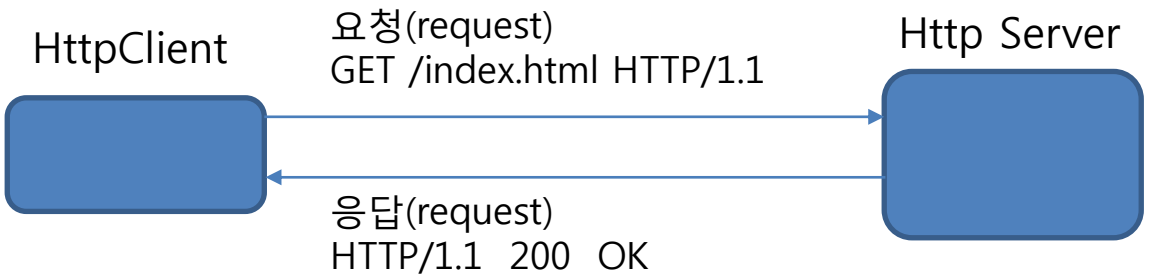
HTTP 통신

HTTP 클라이언트와 서버

❖ Http 프로토콜

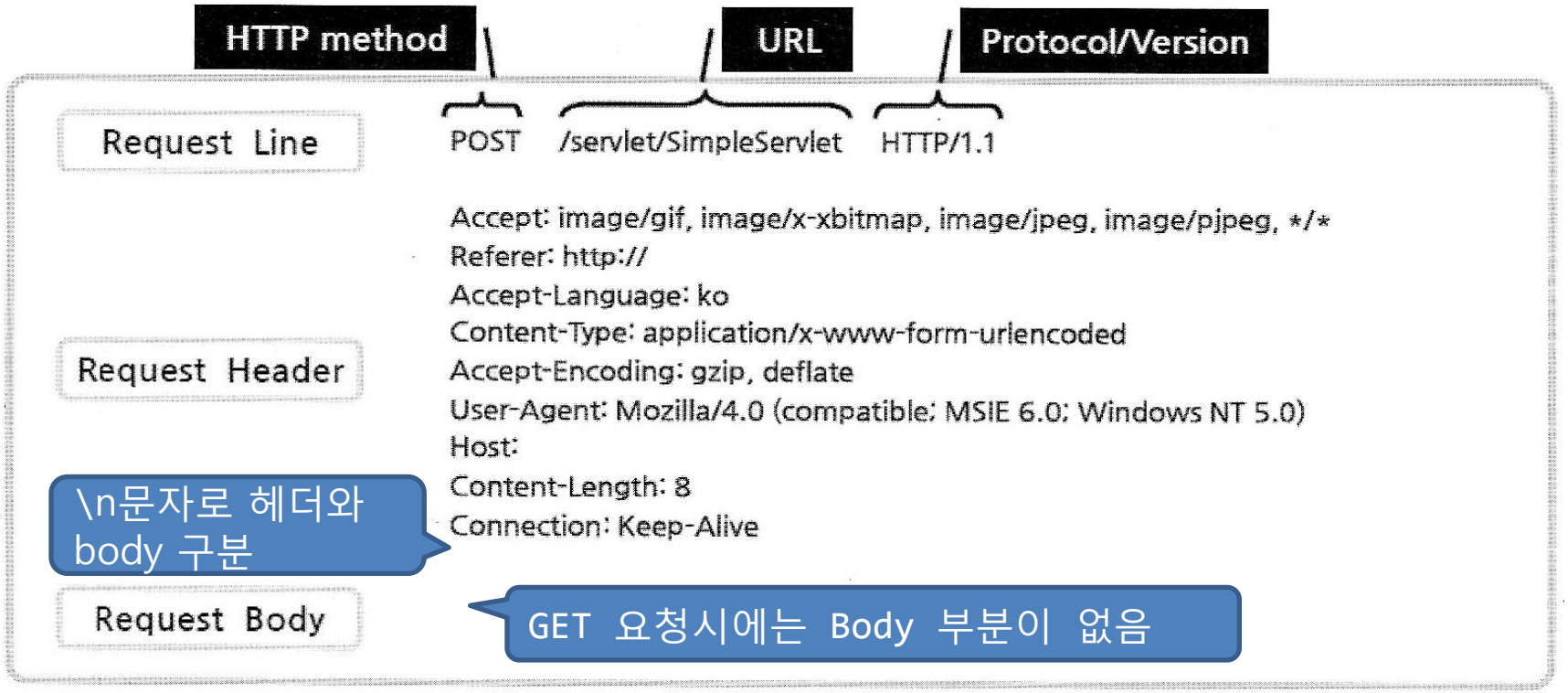
- ○ 주요 특징
  - ▪ 문자 기반 프로토콜
  - ▪ 디폴트로 80 포트 사용
  - ▪ 클라이언트가 서버로 요청을 보내면 서버는 요청 내용을 클라이언트로 응답 후 접속 해제
    → 서버 측에서 접속에 대한 상태 유지 하지 않음 : stateless
    - 이후 접속 시에 누구인지, 이전에 어떤 작업을 했는지에 대한 상태 정보 없음



HTTP 클라이언트와 서버

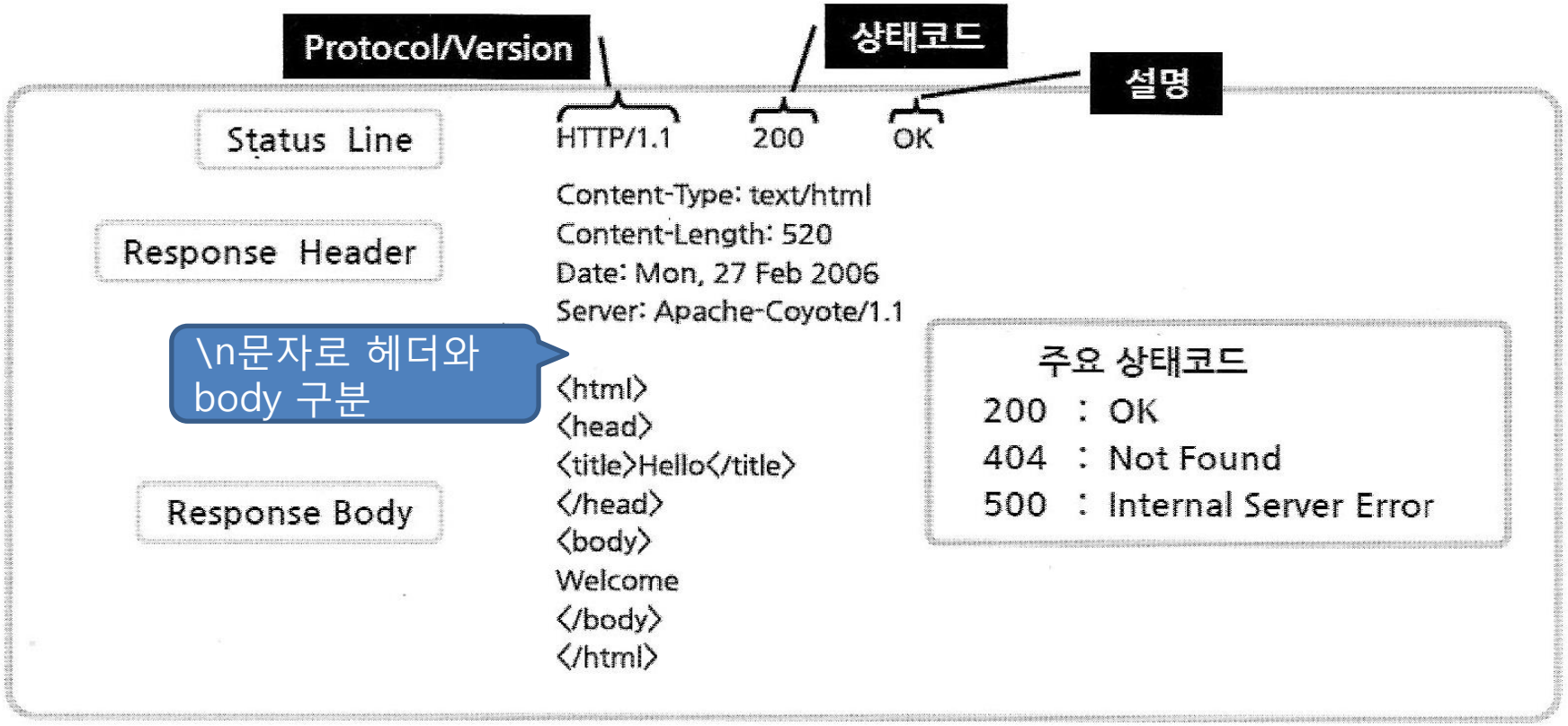
❖ HTTP Request

- ㅇ 요청 라인(Request Line)
  - HTTP 메서드 방식 및 요청 URL과 프로토콜 정보
- ㅇ 요청 헤더(Request Header)
  - 웹 브라우저 정보, 언어, 인코딩 방식, 요청 서버 정보 등 추가 정보
- ㅇ 요청 본체(Request Body)
  - 요청에 필요한 내용



❖ HTTP Response

- ㅇ 상태 라인
  - ▪ 응답 상태 코드 및 프로토콜 정보
- ㅇ 응답 헤더
  - ▪ 응답 처리 날짜, 인코딩 방식, 요청 서버 정보 등과 같은 추가 정보
- ㅇ 응답 본체
  - ▪ 응답에 필요한 내용. 일반적으로 HTML 문서



HTTP 클라이언트와 서버

❖ Http 분석

- 패키지 명 : http
- Http 프로토콜 클래스
  - HttpHeader
  - HttpRequest
  - HttpResponse
- Http 클라이언트/서버 프로그램
  - HttpClient
  - HttpServer
- GET, POST 확인용 웹 페이지
  - login.html

❖ HttpHeader 클래스

- http 메시지의 헤더 부분 처리

```
public class HttpHeader {
    // 헤더
    HashMap<String, String> headerMap = new HashMap<>();

    // 헤더 읽기
    public void readHeader(BufferedReader reader) throws Exception {
        String header;
        while(true) {
            header = reader.readLine();  // 헤더 읽기
            if(header.equals("")) {   // 헤더가 끝났다면 ("\n"만 있는 경우)
                System.out.println();
                break;
            }
            String []entry = header.split(":");
            headerMap.put(entry[0], entry[1].trim());
            System.out.println(header);
        }
    }
}
```

❖ HttpRequest 클래스

- http 요청 메시지 처리

```
public class HttpRequest {
    String method;          // 요청 Method
    String requestPath;     // 요청 URL
    HttpHeader header = new HttpHeader();  // 헤더

    BufferedReader reader; // 메시지 수신용
    PrintWriter  writer;   // 메시지 전송용

    public HttpRequest(InputStream is, OutputStream out) {
        reader = new BufferedReader(new InputStreamReader(is));
        writer = new PrintWriter(out);
    }

    public String getMethod() {
        return method;
    }

    public String getRequestPath() {
        return requestPath;
    }
```

❖ HttpRequest 클래스

- http 요청 메시지 처리

```
// Http 클라이언트용 메서드 ----------------------------------------
// 요청 전송
public void sendRequest(String method, String page) {
    writer.print(method + " " + page + " HTTP/1.1\n\n");
    writer.flush();
}

// Http 서버용 메서드 ----------------------------------------------
// 헤더 읽기
public void readHeader() throws Exception {
    header.readHeader(reader);
}
```

❖ HttpRequest 클래스

- http 요청 메시지 처리

```
    // 요청 읽기
    public void readRequest() throws Exception {
        String requestStr = reader.readLine();

        String []buf = requestStr.split(" ");  // 공백을 기준으로 분리
        method = buf[0];    // 메서드 부분
        requestPath = buf[1];  // 요청 경로 부분

        System.out.println(requestStr);
    }

    // request body 읽기
    public void readBody() throws Exception {
        if(method.toUpperCase().equals("GET")) // GET인 경우 BODY 없음
            return;
        int b;
        do{
            b = reader.read();
            System.out.print((char)b);
        } while(b!=-1);
    }
}
```

❖ HttpResponse 클래스

- http 응답 메시지 처리

```
public class HttpResponse {
   String status;  // 응답 상태
   String desc;    // 상태 설명
   HttpHeader header = new HttpHeader();  // 헤더

   BufferedReader reader;
   PrintWriter  writer;

   public HttpResponse(InputStream is, OutputStream out) {
      reader = new BufferedReader(new InputStreamReader(is));
      writer = new PrintWriter(out);
   }
```

❖ HttpResponse 클래스

- http 응답 메시지 처리

```
// Http 클라이언트용 메시지 ----------------------------------------
// 응답 상태 수신
public void readStatus() throws Exception {
    String line = reader.readLine()
    String []buf = line.split(" ");
    status = buf[1];    // 상태 코드 부분
    desc = buf[2];      // 상태 설명
    System.out.println(line);
}

public void readHeader() throws Exception {
    header.readHeader(reader);
}

// body 수신
public void readBody() throws Exception {
    int b;
    do{
        b = reader.read();
        System.out.print((char)b);
    } while(b!=-1);
}
```

❖ HttpResponse 클래스

- http 응답 메시지 처리

```
    // Http 서버용 메서드 ----------------------------------------------
    // 응답 코드 전송
    public void sendStatus(String status, String desc) {
        String response = "HTTP/1.1 " + status + " " + desc;
        writer.println(response);
    }

    // 응답 헤더 전송
    public void sendHeader() {
        writer.println("Content-Type: text/html");
        writer.println();  // 헤더의 끝
    }

    // body 보내기
    public void sendBody(String body) {
        writer.println(body);
        writer.flush();
    }
}
```

❖ HttpClient 클래스

- http 요청 메시지를 서버에 전송

```
public class HttpClient {
    public static void main(String[] args) {
        // 사용자로부터 요청 URL 입력
        Scanner s = new Scanner(System.in);
        System.out.print("서버> ");
        String server= s.nextLine();

        try(Socket socket = new Socket(server, 80)) {

            System.out.print("요청파일> ");
            String requestPath = s.nextLine();

            // 요청 전송
            HttpRequest request = new HttpRequest(socket.getInputStream(),
                                                  socket.getOutputStream());
            request.sendRequest("GET", requestPath);  // 요청 보내기
```

❖ HttpClient 클래스

- http 요청 메시지를 서버에 전송

```
            // 서버 응답 수신
            HttpResponse response = new HttpResponse(
                        socket.getInputStream(), socket.getOutputStream());

            response.readStatus(); // 응답 받기
            response.readHeader(); // 헤더 받기
            response.readBody();   // body 부분 읽기
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

}
```

❖ HttpServer 클래스

- http 요청 메시지를 수신하고 클리이언트로 http 응답 전송

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

        try(ServerSocket server = new ServerSocket(80)) {
            Socket socket = server.accept();

            // 클라이언트로부터 요청 수신
            HttpRequest request = new HttpRequest(socket.getInputStream(),
                                                  socket.getOutputStream());

            request.readRequest();     // METHOD, 요청 경로 수신
            request.readHeader();      // 헤더 수신
            request.readBody();        // body 수신 – GET일 경우 내용 없음
```

❖ HttpServer 클래스

- http 요청 메시지를 수신하고 클리이언트로 http 응답 전송

```
            // 클라이언트로 응답 전송
            HttpResponse response = new HttpResponse(
                        socket.getInputStream(), socket.getOutputStream());

            response.sendStatus("200", "OK");   // 상태 전송
            response.sendHeader();              // 헤더 전송
            response.sendBody("수신완료");      // body 부분 전송

        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

❖ 테스트

- HttpServer 실행
  - 웹 브라우저로 localhost/test/index.html 접속

- HttpClient 실행
  - 접속 서버 입력 : 예) www.naver.com
  - 요청 파일 입력 : 예) / 또는 페이지

❖ Html 페이지

```
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
<body>

<form action="http://localhost/index.jsp" method="GET" >
        User ID <input type="text" name="id"/><br/>
        Password <input type="password" name="password"/> <br/>
        <input type="submit"> <br/>
</form>

</body>
</html>
```

GET 또는 POST

❖ GET/POST 확인

- HttpServer 실행
- 앞에서 만든 html 파일의 확인버튼 클릭