 프로그램을 작성할 때 사용한다.

-ServerSocket 클래스를 이용하여 만든다.

-중복되지 않도록 포트 번호를 지정한다.

(네트워크 포트는 16비트 0~65535까지 지정 가능 / 단, 고정된 포트를 사용하면 안됨)

*소켓의 종류

2)클라이언트 소켓

-클라이언트측 프로그램을 작성할 때 사용한다.

-Socket 클래스를 이용하여 만든다.

-접속하려는 상대편 인터넷주소(IP 주소, 도메인 주소)를 지정

-접속하려는 상대편의 포트 번호를 지정한다.

*소켓 코드 작성 시 클라이언트 소켓 코드 부터 작성 하는 것이 편하다.

2021-01-19(SocketServer)
코드해석 및 질문

```java
public static void main(String[] args) {
    // 문자열을 숫자로 나타내는 기법(Integer.parseInt)
    int port = Integer.parseInt( s: "33333");

    try {
        // try-catch 문 try - 예외가 발생할 가능성이 있는 문장들을 넣는다.
        // try-catch 문 은 문장이 하나뿐이어도 괄호를 생략할 수 없다.
        // 소켓이란? 네트워킹을 할 수 있는 클래스 객체
        // 서버 소켓 생성시 서비스 번호를 부여해야 하는데
        // 이 서비스 번호로 port(33333)를 설정했다.
        // 네트워크 포트가 16비트를 사용하므로 제한은 0~65535
        // (단, 고정된 포드를 사용하면 안됨)
        ServerSocket servSock = new ServerSocket(port);

        System.out.println("Server: Listening - " + port);
```

```java
while (true) {
    // accept()의 경우 클라이언트가 접속을 요청했는지 체크해서
    // 만약 요청 있었다면 요청을 승인한다.
    // (accept()는 블로킹 연산이다)
    // Q. 블로킹 연산이란 무엇인가?

    // 결국 sock 클라이언트 소켓을 의미한다.
    // 그래서 좀 더 가독성이 좋은 코드는 Socket clntSock으로 작성하면 좋을 듯
    Socket sock = servSock.accept();
    //클라이언트와 통신할 수 있게 양쪽 소켓 연결(서로 동기화)
    //ex) 전화 왔을 때 통화하기 슬라이드가 accept()라고 보면 됨
```

2021-01-19(SocketServer)
코드해석 및 질문

```
} catch (IOException e) {
    // IOException - 예외객체 / e - 참조 변수
    // 입출력 예외가 발생했을 경우 이를 처리하기 위한 문장을 적는다.
    // -> 오류의 사유 파악이 중요 Run을 가동하여 나오는 오류메시지를 확인후 입력하여 예외처리
    // 일치하는 catch블럭이 없으면 예외는 처리 안됨
    // cf) Exception(모든 예외의 최고 조상)이 선언된 catch블럭은 모든 예외 처리 가능하며
    // 여러 catch블럭이 있을 경우 마지막 catch블럭에 위치해야한다.


    // IOException은 입출력 예외처리로
    // I/O 예외가 발생하면 무엇인가 잘못되었음을 감지하고
    // 어디가 잘못되었는지 출력하도록 구성된다.
    // ex) 통신중에 갑자기 네트워크 불안정으로 통신이 끊기면
    //      catch가 I/O 예외를 감지하고 아래 코드가 동작하게 된다.
    // 아래코드는 에러 메시지를 출력하는 코드로 언제나 동일하게 작성하면 됨.

    // getMessage() - 발생한 예외클래스의 저장된 메시지를 얻을 수 있음
    System.out.println("Server Exception: " + e.getMessage());

    // 예외 발생 당시의 콜(호출) 스택(메서드 호출)이 어떤식으로 이뤄졌는지 상태(메서드의 정보와 예외 메시지)를 보여줌(화면에 출력)
    // 디버깅을 위해서 많이 사용 하는 편
    e.printStackTrace();
```

2021-01-19(SocketClient)
코드해석 및 질문

```
String str = "Hello Network Programming";
// 위의 문자열을 바이트 단위로 쪼개서 서버(SocketServerTest)로 전송(출력할 수 있게) - 연결이 된다
out.write(str.getBytes());

// 서버의 입력을 생성(수신 준비)
// Q. 이 부분도 서버처럼 소켓을 통해 읽을 땐 무조건 아래 형식으로 고정해서 사용 해야하는 건가?
InputStream in = sock.getInputStream();
BufferedReader reader = new BufferedReader(new InputStreamReader(in));

// 서버가 보낸 내용을 time에 저장하고 출력한다.
String time = reader.readLine();
System.out.println(time);
```