KWEB Study Week4:
HTTP & Web Server

KWEB 2학기 준회원 스터디

M.G. BAE

J.H. BAEK

Today's Contents

1. Before Study
2. Concept of HTTP
3. HTTP Request
4. HTTP Response
5. Web Server
6. 과제

2017 Fall Semester

Before Study

- 여러분들! 오랜만입니다~ 뭐 어떻든 중간고사가 완료되었네요..
- 지난 주차(?)에 우리는 DB 기초에 관해 다루었고, NoSQL 중 MongoDB를 활용해 실습해보았습니다.
- 3주차의 목표는 웹 서비스를 이루는 HTTP 에 관해 배우는 것입니다.
- 지난 수업들 한 번 되짚어보고 시작할게요!

Previous Study

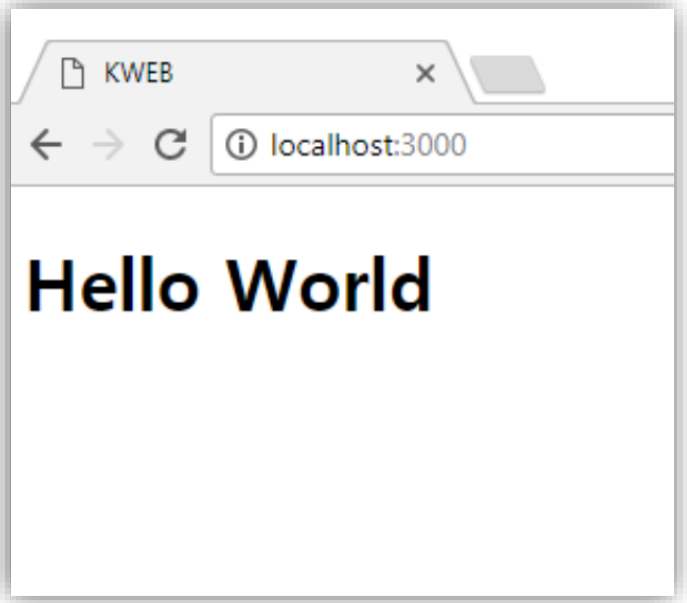
- 0주차: Node.js 소개 및 설치
- 1주차: Node.js 기초, Module, View Template, npm
- 2주차: Asynchronous 이해 및 Javascript (Object, Function, Clojure, Class)
- 3주차: DB 기초 (Relational DB, NoSQL, etc.)

What is HTTP

- 우리는 0주차 부터 실습을 통해 HTTP서버를 띄우고 여러가지 작업을 해보았습니다.

- 그런데 HTTP라는 놈은 무엇일까요..?

```
const http = require('http');
const path = require('path');
const fs = require('fs');
// const process = require('process');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  // res.end('Hello World\n');

  const indexContent = fs.readFileSync(path.join(__dirname, 'index.html'));
  res.end(indexContent);
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});

```



HTTP

- HTTP (Hyper Text Transfer Protocol)
  - 인터넷에서, 웹 서버와 사용자의 인터넷 브라우저 사이에 문서를 전송하기 위해 사용되는 통신 규약

- 여기서 Protocol은 **통신 규약**입니다. (전공 중에 데이터통신이나 컴퓨터 네트워크 수업에서 다룰 것입니다.)

- 다시 말하면, Protocol은 네트워크상의 단말들이 메세지를 주고받는 것에 대한 **규칙**과 **양식**을 정하는 규약입니다.

- 즉, HTTP는 정해진 규칙과 양식에 따라서 Hyper Text(HTML)을 전송하는 것에 대한 규약입니다!

HTTP

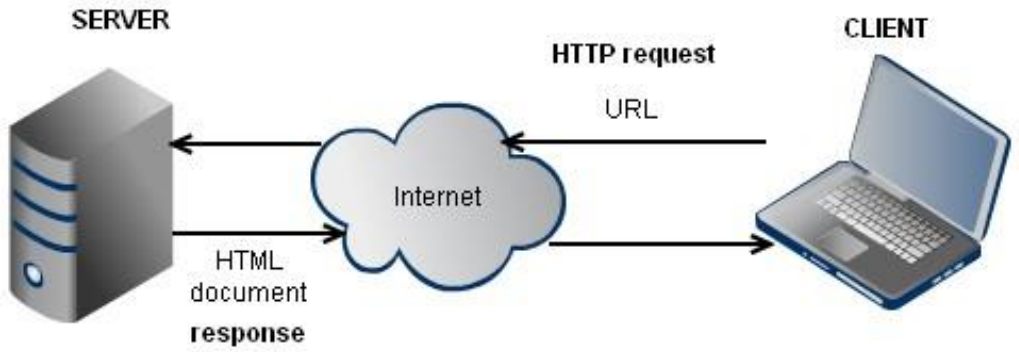
- HTTP는 앞서 말한대로 정해진 규칙과 양식을 갖습니다.
  - 규칙 - 언제 어떻게 어떤 메세지를 전송하는지!
  - 양식 - 전송하는 메세지의 형식! (참고로, HTML은 아닙니다.)

- OSI 7 layer Model에 따르면 Application Layer에서 사용하는 Protocol입니다.
  - 역시나, 데이터통신과 컴퓨터 네트워크 시간에 배우실 겁니다.

HTTP 규칙

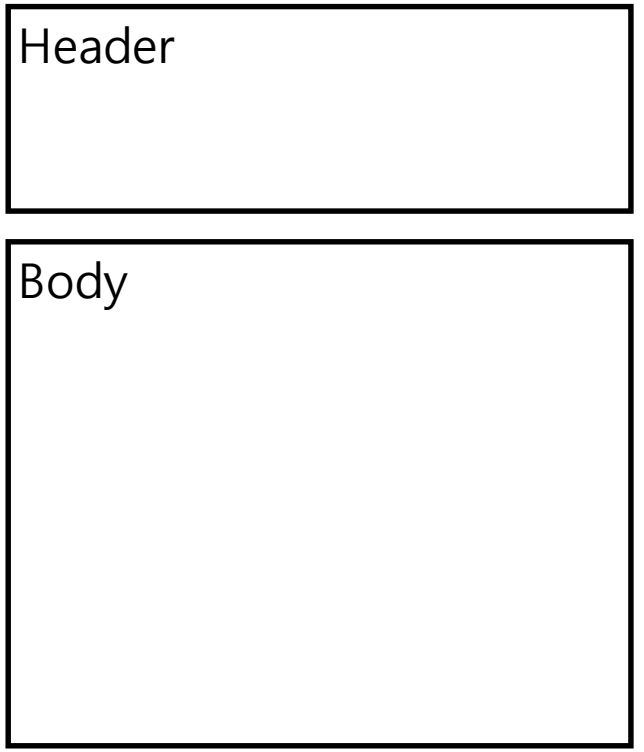
- HTTP의 규칙은 기본적으로 server/client 모델을 따릅니다.
  - Server: Web server sends resources in response in requests using HTTP
  - Client: browser that requests, receives and displays web resources using HTTP

- server/client 모델에서는 클라이언트는 요청 (Request)를 서버로 전송하고 서버는 이 요청을 해석해 응답 (Response)를 생성해 클라이언트에게 전송합니다.



HTTP 양식

- HTTP의 양식은 Header와 Body로 이루어 집니다.
- Header는 요청/응답에 대한 Meta data를 담습니다.
- Body는 실제 데이터를 담습니다.
- HTTP 1.1에 대해 Request / Resonpose 를 예시와 함께 알아봅시다.



HTTP Request

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string&/paramsXML=string
```

Header

Body

Method

Path

Host

HTTP version

Content-Type

HTTP Request - Method

- 간단하게 요청의 종류(Type)정도로 생각 하면 됩니다.

- GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE 등의 메소드가 있습니다.
  - GET 방식은 간단한 요청 (Ex. 페이지 로딩)등을 할 때 사용됩니다.
  - POST, PUT, DELETE 방식은 서버에게 본격적으로 뭔가를 요청 할 때 사용됩니다.
  - HEAD, OPTIONS, TRACE 방식은 연결을 점검할 때 사용됩니다.

- 대부분의 경우에는 GET, POST 방식 정도만 사용합니다.

HTTP Request - Path

- 서버에게 클라이언트가 무엇을 원하는지 알려 줍니다.
- 보통 파일의 경로를 적거나 Web Application에 미리 정의된 path를 이용하여 적습니다.
- 웹서버는 해당하는 리소스를 찾아 응답하게 됩니다.
- GET 메소드 사용시에 path에 간단한 parameter가 삽입될 수 있습니다.

HTTP Request - Host

- 메세지를 받을 Host(서버)를 나타냅니다.
- <[domain or ip]:[port]> 형식을 가지며 웹서버는 전달받은 내용에 따라 path의 내용을 찾을 곳을 선정 합니다.
- 경우에 따라서 하나의 물리적인 서버에 여러 개의 웹 어플리케이션이 올라가고 HTTP 헤더의 Host를 이용해 어떤 웹 어플리케이션에 요청을 보낼지 결정하기도 합니다. (Virtual Host 라고도 합니다)

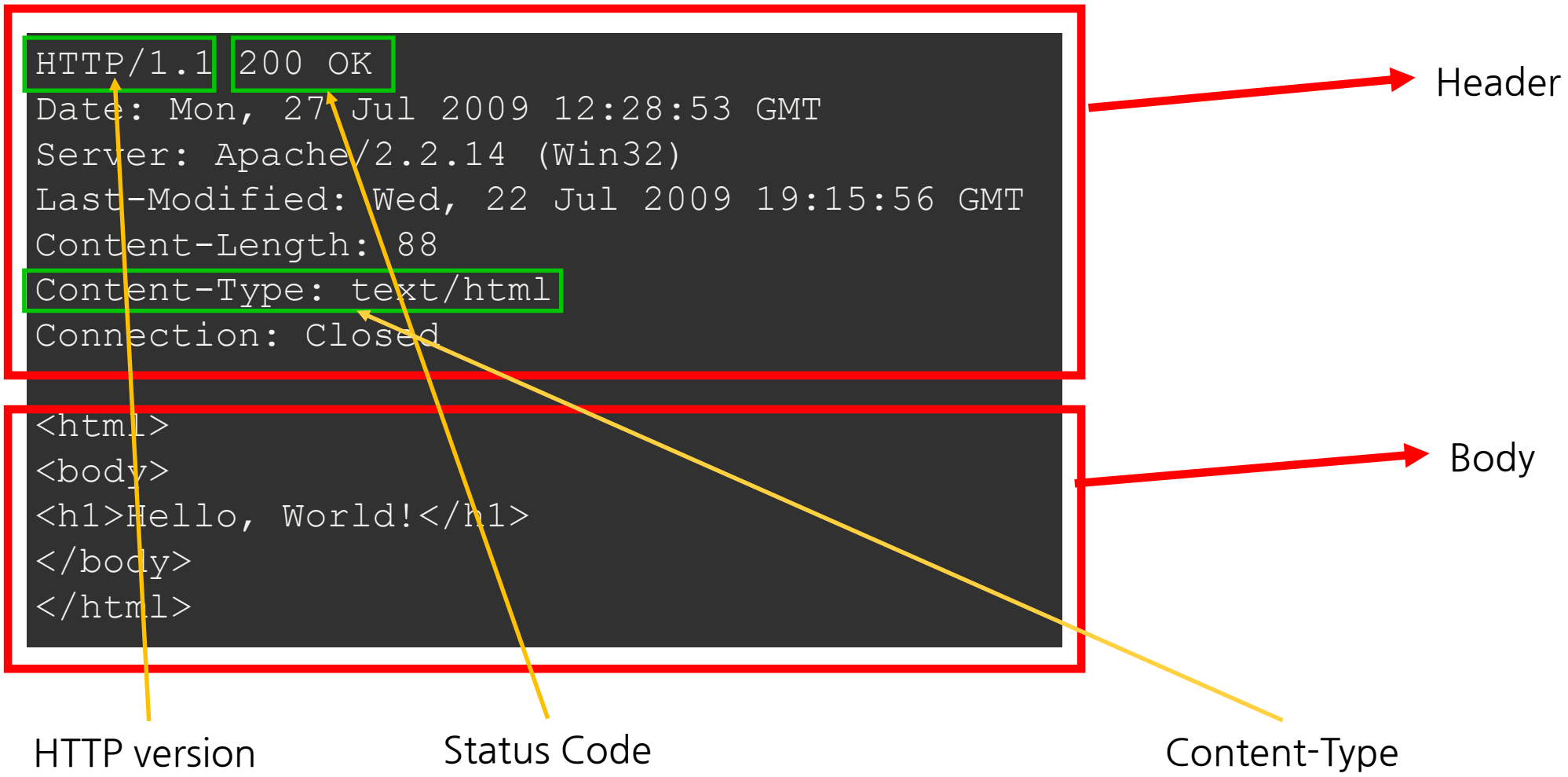
HTTP Request - Version

- 전송에 사용되는 HTTP의 버전을 나타냅니다.
- 현재 대부분의 웹은 HTTP/1.1을 사용합니다.
- 일부 구글이나 언론사들에서 HTTP/2를 사용한 서비스를 제공하고 있습니다.

HTTP Request - Content-type

- 요청 Body의 내용의 타입을 나타냅니다.
- 예시 에서의 application/x-www-form-urlencoded는 application 분류의 url encoding된 웹 폼을 뜻합니다.
- 대부분의 경우에 html 폼의 submit 기능을 사용하면 브라우저가 자동으로 데이터를 인코딩하여 Body에 기입하고 Header에 content-type을 기입합니다.
- Response에서도 응답 내용의 형식을 나타내기 위해 사용합니다.

HTTP Response



HTTP Response - Status Code

Informational Status Codes

100 – Continue [The server is ready to receive the rest of the request.]

101 – Switching Protocols [Client specifies that the server should use a certain protocol and the server will give this response when it is ready to switch.]

Client Request Successful

200 – OK [Success! This is what you want.]

201 – Created [Successfully created the URI specified by the client.]

202 – Accepted [Accepted for processing but the server has not finished processing it.]

203 – Non-Authoritative Information [Information in the response header did not originate from this server. Copied from another server.]

204 – No Content [Request is complete without any information being sent back in the response.]

205 – Reset Content [Client should reset the current document. Ie. A form with existing values.]

206 – Partial Content [Server has fulfilled the partial GET request for the resource. In response to a Range request from the client. Or if someone hits stop.]

Request Redirected

300 – Multiple Choices [Requested resource corresponds to a set of documents. Server sends information about each one and a URL to request them from so that the client can choose.]

301 – Moved Permanently [Requested resource does not exist on the server. A Location header is sent to the client to redirect it to the new URL. Client continues to use the new URL in future requests.]

302 – Moved Temporarily [Requested resource has temporarily moved. A Location header is sent to the client to redirect it to the new URL. Client continues to use the old URL in future requests.]

303 – See Other [The requested resource can be found in a different location indicated by the Location header, and the client should use the GET method to retrieve it.]

304 – Not Modified [Used to respond to the If-Modified-Since request header. Indicates that the requested document has not been modified since the the specified date, and the client should use a cached copy.]

305 – Use Proxy [The client should use a proxy, specified by the Location header, to retrieve the URL.]

307 – Temporary Redirect [The requested resource has been temporarily redirected to a different location. A Location header is sent to redirect the client to the new URL. The client continues to use the old URL in future requests.]

Client Request Incomplete

400 – Bad Request [The server detected a syntax error in the client's request.]

401 – Unauthorized [The request requires user authentication. The server sends the WWW-Authenticate header to indicate the authentication type and realm for the requested resource.]

402 – Payment Required [reserved for future.]

403 – Forbidden [Access to the requested resource is forbidden. The request should not be repeated by the client.]

404 – Not Found [The requested document does not exist on the server.]

405 – Method Not Allowed [The request method used by the client is unacceptable. The server sends the Allow header stating what methods are acceptable to access the requested resource.]

406 – Not Acceptable [The requested resource is not available in a format that the client can accept, based on the accept headers received by the server. If the request was not a HEAD request, the server can send Content-Language, Content-Encoding and Content-Type headers to indicate which formats are available.]

407 – Proxy Authentication Required [Unauthorized access request to a proxy server. The client must first authenticate itself with the proxy. The server sends the Proxy-Authenticate header indicating the authentication scheme and realm for the requested resource.]

408 – Request Time-Out [The client has failed to complete its request within the request timeout period used by the server. However, the client can re-request.]

409 – Conflict [The client request conflicts with another request. The server can add information about the type of conflict along with the status code.]

410 – Gone [The requested resource is permanently gone from the server.]

411 – Length Required [The client must supply a Content-Length header in its request.]

412 – Precondition Failed [When a client sends a request with one or more If... headers, the server uses this code to indicate that one or more of the conditions specified in these headers is FALSE.]

413 – Request Entity Too Large [The server refuses to process the request because its message body is too large. The server can close connection to stop the client from continuing the request.]

414 – Request-URI Too Long [The server refuses to process the request, because the specified URI is too long.]

415 – Unsupported Media Type [The server refuses to process the request, because it does not support the message body's format.]

417 – Expectation Failed [The server failed to meet the requirements of the Expect request-header.]

Server Errors

500 – Internal Server Error [A server configuration setting or an external program has caused an error.]

501 – Not Implemented [The server does not support the functionality required to fulfill the request.]

502 – Bad Gateway [The server encountered an invalid response from an upstream server or proxy.]

503 – Service Unavailable [The service is temporarily unavailable. The server can send a Retry-After header to indicate when the service may become available again.]

504 – Gateway Time-Out [The gateway or proxy has timed out.]

505 – HTTP Version Not Supported [The version of HTTP used by the client is not supported.]

Unused status codes

306- Switch Proxy

416- Requested range not satisfiable

506- Redirection failed

HTTP protocol version 1.1 Server Response Codes

http://www.w3.org/Protocols/rfc2616/rfc2616.html

Chart created September 5, 2000 by Suso Banderas(suso@suso.org). Most of the summary information was gathered from Appendix A of "Apache Server Administrator's Handbook" by Mohammed J. Kabir.

- 요청에 대한 처리 상태를 나타냅니다.
  - 100번대: 정보성
  - 200번대: 성공
  - 300번대: 리소스의 이동 (리다이렉트)
  - 400번대: 리소스가 접근하지 못했을 때 (Client 오류)
  - 500번대: 처리중의 오류 (Server 오류)

- Content-type은 Request와 비슷합니다.

Web Server

- 웹서버는 HTTP 요청을 처리하는 소프트웨어와 하드웨어의 집합이다.
- 웹서버는 HTTP요청에 따라 요청된 파일을 찾아 주거나 서버사이드 스크립트를 실행시킨다.
- Node.js로 작성되어 실행된 인스턴스는 웹 어플리케이션 또는 웹 어플리케이션 컨테이너라고 불린다.
- 웹 서버는 이러한 웹 어플리케이션 (컨테이너)와 외부의 요청을 중개하는 역할을 하기도 한다. (Reverse Proxy)

Web Server

- 웹 서버를 통해 HTTP 에 관한 실습을 할 예정이었으나, 다음 주에 배울 Express.js를 배우고 나서 해볼 예정입니다.

- 고로! 담주부터 본격적으로 Node.js (feat. Express.js)를 통한 웹페이지 만들기가 진행될 예정입니다.

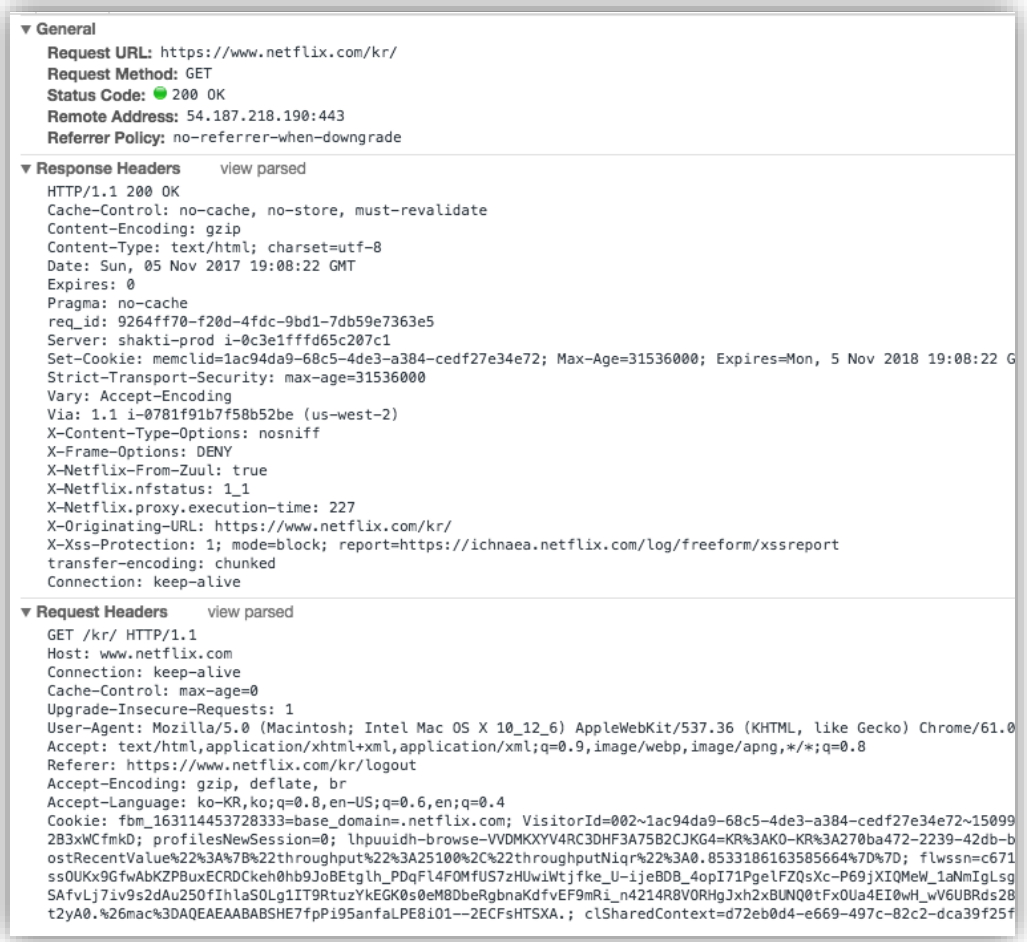
과제 (~ 11/12 23:59)

- 4주차는 HTTP에 관해 공부했습니다. (중간고사 후 첫 스터디라 짧게 진행했습니다.)
- 과제는 1,2 두개 입니다!
- 제출기한: 11월 12일 자정 - 제출은 이번에는 저한테 다 주세요!
- 과제 제출 E-mail
  - 15 배민근 (baemingun@naver.com)

과제

- 크롬으로 아무 사이트나 들어가서 개발자 도구를 이용해 network 탭에서 HTTP Request / Response 하나를 캡처 해서 5줄 안쪽으로 간단한 분석을 써주세요.

- 오른쪽 사진은 크롬 네트워크 탭에서 찍은 HTTP 요청의 예입니다.



▼ General

Request URL: https://www.netflix.com/kr/
Request Method: GET
Status Code: 200 OK
Remote Address: 54.187.218.190:443
Referrer Policy: no-referrer-when-downgrade

▼ Response Headers view parsed

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Sun, 05 Nov 2017 19:08:22 GMT
Expires: 0
Pragma: no-cache
req_id: 9264ff70-f20d-4fdc-9bd1-7db59e7363e5
Server: shakti-prod i-0c3e1fffd65c207c1
Set-Cookie: memclid=1ac94da9-68c5-4de3-a384-cedf27e34e72; Max-Age=31536000; Expires=Mon, 5 Nov 2018 19:08:22 G
Strict-Transport-Security: max-age=31536000
Vary: Accept-Encoding
Via: 1.1 i-0781f91b7f58b52be (us-west-2)
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Netflix-From-Zuul: true
X-Netflix.nfstatus: 1_1
X-Netflix.proxy.execution-time: 227
X-Originating-URL: https://www.netflix.com/kr/
X-Xss-Protection: 1; mode=block; report=https://ichnaea.netflix.com/log/freeform/xssreport
transfer-encoding: chunked
Connection: keep-alive
```

▼ Request Headers view parsed

```
GET /kr/ HTTP/1.1
Host: www.netflix.com
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: https://www.netflix.com/kr/logout
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: fbm_163114453728333=base_domain=.netflix.com; VisitorId=002~1ac94da9-68c5-4de3-a384-cedf27e34e72~15099
2B3xWCfmkD; profilesNewSession=0; lhpuuidh-browse-VVDMKXYV4RC3DHF3A75B2CJKG4=KR%3AKO-KR%3A270ba472-2239-42db-b
ostRecentValue%22%3A%7B%22throughput%22%3A25100%2C%22throughputNiqr%22%3A0.8533186163585664%7D%7D; flwssn=c671
ssOUKx9GfwAbKZPBuxECRDCkeh0hb9JoBEtglh_PDqFl4FOMfUS7zHUwiWtjfke_U-ijeBDB_4opI71PgelFZQsXc-P69jXIQMeW_1aNmIgLsg
SAfvLj7iv9s2dAu25OfIhlaSOLg1IT9RtuzYkEGK0s0eM8DbeRgbnaKdfvEF9mRi_n4214R8VORHgJxh2xBUNQ0tFxOUa4EI0wH_wV6UBRds28
t2yA0.%26mac%3DAQEAEAABABSHE7fpPi95anfaLPE8iO1--2ECFsHTSXA.; clSharedContext=d72eb0d4-e669-497c-82c2-dca39f25f
```

It’s all today!