HTTP 메서드 활용

- 클라이언트에서 서버로 데이터 전송
- HTTP API 설계 예시

클라이언트에서 서버로 데이터 전송

→데이터 전달 방식은 크게 2가지

① query parameter 통한 data 전송
- GET
- 주로 sort filter (검색어)

② message body 통한 data 전송
- POST, PUT, PATCH
- 회원 가입, 상품 주문, 리소스 등록, 리소스 변경

→ 4가지 상황

· 정적 data 조회

- image, 정적 text 문서

· 동적 data 조회

- 주로 검색, 게시판 목록에서
  sort filter (검색어)

· HTML Form 통한 data 전송

- 회원 가입, 상품 주문, 데이터 변경

· HTML API 통한 data 전송

→→ 정적 데이터 조회
(query param. 미사용)

/~/~.jpg

```
GET /~/~.jpg HTTP/1.1
Host: ~~~
```

```
HTTP/1.1 200 OK
Content-Type: image/jpeg
,, -Length: ~~~

~~~
~~~
```

• 정리

- image, 정적 text 문서
- 조회는 GET 사용
- 정적 data는 일반적으로
  query param. 없이
  resource path로 단순하게 조회 가능

→→ 동적 데이터 조회
(query param. 사용)

ex) ~//~/search?q=hello&hl=ko

q. param. 기반으로
sort filtering 해서
결과를 동적으로
생성

정리

- 주로 검색, 게시판 목록에서 sort filter (검색어)
- (조회 조건 줄여주는 필터 / 조회 결과 정렬하는 정렬조건)에 주로 사용
- 조회는 GET 사용
  - q. param. 사용해서 data 전달

HTML Form 데이터 전송

→ POST 전송 - 저장

```
<form action="/save"
      method="post">
  <input ~ ~="username" />
  .. .. ..="age" />

  ~~~~> 전송 ~
</form>
```

web browser가 생성한
req. HTTP message

```
POST /save HTTP/1.1
Host: ~
~-Type: application/x-www-form~

username=kim&age=20
```

→ GET 전송 - 저장

```
<form action="/save"
      method="get">
  <input ~ ~="username"/>
  .. .. ..="age"/>

  ~~~~> 전송 ~
</form>
```

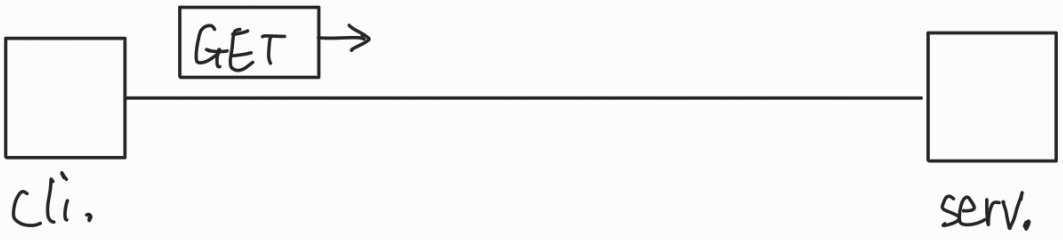
web browser가 생성한
req. HTTP message

```
GET /save?username=~&age=~ HTTP/1.1
Host: ~~
```

↑

GET은 조회에만 사용!

resource 변경 발생 부분엔 X



→ GET 전송 - 조회

```
<form action="/save"
      method="get">
  <input ~ ~="username"/>
  .. .. ..="age"/>

  ~~~~> 전송 ~
</form>
```

web browser가 생성한
req. HTTP message

```
GET /members?username~~ HTTP/1.1
Host: ~~
```

→ multipart/form-data

~~~

file: 파일선택

전송

~~~enctype="multipart/form-data"

~~~

~~~

~~~

----XXX

Content-Disposition: form-data; ~

kim

----XXX

~~~

20

----XXX

~~~

Content-Type: image/png

~~~

~---XXX--

HTML Form 데이터 전송

- HTML Form submit 시, ex) 회원 가입, ...
  POST 전송

  - Content-type: app~/x-www-form-urlencoded
    - form의 내용을 message body 통해 전송 (key=value 형식)
    - 전송 data를 url encoding 처리
  - HTML Form은 GET 전송도 가능

  - Content-type: multipart/form-data
    - file upload 같은 bin. data 전송 시 사용
    - 다른 종류의 여러 파일과 같이 ~ 가능
  - HTML Form 전송은 POST, GET 만 지원!

HTML API 전송

application/json

정리

- serv. to serv.
  : 백엔드 시스템 통신
- app. cli.
  : iOS, Android
- web cli.
  - JS 통한 통신 (AJAX)
    ex) React, VueJs 같은 web cli. 와 API 통신
- POST, PUT, PATCH : message body 통해 전송
- GET : cf. param. 통한 data 전달

HTTP API 설계 예시

- HTTP API - 컬렉션
  - POST 기반 등록
    - ex) 회원 관리 API 제공
- HTTP API - 스토어
  - PUT 기반 등록
    - ex) 정적 컨텐츠 관리, 원격 파일 관리
- HTML FORM 사용
  - web page 회원 관리
  - GET, POST 만 지원

회원 관리 시스템

→ API 설계 - POST 기반 등록

- 회원 목록
  ⇒ /members → GET
- 회원 등록
  ⇒ /members → POST
- 회원 조회
  ⇒ /members/{id} → GET
- 회원 수정
  ⇒ /members/{id} → PATCH, PUT, POST
- 회원 삭제
  ⇒ /members/{id} → DELETE

→ POST - 신규 자원 등록 특징

- client는 등록될 resource의 URI를 모른다.
  - 회원 등록
    ⇒ /members → POST

- server가 새로 등록된 resource URI 생성해준다.
  - HTTP/1.1 201 Created
    Location: /members/~

- 컬렉션 (Collection)
  - server가 관리하는 리소스 directory
  - server가 리소스의 URI를 생성하고 관리
  - 여기서 컬렉션은 /members

파일 관리 시스템

→ API 설계 - PUT 기반 등록

- 파일 목록
  ⇒ /files → GET
- 파일 조회
  ⇒ /files/{filename} → GET
- 파일 등록
  ⇒ /files/{filename} → PUT
- 파일 삭제
  ⇒ /files/{filename} → DELETE
- 파일 대량 등록
  ⇒ /files → POST

→ 신규 자원 등록 특징

- client가 리소스 URI를 알고 있어야 한다.
  - file 등록
    ⇒ /files/{filename} → PUT ⟺ cf) POST 등록 시엔 /files 까지만
    ex) PUT /files/star.jpg
- client가 직접 리소스의 URI를 지정한다.
- 스토어 (Store)
  - client가 관리하는 리소스 저장소
  - client가 리소스의 URI를 알고 관리
  - 여기서 스토어는 /files

HTML FORM 사용

→ • HTML FORM은 GET, POST만 지원

• GET, POST만 지원 ⇒ 제약

• AJAX 같은 기술로 해결 가능 ⇒ 회원 API 참고

• 회원 목록 ⇒ /members (GET)

• 회원 등록 폼 ⇒ /members/new (GET)

• 회원 등록 ⇒ /members/new, /members (POST)

• 회원 조회 ⇒ /members/{id} (GET)

• 회원 수정 폼 ⇒ /members/{id}/edit (GET)

• 회원 수정 ⇒ /members/{id}/edit, /members/{id} (POST)

• 회원 삭제 ⇒ /members/{id}/delete (POST)) 컨트롤 URI

(만약 DELETE를 쓸 수 없다면!)

cf) validation

→
- HTML FORM은
  GET, POST 만 지원

- 컨트롤 URI
  - GET, POST 만 지원하므로 제약 존재
  - 이런 제약 해결 위해
    동사로 된 resource 경로 사용
    ↳ POST의 /new, /edit, /delete] 컨트롤 URI
  - HTTP method로 해결 애매한 경우 사용
    (HTTP API 포함)

정리

- HTTP API - collection
  - POST 기반 등록
  - server가 resource URI 결정
- HTTP API - store
  - PUT 기반 등록
  - client가 resource URI 결정
- HTML FORM 사용
  - 순수 HTML + HTML form 사용
  - GET, POST만 지원