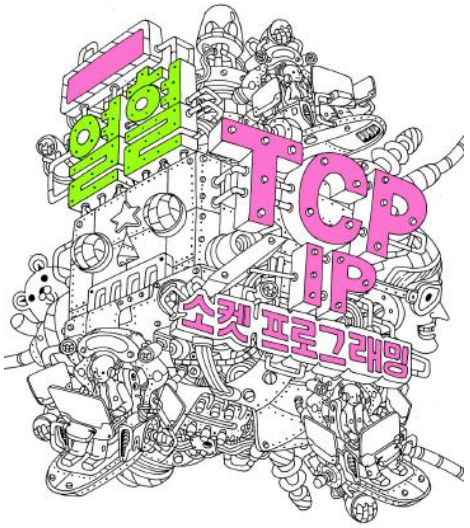
윤성우의 열혈 TCP/IP 소켓 프로그래밍

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

Chapter 24. HTTP 서버 제작하기



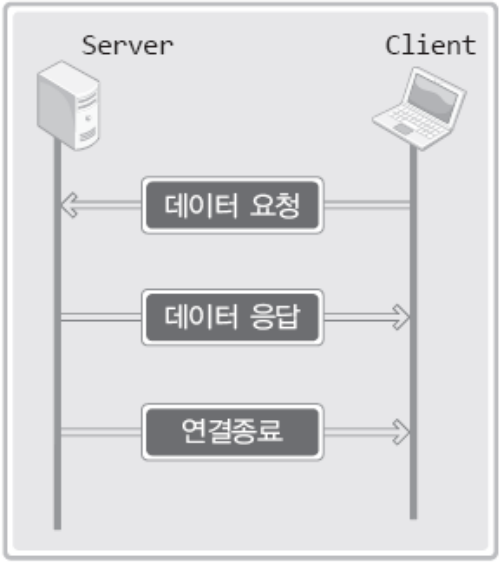
Chapter 24-1. HTTP(Hypertext Transfer Protocol)의 개요

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

웹(Web) 서버의 이해와 HTTP

웹 서버의 기능

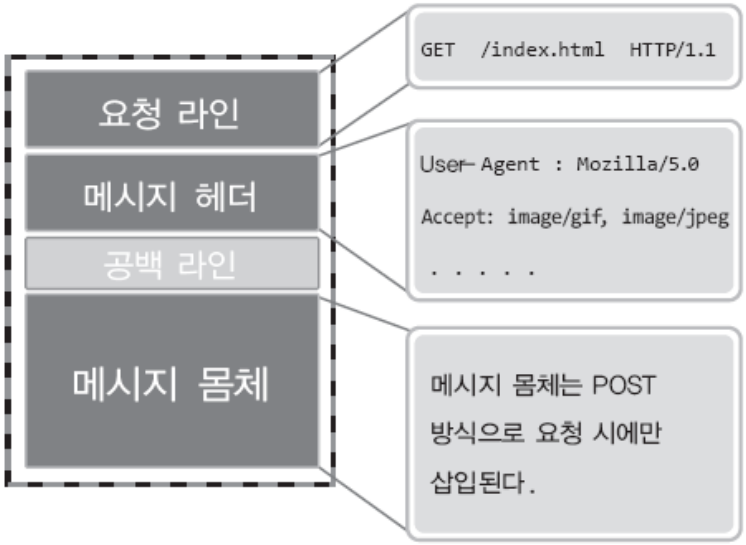
- HTTP 프로토콜을 기반으로 웹 페이지에 해당하는 파일을 클라이언트에게 전송하는 역할의 서버
- HTTP 프로토콜은 Hypertext의 전송을 목적으로 설계된 어플리케이션 레벨의 프로토콜
- Hyptertext란 마우스의 클릭을 통해서 이동이 가능한 일반적으로 HTML로 이뤄진 텍스트를 뜻함



HTTP의 기본적인 통신 방식

상태가 존재하지 않는 stateless 프로토콜

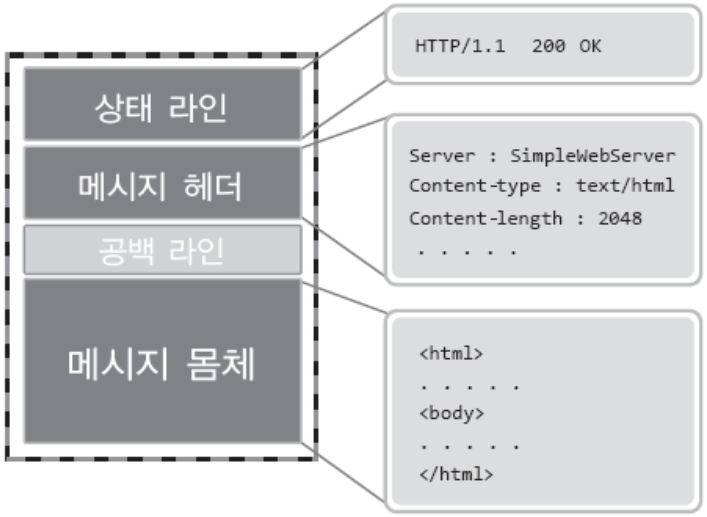
HTTP의 요청과 응답 메시지



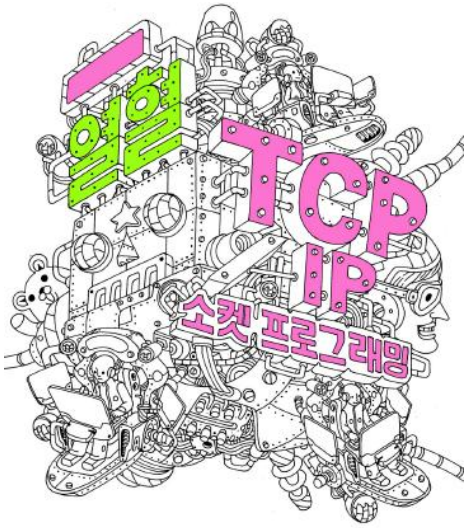
HTTP의 요청 방식

- 200 OK 요청이 성공적으로 처리되었다.
- 404 Not Found 요청한 파일이 존재하지 않는다!
- 400 Bad Request 요청방식이 잘못되었으니 확인해봐라!

요청에 대한 상태코드



HTTP의 응답방식



Chapter 24-2. 매우 간단한 웹 서버의 구현

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

윈도우 기반의 멀티쓰레드 웹 서버 구현1

```
/* 요청 및 응답 */
while(1)
{
    clntAdrSize=sizeof(clntAdr);
    hClntSock=accept(hServSock, (SOCKADDR*)&clntAdr, &clntAdrSize);
    printf("Connection Request : %s:%d\n",
    inet_ntoa(clntAdr.sin_addr), ntohs(clntAdr.sin_port));
    hThread=(HANDLE)_beginthreadex(
        NULL, 0, RequestHandler, (void*)hClntSock, 0, (unsigned *)&dwThreadID);
}
```

```
unsigned WINAPI RequestHandler(void *arg)
{
    SOCKET hClntSock=(SOCKET)arg;
    char buf[BUF_SIZE];
    char method[BUF_SMALL];
    char ct[BUF_SMALL];
    char fileName[BUF_SMALL];
    recv(hClntSock, buf, BUF_SIZE, 0);
    if(strstr(buf, "HTTP/")==NULL)     // HTTP에 의한 요청인지 확인
    {
        SendErrorMSG(hClntSock);
        closesocket(hClntSock);
        return 1;
    }
    strcpy(method, strtok(buf, " /"));
    if(strcmp(method, "GET"))      // GET 방식 요청인지 확인
        SendErrorMSG(hClntSock);
    strcpy(fileName, strtok(NULL, " /"));      // 요청 파일이름 확인
    strcpy(ct, ContentType(fileName));     // Content-type 확인
    SendData(hClntSock, ct, fileName);     // 응 답
    return 0;
}
```

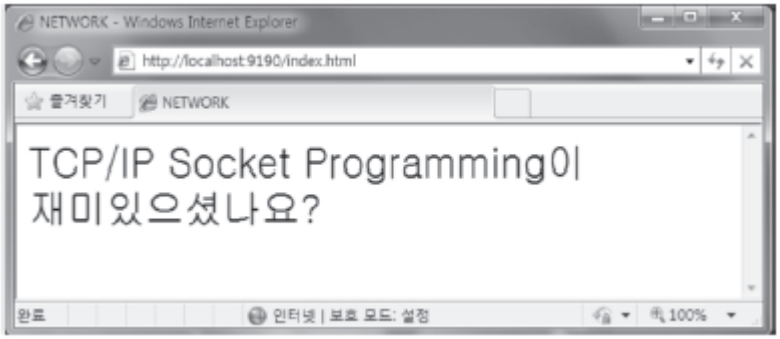
윈도우 기반의 멀티쓰레드 웹 서버 구현2

```
void SendData(SOCKET sock, char* ct, char* fileName)
{
    char protocol[]="HTTP/1.0 200 OK\r\n";
    char servName[]="Server:simple web server\r\n";
    char cntLen[]="Content-length:2048\r\n";
    char cntType[BUF_SMALL];
    char buf[BUF_SIZE];
    FILE* sendFile;
    sprintf(cntType, "Content-type:%s\r\n\r\n", ct);
    if((sendFile=fopen(fileName, "r"))==NULL)
    {
        SendErrorMSG(sock);
        return;
    }
    /* 헤더 정보 전송 */
    send(sock, protocol, strlen(protocol), 0);
    send(sock, servName, strlen(servName), 0);
    send(sock, cntLen, strlen(cntLen), 0);
    send(sock, cntType, strlen(cntType), 0);
    /* 요청 데이터 전송 */
    while(fgets(buf, BUF_SIZE, sendFile)!=NULL)
        send(sock, buf, strlen(buf), 0);
    closesocket(sock);    // HTTP 프로토콜에 의해서 응답 후 종료
}
```

```
char* ContentType(char* file)    // Content-Type 구분
{
    char extension[BUF_SMALL];
    char fileName[BUF_SMALL];
    strcpy(fileName, file);
    strtok(fileName, ".");
    strcpy(extension, strtok(NULL, "."));
    if(!strcmp(extension, "html")||!strcmp(extension, "htm"))
        return "text/html";
    else
        return "text/plain";
}
```

윈도우 기반의 멀티쓰레드 웹 서버 구현3

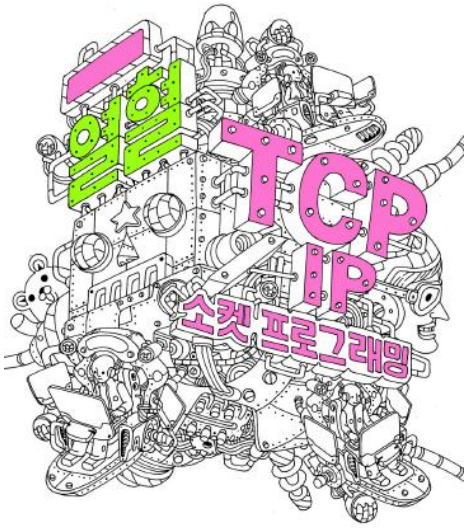
```
void SendErrorMSG(SOCKET sock)   // 오류 발생시 메시지 전달
{
    char protocol[]="HTTP/1.0 400 Bad Request\r\n";
    char servName[]="Server:simple web server\r\n";
    char cntLen[]="Content-length:2048\r\n";
    char cntType[]="Content-type:text/html\r\n\r\n";
    char content[]="<html><head><title>NETWORK</title></head>"
        "<body><font size=+5><br>오류 발생! 요청 파일명 및 요청 방식 확인!"
        "</font></body></html>";
    send(sock, protocol, strlen(protocol), 0);
    send(sock, servName, strlen(servName), 0);
    send(sock, cntLen, strlen(cntLen), 0);
    send(sock, cntType, strlen(cntType), 0);
    send(sock, content, strlen(content), 0);
    closesocket(sock);
}
```

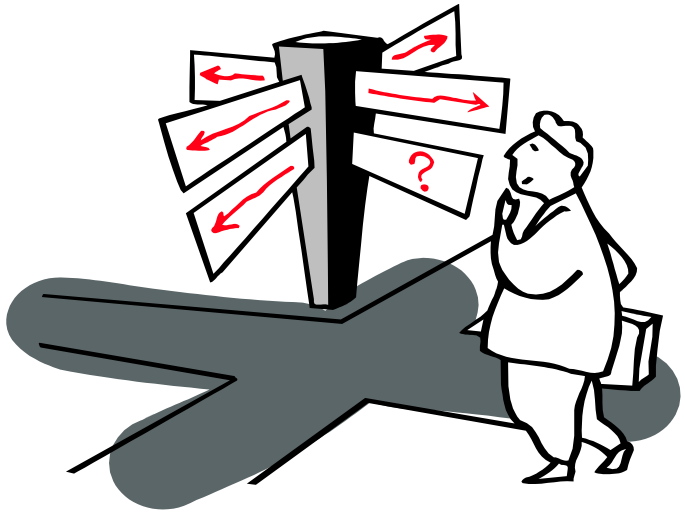


리눅스 기반의 멀티쓰레드 웹 서버 구현

윈도우 기반의 웹 서버를 단순히 리눅스 기반으로 옮겼으며,

입출력의 과정에서 표준 입출력 함수를 사용하였다.





Chapter 24가 끝났습니다. 질문 있으신지요?