목차

**HTTP & HTTPS 1**

개념 및 특징2

공개키 암호화 과정 3

**프로토콜 설명 4**

통신 프로토콜의 기본 구성 요소5

통신 프로토콜의 주요 기능5

통신 프로토콜의 종류5

**URL & URI 7**

개념 및 특징5

**POST메서드 & GET메서드 8**

**HTTP 패킷 9**

**GET방식 vs POST 방식**

**각 메서드 전송 형태 4**

**프로토콜 헤더 포멧 4**

L2 : Ethenet Header5

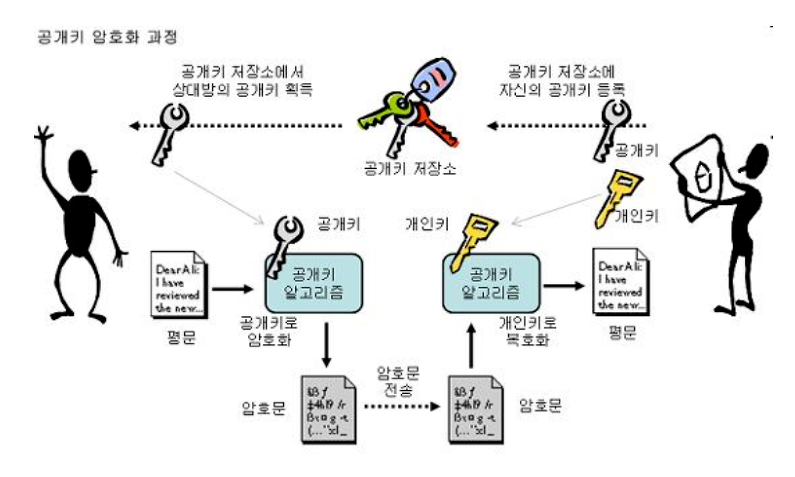
L3 : IP Header5

L4 : TCP/UDP Header5

**HTTP & HTTPS**

|  |  |  |
| --- | --- | --- |
|  | **HTTP(HyperText Tranfer Protocol)** | **HTTPS(HyperText Tranfer Protocol Secure Socket)** |
| 개념 | -웹 브라우저(Client)와 웹서버(Server)간에 메시지 교환 프로토콜 | -같은 통신 프로토콜, HTTP와 거의 흡사 |
| 교환 방식 | 단순 텍스트 |  |
| 특징 | -만약 누군가 주고받는 네트워크 패킷을 가로채 본다면 메시지 내용이 그대로 보이게 되므로 보안 취약 | -HTTP의 취약점인 보안 취약을 보완하기 위해 주고받는 모든 메시지를 암호화  -암호화 방식 : 대칭/비대칭  -공유된 세션키(대칭키)를 모르는 상황에서 암호를 풀기는 불가능  -SSL(Secure Sockey Layer)를 이용하여 정보를 암호화.  -HTTP 메시지(Text)를 암호화하는 것  -\*공개키 암호화 방식 |

\*공개키 암호화 과정



<https://jeong-pro.tistory.com/89>

**프로토콜 설명**

: 서로 다른 시스템(System)에 존재하는 개체(Entity)간의 원활한 통신을 위한 SW적 HW적 약속이나 규칙 및 규약을 말한다.

(System : 터미널,컴퓨터 , Entity : 데이터베이스,사용자 프로그램, 파일 전송 프로그램)

* 통신 프로토콜의 기본 구성 요소
  + 구문(Systax) : 데이터 형식, 부호화 신호레벨 (Signal Level) 등의 요쇼
  + 의미(Semantics) : 전송 제어 및 오류 처리를 위한 정보 등을 규정
  + 시간(Timing) : 두 개체 간의 통신 속도를 조정하거나 메시지의 전송 및 순서에 대한 특성을 가르킨다.
* 통신 프로토콜의 주요 기능
  + 단편화와 재결합
  + 캡슐화
  + 캡슐의 주요 제어 정보
  + 연결 제어
  + 흐름 제어
  + 오류 제어
  + 동기화
* 통신 프로토콜의 주요기능
  + 순서제어
  + 주소 지정
  + 다중화
  + 경로 선택
* 통신 프로토콜의 종류
  + ARPANET : 현재 사용 중인 인터넷의 모체
  + SNA : 네트워크상의 통신 규약
  + X.25 : 데이터 단말 장치 (DTE) 와 데이터 회선 종단 장치 (DCE) 간의 인터페이스로 물리 계층, 데이터 링크 계층 패킷 계층들에 대한 기능으로 구성, 흐름 및 오류 제어 기능을 제공
  + OSI : 개방형 시스템 간 상호 접속 통신의 종단에서 이루어지는 기능을 7계층으로 정의
  + ISDN : 발신 가입자로부터 수신자까지의 모든 전송, 교환 과정이 디지털 방식으로 처리되며 음성과 비음성,영상 등 서비스를 종합적으로 처리하는 종합 정보 통신망
  + 지능망 (IN) : 기존의 통신망에 컴퓨터를 연결하여 새로운 기술을 쉽게 접목할 수 있게한 지능화된 네트워크
  + PPP(Point To Protocal) : 전화선 모뎀을 이용하여 인터넷을 전송하기 위한 프레임 프로토콜로 이전에 사용했던 방식인 SLIP를 개선하여 에러 검출 기능과 복구 기능을 추가한 프로토콜. PPP는 HDLC와 유사한 점은 있지만 HDLC는 비트 위주 프레임, PPP는 문자 위주 프레임.



* 고급 : 사용자의 접근이 가능하고 이해하기 쉬운 계층이다.
* 중급 : 고급과 중급의 연계성 유지 기능으로 OS 수준의 통신 제어 프로그램 계층이다.
* 저급 : 전문적인 기술이 필요한 기계적,전기적, 신호규정 계층이다.
* LAN : 기계적인 설비를 하고 두개의 컴퓨터가 왔다갔다 신호를 하면 LAN=간단한 pc방 이다.
* VAN : 거기에다가 인터넷,통신,사용자App을 설치하면 완전한 pc방

**URL & URI**

|  |  |  |
| --- | --- | --- |
|  | **URL(Uniform Resource Locator)** | **URI(Uniform Resource Identifier)** |
| 개념 | -네트워크 상에서 자원이 어디 있는지를 알려주기 위한 규약이다.  즉, 컴퓨터 네트워크와 검색 매커니즘에서의 위치를 지정하는 웹 리소스에 대한 참조이다. | -인터넷에 있는 자원을 나타내는 유일한 주소  -URI의 존재는 인터넷에서 요구되는 기본조건으로서 인터넷 프로토콜에 항상 붙어 다님. |
| Ex) | - <http://ko.wikipedia.org/>  라는 주소는 서버도 나타내기 때문에 URL이면서 URI이다.  - <http://sunychoi.github.io/java/2015/04/12/java-bigdecimal.html> 라는 주소는  sunychoi.github.io 호스트 주소 하위에 java/2015/04/12 디렉토리아래 java-bigdecimal.html 이라는 자원의 위치를 가리키고 있으므로 URL이면서 URI이다.  - <https://www.google.co.kr/search?q=uri> 라는 주소는 https 프로토콜을 가지고있고 호스트 이름을 가지고 있다. 하지만 그 뒤에  /search?q=uri와 같은 문자열이 붙어 있음 q=rui 는 query string 인 q의 값에 따라 여러가지 결과값을 가져올 수 있다.  URL : <https://www.google.co.kr/search> 여기까지 이고 q=uri라는 식별자가 필요하므로, https://www.google.co.kr/search?q=uri 이 주소는 URI이지만 URL은 아니다. | |
| 특징 | -URL은 웹 주소 뿐만 아니라 컴퓨터 네트워크상의 자원을 모두 나타낼 수 있다.  -주소에 접근하려면 해당 URL에 맞는 프로토콜을 알아야한다.  -FTP 프로토콜인 경우 FTP 클라이언트를 이용, HTTP인 경우 웹 브라우저를 이용해야함, 텔넷인 경우 텔넷 프로그램을 이용하여 접속해야함. | -URL,URN의 상위 개념  -위의 ex로 이해 |

**POST메서드 & GET메서드**

: 웹 서비스 개발에 주로 사용하는 메서드이다. 사용자가 URL을 브라우저 주소창에 작성하고 엔터를 누르면 원하는 웹페이지가 나온다. 사용자는 웹페이지를 보기위해 단순한 일을 한 것 이지만, 특정 웹페이지를 사용자 웹브라우저에게 보여주기 위해서는 내부적인 처리(요청,응답들이 있다.

-**요청** : 클라이언트가 서버에게 웹페이지를 보여달라고 말하는 것.

-**응답** : 서버가 클라이언트에게 요청받은 것에 대한 대답으로, 웹페이지 내용을 표현하기 위해 html문서로 주는 것을 응답이라 부른다.

**HTTP 패킷**

: 클라이언트가 서버로 요청을 했을 때, 보내는 데이터를 HTTP 패킷이라 표현함.

HTTP 프로토콜을 쓰므로, 앞에 HTTP가 붙고 인터넷을 통해 보내는 데이터를 패킷이라 표현하므로, HTTP패킷 이라 부른다. HTTP패킷의 구조는 크게 헤더와 바디로 나뉘어진다.

헤더에는 8가지 HTTP 메서드 방식 중 무엇을 썻는지, 클라이언트의 정보, 브라우저 정보, 접속할 URL 등등 과 같은 클라이언트 정보를 담는다.

바디는 보통 비어있다. 하지만, 특정 데이터를 담아서 서버에게 요청을 보낼 수 있다.

\*8가지 HTTP 메서드 : 1)GET 2)POST 3)HEAD 4)OPTION 5)PUT 6)DELETE 7)TRACE 8)CONNECT

**GET방식 vs POST 방식**

: 두 방식 모두, 서버에 요청을 하는 메서드이다. 클라이언트가 서버에 요청을 할 때, 제공해야 하는 자원이 있다고 하자.

Ex)로그인을 하는 경우 -> 아이디와 패스워드는 클라이언트가 작성한 후, 그 정보를 서버에 요청하여 클라이언트가 작성한 아이디와 패스워드가 올바른 것인지 검사를한다.

|  |  |  |
| --- | --- | --- |
|  | **POST** | **GET** |
| 개념 | WWW에서 사용되는 HTTP가 지원하는 요청 방식이다. | GET방식은 URL에 데이터를 붙여서 보냄 |
| 특징 | -요청 URI(URL)에 폼 입력을 처리하기 위해 구성한 서버측 스크립트(ASP,PHP,JSP) 혹은 CGI 프로그램으로 구성되고 Form Action과 함께 전송됨 이때 헤더 정보에 포함되지 않고 데이터 부분에 요청 정보가 들어가게 된다.  -키 값의 쌍은 ‘&’ 문자로 구분되며 각 키는 ‘=’ 문자의 값과 구분된다. 키와 값들은 둘 다 공백을 ‘+’ 문자로 대체하여 영어숫자가 아닌 그 밖의 모든 문자는 퍼센트 인코딩 처리한다.  -데이터 전송을 기반으로 한 요청 메서드  - BODY에다가 데이터를 넣어서 보낸다.  따라서 헤더필드 중 body를 설명하는 content-type이라는 헤더 필드가 들어가고 어떤 타입인지 명시한다. | -URI(URL)가 가진 정보를 검색하기 위해 서버 측에 요청하는 형태 |
| Ex) | Name: Gareth Wylie  Age: 24  Formula: a + b == 13%!  위의 키 값 쌍은  Name=Gareth+Wylie&Age=24&Formula=a+%2B+b+%3D%3D+13%25%21  다음과 같이 인코딩된다.  다음과 같이 클라이언틔 데이터를 URL 뒤에 붙여서 보낸다.  [컨텐츠 타입]   application/x-www-form-urlencoded  (default)   text/plain   multipart/form-data | [[www.example.com?id=mommoo&pass=1234](http://www.example.com?id=mommoo&pass=1234)] (1)‘?’를 통해서 URL의 끝을 알리면서 데이터 표현의 시작점을 알린다. 데이터는 key와 value 쌍으로 넣어야한다.  Key : id,pass  Value : mommoo,1234  (2)중간의 ‘&’마크는 구분자 이다. 2개이상의 key-value쌍 데이터를 보보낼 때는 &마크로 구분  (3)URL에 붙이므로,HTTP패킷의 헤더에 포함되여 서버에 요청 따라서, BODY에 특별한 내용을 넣을 것이 없으므로 BODY가 빈상태로 보내진다. 따라서 헤더에 바디를 설명하는 Content-Type이라는 헤더필드는 들어가지 않는다.  (4)URL형태로 표현되므로, 특정 페이지를 다른사람 에게 접속하게 할 수 있다.  (5)간단한 데이터를 넣도록 설계 되어 데이터를 보내는 양의 한계가 있다. |

\*퍼스트 인코딩 : URL에 문자를 표현하는 문자 인코딩 방법. 이 방법에 따르면 알파벳이나 숫자 등 몇몇 문자를 제외한 값은 옥텟 단위로 묶어서,16진수 값으로 인코딩한다.ex) %EC%9C%84%ED%82%A4%EB%B0%B1%EA%B3%BC

**HEAD 메서드 방식**

**;** HEAD 요청 방식은 GET과 유사한 방식이나 웹 서버에서 헤더 정보이외에는 어떠한 정보도 보내지 않는다. 웹 서버의 다운 여부 점검(Health Check) 이나 웹 서버 정보 (버전 등) 등을 얻기 위해 사용.

**OPTIONS 메서드 방식**

: 해당 메서드를 통해 시스템에서 지원되는 메서드 종류를 확인할 수 있다.

**PUT 메서드 방식**

: POST와 유사한 전송 구조를 가지기 때문에 헤더 이외에 메시지(데이터)가 함께 전송된다. 원격지 서버에 지정한 콘텐츠를 저장하기 위해 사용되며 홈페이지 변조에 맣이 악용되고 있다.

**DELETE 메서드 방식**

: 원격지 웹 서버에 파일을 삭제하기 위해 사용되며 PUT과는 반대 개념의 메서드이다.

**TRACE 메서드 방식**

: 원격지 서버에 Loopback(루프백) 메시지를 호출하기 위해 사용된다.

**CONNECT 메서드 방식**

**:** 웹 서버에 프락시 기능을 요청할 때 사용된다.

**각 메서드 전송 형태**

|  |  |
| --- | --- |
|  | **전송 형태** |
| **GET** | GET [request-uri]?query\_string  HTTP/1.1\r\n  Host:[Hostname] 혹은 [IP] \r\n |
| **POST** | POST [request-uri]?query\_string  HTTP/1.1\r\n  HOST:[Hostname] 혹은 [IP] \r\n  Content-Lenght:[Lenght in Bytes] \r\n  \r\n  [query-string] 혹은 [데이터] |
| **HEAD** | HEAD [request-uri] HTTP/1.1\r\n  Host:[Hostname] 혹은 [IP] \r\n |
| **OPTIONS** | PUT [request-uri] HTTP/1.1\r\n  Host:[Hostname] 혹은 [IP] \r\n  Content-Lenght:[Length in Bytes] \r\n  Content-Type:[Content Type] \r\n  \r\n  [데이터] |
| **PUT** | PUT [request-uri] HTTP/1.1\r\n  Host:[Hostname] 혹은 [IP] \r\n  Content-Lenght:[Length in Bytes] \r\n  Content-Type:[Content Type] \r\n  \r\n  [데이터] |
| **DELETE** | DELETE [request-uri] HTTP/1.1\r\n  Host:[Hostname] 혹은 [IP] \r\n  \r\n |
| **TRACE** | TRACE [request-uri] HTTP/1.1\r\n  Host:[Hostname] 혹은 [IP] \r\n  \r\n |
| **CONNECT** | CONNECT [request-uri] HTTP/1.1\r\n  Host:[Hostname] 혹은 [IP] \r\n  \r\n |

**프로토콜 헤더 포멧**

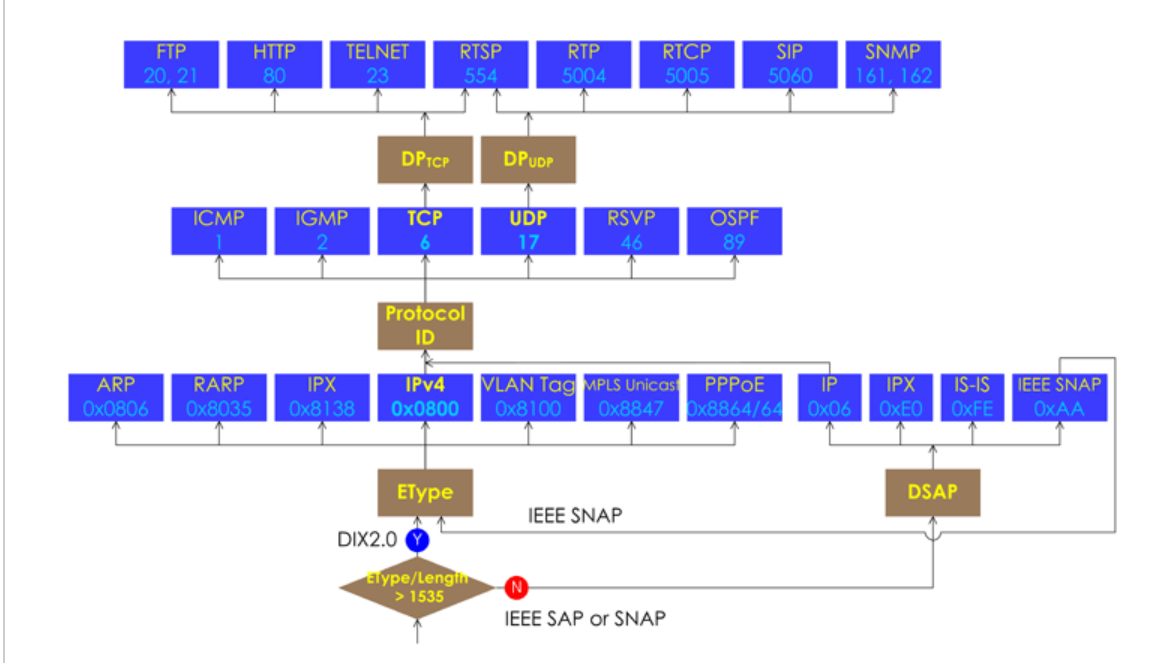


Figure 1

**L2 : Ethenet Header**

**:** Destiantion MAC address(6B), Source MAC address(6B) 그리고 Ethenet Type(2B)로 구성됨,Ethenet Type의 값이 1536(0x600) 보다 크면 RFC 894에서 규정한 DIX 2.0 포맷이 됨, 만약 그 보다 작으면 IEEE에서 정한 SNAP혹은 SAP포멧이 됨, 현재 유선 Ethenet망에서 다니는 대부분의 패킷은 이 DIX2.0

Ethenet Type 필드는 L3 패킷의 타입을 명시하는 역할, Ethenet Type=0x0800이면 IPv4패킷 0x0806이면 ARP 패킷이다. 0x8100이면 VLAN Tag가 Ethenet 헤더와 L3 패킷사이에 붙는다는 의미. 이 VLAN Tag에는 VLAN ID와 Qos정보를 담고 있는 802.1p(cos) 가 포함 그리고 VLAN tag이후에 L3 패킷 타입을 명시하기위해 다시 Ethenet Type이 나오게 됨.

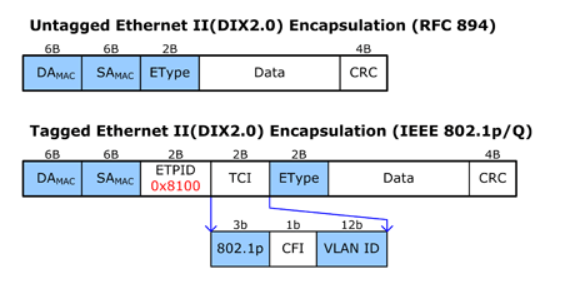


Figure 2 Ethenet Header

**L3 : IP Header**

**:** Ethernet Type=0x0800이면 IPv4(IP version 4) 패킷임. 현재 전 지구상에 다니는 대부분의 패킷. IPv4 헤더는 통상적으로 20B(바이트) 여기에 Source IP Address(4B), Destination IP Address(4B)포함. 이 중에 Protocol ID(2B)는 L4 패킷 타입을 명시하는 역할을 함. 그래서 맨 위 그림과 같이 Protocol ID=1이면 ICMP(Ping이 ICMP 패킷의 일종), 2이면 Multicast에서 사용되는 IGMP(집에 IPTV에서 채널 바꾸면 셋톱박스에서 망으로 IGMPv2를 보냄), 6이면 TCP, 17이면 UDP입니다.

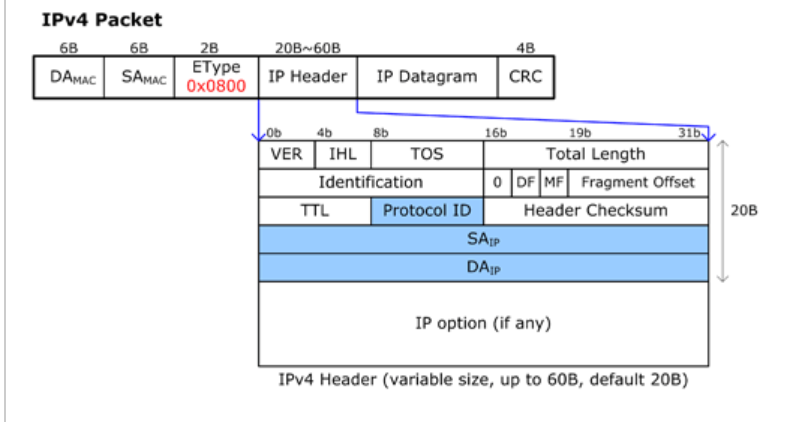


Figure 3 IP Header

**L4 : TCP/UDP Header**

: . UDP 헤더는 8B(바이트)이고 TCP 헤더는 20B. 이 두 헤더 모두 첫 2B가 Source Port Number(SP)이고, 다음 2B가 Destination Port Number(DP)이다. 여기서 SP에는 패킷 송신 단말에서 고른 random한 값이 들어감.(이를 Ephemeral Port라고 부릅니다. DP에 바로 "응용 트래픽"의 타입이 들어가게 됨. HTTP는 TCP를 사용하는 응용이므로 TCP 헤더의 DP=80이고, VoIP에서 사용되는 SIP는 UDP 헤더의 DP=5060이다.

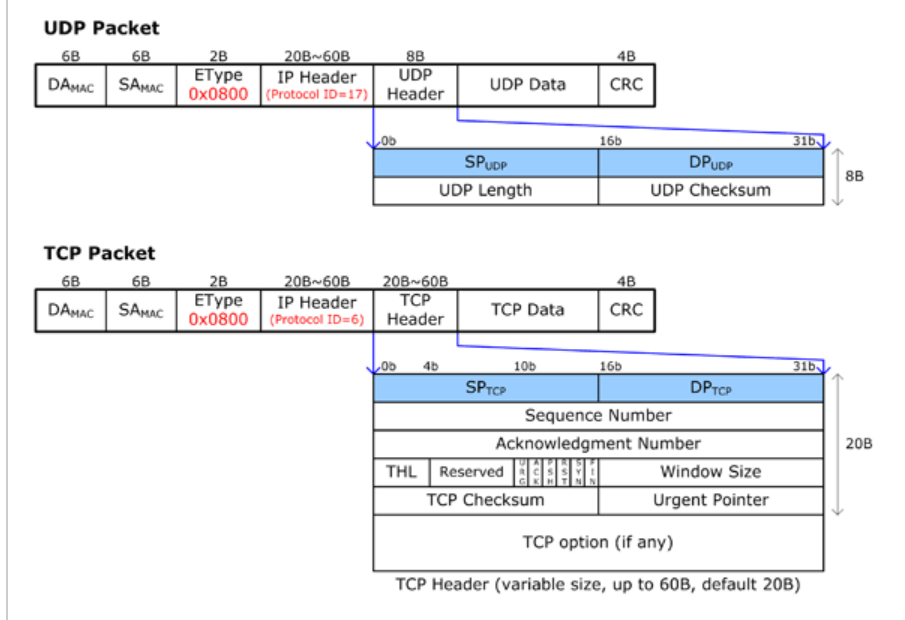


Figure 4 TCP/UDP

**HTTP/응답코드**

1. 1xx : 정보 전달 : 요청을 받았고, 작업을 진행 중이라는 의미

* 100 Continue
* 101 Switching Protocols
* 102 Processing

1. 2xx : 성공 : 이 작업을 성공적으로 받았고, 이해했으며, 받아들여졌다는 의미.200과 206을 제외하고는 볼 일이 거의 없는 코드

* 200 OK : 성공적으로 처리했을 때 쓰인다. 가장 일반적으로 볼 수 있는 HTTP 상태
* 206 Parial Content : 콘텐츠의 일부 부문만 제공한다. 보통 클라이언트에서 시작 범위나 다운로드할 범위를 지정한 경우 자동으로 해당 부부만 제공할 때 사용하는 코드

1. 3xx : 리다이렉션 : 이 요청을 완료하기 위해서는 리다이렉션이 이루어져야 한다는 의미, 짧은 주소(단축 URL) 서비스의 경우 접속 시 301이나 302 코드를 보내고, 헤더의 location에 리다이렉션할 실제 URL을 적어 보낸다.

* 300 Multiple Choices
* 301 Moved Permanently: 영구적으로 컨텐츠가 이동했을 때 쓴다.
* 302 Found: 다른 페이지로 이동하지만, 나중에 바뀔 수 있음.
* 303 See Other
* 304 Not Modified: 200 다음으로 많이 볼 수 있는 HTTP 상태이다. 이 경우 보통 브라우저에 캐시되어 있는 버전을 쓴다.
* 305 Use Proxy 요청자는 프록시를 사용하여 요청한 페이지만 액세스할 수 있다. 서버가 이 응답을 표시하면 요청자가 사용할 프록시를 가리키는 것이기도 하다.
* 306 Switch Proxy
* 307 Temporary Redirect 일시 리다이렉트
* 308 Permanent Redirect 영구 리다이렉트
* 307,308은 아직 [초안 단계](https://tools.ietf.org/html/rfc7238)의 표준으로 대부분의 브라우저에서 해석하지 못한다.

1. 4xx : 클라이언트 오류 : 이 요청은 올바르지 않다는 의미이다. 여기서부터 브라우저에 직접 표출된다. 굵게 강조된 것은 자주 보이는 오류들이다.

* **400 Bad Request**: 요청 자체가 잘못되었을때 사용하는 코드이다.
* **401 Unauthorized**: 인증이 필요한 리소스에 인증 없이 접근할 경우 발생한다. 이 응답 코드를 사용할 때에는 반드시 브라우저에게 어느 인증 방식[[1]](https://namu.wiki/w/HTTP/%EC%9D%91%EB%8B%B5%20%EC%BD%94%EB%93%9C#fn-1)을 사용할 것인지 보내 주어야 한다. 단순히 권한이 없는 경우 이 응답 코드 대신 아래 403 Forbidden을 사용해야 한다.
* [**403 Forbidden**](https://namu.wiki/w/403%20Forbidden): 서버가 요청을 거부할 때 발생한다. 관리자가 해당 사용자를 차단했거나 서버에 index.html 이 없는 경우에도[[2]](https://namu.wiki/w/HTTP/%EC%9D%91%EB%8B%B5%20%EC%BD%94%EB%93%9C#fn-2) 발생할 수 있다. 혹은 권한이 없을 때(로그인 여부와는 무관하다)에도 발생한다.
* [404 Not Found](https://namu.wiki/w/404%20Not%20Found): 찾는 리소스가 없다는 뜻이다.
* 405 Method Not Allowed : PUT이나 DELETE 등 서버에서 허용되지 않은 메소드로 요청시 사용하는 코드이다.
* 406 Not Acceptable : 요청은 정상이나 서버에서 받아들일 수 없는 요청일시 사용하는 코드이다. 보통 웹 방화벽에 걸리는 경우 이 코드가 반환된다.
* 407 Proxy Authentication Required : 프록시 인증이 필요할 경우
* **408 Request Timeout** : 요청 중 시간이 초과되었을때 사용하는 코드이다.
* 409 Conflict
* 410 Gone : 404와는 달리 찾는 리소스가 영원히 사라진 경우 사용하는 코드이다.
* 411 Length Required
* 412 Precondition Failed
* 413 Requested Entity Too Large : 요청 본문이 너무 긴 경우 발생한다. 서버 소프트웨어로 엔진엑스를 사용하는 경우 기본 설정 그대로 사용하면 큰 첨부파일을 올릴 때 이 오류 코드가 발생하게 된다.
* 414 Requested URL Too Long: URL이 너무 길 때 발생한다.
* 415 Unsupported Media Type
* 416 Requested Range Not Satisfiable : 요청 헤더의 Range로 지정한 범위가 잘못되었을 때 발생한다.
* 429 Too Many Requests: 일정 시간 동안 너무 많은 요청을 보냈을 때 이를 거부하기 위해 사용한다. 나무위키에서는 페이지 소스(/raw/문서명) 보기를 너무 자주 요청했을 때 임시 차단을 걸기 위해 사용한다.

1. 5xx : 서버 오류 : 올바른 요청에 대해 서버가 응답할 수 없다는 의미

* [**500 Internal Server Error**](https://namu.wiki/w/500%20Internal%20Server%20Error): 서버에 오류가 발생해 작업을 수행할 수 없을 때 뜬다. 보통 설정이나 퍼미션 문제. 아니면 HTTP 요청을 통해 호출한 문서가 실제 HTML 문서가 아니라 JSP, PHP, 서블릿 등의 프로그램일 경우 그 프로그램이 동작하다 세미콜론 빼먹는 등의 각종 에러로 비정상종료하는경우 이 응답코드를 보낸다.
* 501 Not Implemented
* [**502 Bad Gateway**](https://namu.wiki/w/502%20Bad%20Gateway): [게이트웨이](https://namu.wiki/w/%EA%B2%8C%EC%9D%B4%ED%8A%B8%EC%9B%A8%EC%9D%B4)가 잘못 되었을 때 나온다.
* **503 Service Temporarily Unavailable**: 서버를 현재 일시적으로 사용할 수 없을 때 뜬다. 유지보수중이거나, 터졌거나 할 때 발생함.
* **504 Gateway Timeout**
* 505 HTTP Version Not Supported: HTTP 버전을 서버가 처리할 수 없다. 웬만해서는 볼 수 없다.
* 509 Apache bw/limited extension: 대역폭 제한 초과,
* **520**[**Unknown**](https://namu.wiki/w/Unknown)[**Error**](https://namu.wiki/w/Error): 말 그대로 알 수 없는 오류.

**쿠키와 세션**

**[쿠키]**

: 쿠키는 클라이언트 로컬에 저장되는 키와 값이 들어아있는 작은 데이터 파일이다. 쿠키에는 이름, 값, 만료날짜(쿠키 저장기간),경로에 대한 정보가있다. 쿠키는 일정시간동안 데이터를 저장할 수 있다. (로그인 상태 유지에 활용) 쿠키는 클라이언트의 상태 정보를 로컬에 저장했다가 참조한다.

\*쿠키의 프로세스

1. 브라우저에서 웹페이지 접속

2. 클라이언트가 요청한 웹페이지를 받으면서 쿠키를 클라이언트 로컬(하드)에 저장

3. 클라이언트가 재 요청시 웹페이지 요청과 함께 쿠키값도 전송

4. 지속적으로 로그인 정보를 가지고 있는 것처럼 사용

\*쿠키의 사용 사례

- 자동 로그인 팝업에서 ‘오늘 더 이상 이 창을 보지 않음’ 체크,쇼핑몰의 장바구니

\*쿠키의 제한

- 클라이언트에 300개까지 쿠키저장 가능, 하나의 도메인당 20개의 값만 가질 수 있음, 하나의 쿠키값은 4KB까지 저장 Response Header에 Set-Cookie 속성을 사용하면 클라이언트에 쿠키를 만들 수 있다.

쿠키는 사용자가 따로 요청하지 않아도 브라우저가 Resqust시에 Requset Header를 넣어서 **자동으로 서버에** 전송한다.

**[세션]**

: 일정 시간동안 같은 브라우저로부터 들어오는 일련의 요구를 하나의 상태로 보고 그 상태를 유지하는 기술 즉, 웹 브라우저를 통해 웹 서버에 접속한 이후로 브라우저를 종료할 때 까지 유지되는 상태 클라이언트가 Request를 보내면, 해당 서버의 엔진이 클라이언트에게 유일한 ID를 부여하는 데 이것이 세션이다.

**- 세션 프로세스**

1. 클라이언트가 서버에 접속시 세션 ID를 발급

2. 서버에서는 클라이언트로 발급해준 세션 ID를 쿠키를 사용해 저장 (JSESSIONID)

3. 클라이언트는 다시 접속할 때, 이 쿠키(JSESSIONID)를 이용해서 세션ID값을 서버에 전달

즉, 세션을 구별하기 위해 ID가 필요하고 그 ID만 쿠키를 이용해서 저장해놓는다. (쿠키사용) 쿠키는 자동으로 서버에 전송되니까 서버에서 세션아이디에 따른 처리를 할 수 있음

예를들면 게시판에 글을 작성할 때 작성 버튼을 누르면 세션에 있는 아이디를 참조해서 작성자를 지정하게 한다.

\*세션 사용 사례

-로그인 정보 유지

**쿠키와 세션의 차이**

|  |  |  |
| --- | --- | --- |
|  | **쿠키** | **세션** |
| **저장 위치** | 클라이언트에 파일로 저장 | 서버에저장 |
| **보안** | 클라이언트 로컬에 저장되기 때문에 변질되거나 request에서 스나이핑 당할 우려가 있어서 보안에 취약하다. | 쿠키를 이용해서 sessionid 만 저장하고 그것으로 구분해서 서버에서 처리하기 때문에 비교적 보안성이 좋다. |
| **라이프 사이클** | 쿠키도 만료시간이 있지만 파일로 저장되기 때문에 브라우저를 종료해도 계속해서 정보가 남아 있을 수 있다.  또한 만료기간을 넉넉하게 잡아두면 쿠키삭제를 할 때 까지 유지될 수도 있다. | 세션도 만료기간을 정할 수 있지만 브라우저가 종료되면 만료시간에 상관없이 삭제된다. |
| **속도** | 쿠키에 정보가 있기 때문에 서버에 요청시 속도가 빠르다. | 세션은 정보가 서버에 있기 때문에 처리가 요구되어 비교적 느린 속도를 낸다. |

**\*세션을 주로 사용하면 좋은데 왜 쿠키를 사용할까?**

**-** 세션은 서버의 자원을 사용하기 때문에 무분별하게 만들다보면 서버의 메모리가 감당할 수 없어질 수가 있고 속도가 느려질 수 있기 때문

\*쿠키,세션은 캐시와 엄연히 다르다.

캐시는 이미지나 css,js파일 등이 사용자의 브라우저에 저장이 되는 것이다.이를 이용해 자원이 아껴지는 것, 한 번 캐시에 저장되면 브라우저를 참고하기 때문에 서버에서 변경이 되어도 사용자는 변경되지 않게 보일 수 있는데 이런 부분을 캐시가 지워주거나 서버에서 클라이언트로 응답을 보낼 때 header에 캐시 만료시간을 명시하는 방법등을 이용할 수 있다.

\*보통 쿠키와 세션의 차이를 물어볼 때 저장위치와 보안에 대해서는 잘 말하는데 사실 중요한 것은 라이프사이클을 애기하는 것이다.

또한, 이런 이론적인 것 뿐만 아니라 실제로 쿠키,세션을 사용해 보았는지가 중요함.