디지털컨버전스 기반 UI UX Front 전문 개발자 양성과정

Lecturer silenc3502 Sanghoon Lee (이상훈)
gcccompil3r@gmail.com

Student may_hz HyeonJeong Choi (최현정)
hyeonjeong9943@gmail.com

SocketServer

Date 2021_01_19

```
public static void main(String[] args) {
  int port = Integer.parseInt("33333");

  try {
      ServerSocket servSock =
                  new ServerSocket(port);
      System.out.println(
                  "Server: Listening - " + port);
      while(true) {
        Socket sock = clntSock.accept();
        // 소켓 접속 요청이 올때까지 대기!
        System.out.println(
                  "[" + sock.getInetAddress() +
                              "] client connected"
        ); // 접속이 되었을때 출력된다

        OutputStream out = sock.getOutputStream();
        // 응답을 위한 스트림을 얻어온다
        PrintWriter writer = new PrintWriter(out, true);
        // Println의 결과를 out으로 전송한다
        writer.println(new Date().toString());
        // 현재날짜를 출력한다.
        InputStream in = sock.getInputStream();
        // 입력을 위한 스트림을 얻어온다
        BufferedReader reader =
                  new BufferedReader(new InputStreamReader(in));
        // 클라이언트가 보낸 결과를 읽는다.
        System.out.println("msg: " + reader.readLine());
        // 결과를 읽으면 내용을 출력할 수 있게 된다
      }
  }
  catch (IOException e) { // 예외 처리가 발생했을때
      System.out.println( // 에러 메시지를 출력하는 코드
                  "Server Exception: " + e.getMessage());

      e.printStackTrace();
      // 콜 스택(메서드 호출)이 어떤식으로 이뤄졌는지 상태를 보여줌
  }
}
```

문자열을 int값으로 변환해주는 메소드
port번호 33333을 설정했다.

try ~ catch

```
try {
    예외가 발생할 수 있는 코드
} catch ( 예외 클래스 e ) {
    예외처리
}
```

OutputStream / InputStream

바이트 단위 입출력을 위한 최상위 스트림클래스

PrintWriter / BufferedReader

문자 단위 입출력을 위한 하위 스트림 클래스

SocketClient

Date 2021_01_19

```
  try {
      Socket sock = new Socket(hostname, port);
      // 소켓을 생성한다 ( ip 주소, port 번호)
      OutputStream out = sock.getOutputStream();
      // 응답을 위한 스트림을 얻어온다
      String str = "Hello Network Programming";
      out.write(str.getBytes());
      // 응답하면 str을 출력한다
      InputStream in = sock.getInputStream();
      BufferedReader reader =
            new BufferedReader(new InputStreamReader(in));
      // 입력 스트림을 얻어온다 그리고 결과를 읽는다
      String time = reader.readLine();
      System.out.println(time);
      // 결과물 출력
  } catch (UnknownHostException e) { // 예외 처리 1)
      System.out.println(
      "Server Not Found: " + e.getMessage());
      // 서버를 찾을 수 없습니다
  } catch (IOException e) { // 예외 처리 2)
      System.out.println(
      "I/O Error: " + e.getMessage());
      // 입출력 에러
  }
}
```

<< 예외처리를 할때에 throw를 해야하는 이유 >>

throw를 하지 않을 경우 Main에서는 Exception을 전달받지 못하여
개발자가 예외를 인지 못하는 경우가 발생하기 때문이다