HTTP

TCP 기반 client-server 프로토콜이다.

- 보안 취약함 (평문 그대로 드러남)
- TLS를 합쳐 HTTPS 사용함

HTTP Request

```
GET/HTTP/1.1
Host: example.com
```

HTTP Response

```
HTTP/1.1 200 OK
Conetent-Type: text/html
Content-Length: 1255
...
```

HTTP 메소드

GET: 서버에서 리소스 가져오기
HEAD: 실제 resource보다 간략한 정보
POST: body를 통해 요청 데이터 전달
PUT: 리소스 있으면 대체 / 없으면 생성
-> POST와 달리 리소스 URI를 알고있어야 한다.
PATCH: 리소스 부분변경 가능
DELETE: 리소스 삭제

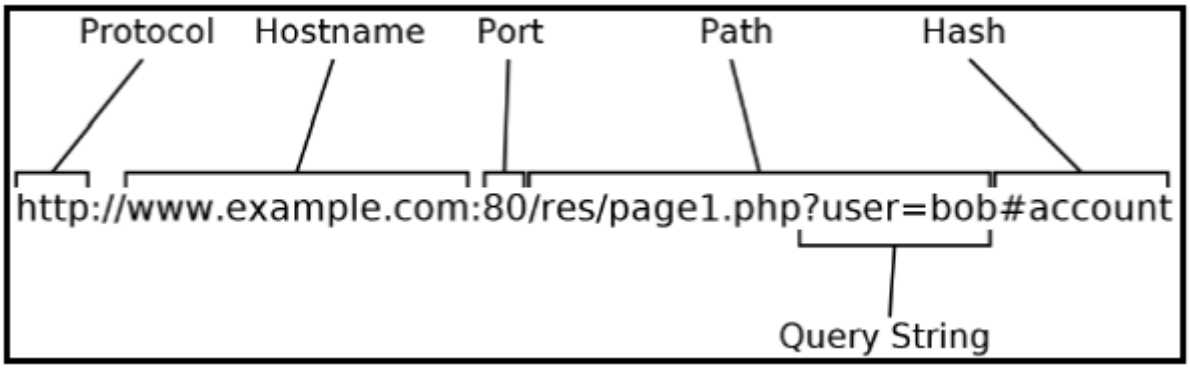
- GET에 HTTP Body 없음
- \r\n을 통해 라인 끝났다는 것을 알림
  Header - Body 사이 개행문자 2개

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=ascii
Transfer-Encoding: chuncked(\r\n\r\n)
44//16진수값
Lorem~~..
37
mod tempor~~..
0 //HTTP Body end
```

State

200 - OK
301 - Moved Permanently
400 - Client 잘못 500 - Server 잘못

URL



A Web client

▶ web_get.c

Code Summary

1. hostname을 받으면 parseURL을 통해 파싱한다.
   hostname, port, protocol.. 등 나옴

2. hostname을 통해 서버와 연결해 HTTP Request를 한다.

3. Response를 받으면 파싱해서 출력한다.

    - 헤더 / 바디 \r\n\r\n기준 파싱
    - 헤더에서 Body 여부 enum으로 판단 -> length, chuncked, connection

4. Body 존재시 L -> V 순서로 계속 읽어들임

POST

Web Client에서 생성한 데이터 서버로 전달한다
loginform 같은 경우도 마찬가지로 동작
-> http는 평문으로 통신하기 때문에 유출위험존재

HTTP Basic

- URI = URL + URN
- scheme://[userinfo@]host[:post][/path][?query]
  https://www.google.com:443/search?q=hello&hl=ko
    - scheme: 프로토콜 (http, https..)
    - userinfo: 사용자 정보 인증.. (거의 사용x)

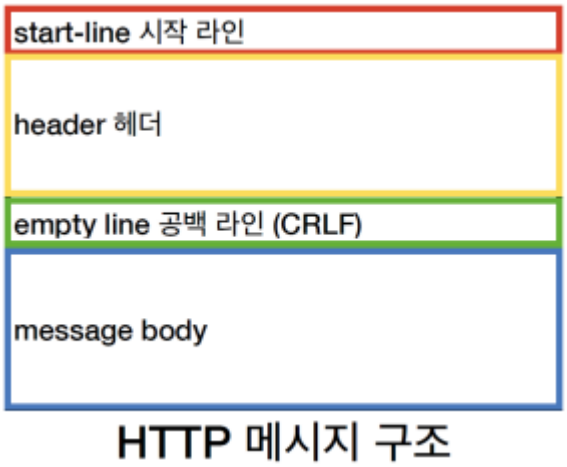
- host: 호스트 이름 (www.naver.com)
- 포트번호: http-80, https-443 (생략가능)
- path: 리소스의 경로, 계층적 구조 (mycard/ssoxong)
- 쿼리: key=value 형태

HTTP 특징

- stateless
  무상태 프로토콜
  서버는 클라이언트 상태를 보존하지 않기 때문에 서버의 확장성이 높지만, 클라이언트에게 불편함을 준다.

- connectionless
  연결을 유지하지 않아 서버자원을 효율적으로 활용할 수 있다.
  하지만 지속적인 HandShake 오버헤드가 발생한다.

HTTP 메세지 구조



HTTP 메시지 구조

HTTP Body에는 실제 전송할 데이터가 담겨있다.

HTTP 메소드 활용

- 정적 데이터: GET 활용해서 리소스로 가져오기
- 동적 데이터: 쿼리 파라미터 기반으로 생성

HTTP 상태코드

1. 200 - 요청 정상 처리 200 ok 201 created 202 accepted - 접수는 됐는데 처리X 204 no content - 응답보낼게없음

2. 300 - 요청 완료를 위한 추가 처리 필요 <--영구 Redirection--> 301 리다이렉트시 GET, 본문 제거 가능성 308 301과 비슷, 본문 유지
   <--일시적 Redirection--> 302 Found - 리다이렉트시, GET, 본문 제거 307 Temporary Redirect - 요청 메서드와 본문 유지 303 See others - 요청 메서드 GET

일시적인 Redirection 예시

PRG(POST/REDIRECT/GET)을 통한 처리
post로 주문 -> 새로고침시 중복주문 방지

1. POST로 주문후에 주문 결과 화면을 GET 메서드로 Redirect
2. 새로고침해도 결과 화면을 GET으로 조회함
3. 중복 주문 대신에 결과 화면만 GET으로 다시 요청함
4. 400 - 클라이언트 오류
   400 Bad Request 401 Unauthorized 403 Forbidden 404 Not Found
5. 500 - 서버 오류

302 Found

- redirect시, 요청 메서드가 GET으로 변하고, 본문이 제거될 수 있다.