HTTP 프로토콜은 TCP와 UDP 프로토콜을 기반으로 하여 웹에서 사용되는 것으로서 클라이언트와 서버 사이에 이뤄지는 요청과 응답 데이터를 전송하는 방법을 말한다.

(1) HTTP 프로토콜의 Request Method 종류와 설명

HTTP 메시지 중 하나인 Request 메시지는 client-server 구조에서 클라이언트가 웹 서버에 파일을 요청할 때 보내는 HTTP 메시지이다. 이 메시지의 구조는 다음과 같다.

<table>
<tr><td>POST/HTTP/1.1</td><td colspan="2">Start-line</td></tr>
<tr><td>Host : localhost : 8000
User-Agent: Mozilla/5.0 (Macintosh; ...) ... Firefox/51.0
Accept: text/html, application/xhtml+xml, ... ,*/*;q=0.8
Accept-Language: en_US, en;q=0.5
Accept-Encoding: gzip, deflate</td><td>Request headers</td><td rowspan="3">Headers</td></tr>
<tr><td>Connection: keep-alive
Upgrade-Insecure-Requests: 1</td><td>General headers</td></tr>
<tr><td>Content-Type: multipart/form-date; boundary=-12656974
Content-Length: 345</td><td>Entity headers</td></tr>
<tr><td>CRLF</td><td colspan="2">Empty</td></tr>
<tr><td>-12656974
(more data)</td><td colspan="2">Body</td></tr>
</table>

Request Method의 종류

- GET

클라이언트가 웹 서버에게 URL에 해당하는 서버의 자료를 요청한다. GET 메소드 사용 시 본문(body)에 해당하는 부분은 비어 있다. URI를 통한 Query String을 다른 페이지로 전송할 때 주로 사용된다. 이때 모든 파라미터는 URL을 통해 전달되며 ?(구분자) 뒤에 오는 값이 파라미터 값으로 여러 개가 올 수 있다. 단, URL을 통해서 정보가 전달되므로 URL형식에 맞지 않거나 길이를 초과할 경우 전달되지 않는다. 이처럼 GET 메소드의 파라미터는 URL에 묻어나오기 때문에 비밀번호나 계좌번호 등 개인정보와 관련된 것들을 다룰 때는 사용하면 안 된다. 예를 들어 게시판 목록을 보는 것은 GET 메소드를 통해서 가능하다.

- HEAD

GET method와 같은 response를 요구하지만, response body는 포함하지 않는다. 즉 HTTP Header 정보만 수신한다. HTTP Header는 클라이언트, 서버 또는 HTTP와 관련된 정보를 담고 있는 General Header, 요청 형식과 서버의 매개 변수인 Request Header, 응답을 보내는 서버에 대한 정보를 담고 있는 Response Header가 있다. 웹 서버의 다운 여부 점검(Health Check)이나 웹 서버의 버전 정보 등을 얻기 위해 사용될 수 있다.

- POST

클라이언트가 HTML Form에 입력한 데이터를 웹서버로 전달할 때 사용되는데, 요청하는 URL 주소에 입력한 정보가 묻어나오지 않아서 보안성이 보장되어야 하는 데이터를 다룰 때 주로 사용된다. 이때 입력한 정보는 글자 수 제한이 없어서 많은 정보를 담을 수 있지만 그만큼 GET 메소드에 비해 처리 속도가 느리다. 글자 수 제한이 없는 대신에 Time Out이 존재하는데 time out에 설정한 시간이 지나도록 아무런 응답이 없다면 에러 페이지가 뜬다. 예를 들어 게시판에 올릴 내용을 작성하여 게시판에 올리는 것은 POST 메소드를 사용하는 것이다.

- PUT

POST 메서드와 비슷한 전송 구조를 가지며 메시지 본문의 내용을 지정된 URL에 저장하는데 이때 저장되는 것은 데이터 전체가 된다. 그래서 PUT 메소드를 사용할 때마다 데이터 전체가 수정되는 것이다. POST와 PUT의 차이는 POST가 정보를 insert한다면 PUT은 update의 개념이다. 따라서 동일한 데이터를 여러 번 POST 메소드를 사용한다면 서버에 아무런 변화도 없지만, PUT을 사용한다면 같은 데이터가 여러 개 생기게 되는 셈이다.

- DELETE

지정된 URL의 리소스를 서버에서 삭제한다. PUT 메소드와 반대되는 개념이다. 안정성 문제로 대부분 서버에서 비활성 모드이다.

- CONNECT

터널링을 목적으로 사용하거나 클라이언트가 원하는 목적지와의 TCP 연결을 HTTP 프록시 서버에 요청할 때 사용된다. 프록시 서버에는 클라이언트를 대신하여 연결을 생성하는데 한 번 생성된 연결은 클라이언트와 서버 사이에 오가는 TCP 스트림을 계속 프록시한다.

터널링 : 가상의 링크를 형성하는 것으로, 하나의 프로토콜이 다른 프로토콜을 감싸는 캡슐화 기능을 통해 운반한다. 일반적으로 터널링은 보안 채널의 역학을 하므로 암호화 기법 적용이 일반적이다.

프록시 서버(proxy server) : 클라이언트가 자신을 통해서 다른 네트워크 서비스에 간접적으로 접속할 수 있게 해주는 컴퓨터 시스템이나 응용 프로그램을 뜻한다. 서버와 클라이언트 사이에 중계기로서 대신 통신을 수행하는 것을 프록시라고 하며 그 중계 기능을 하는 것을 프록시 서버라고 한다. 프록시 서버 중 일부는 요청된 내용을 캐시로 저장해두었다가 캐시 안에 있는 정보를 요구할 때 원격 서버에 접속할 필요없이 프록시 서버의 저장된 캐시를 사용하므로 불필요한 연결이나 전송 시간을 절약할 수 있다. 또한 외부와의 트래픽을 줄이게 됨으로 네트워크 병목 현상을 방지하는 효과도 얻을 수 있다.

- OPTIONS

목적 리소스에 해당하는 통신 옵션들을 설정하기 위해 사용된다. 이때 OPTIONS request 메소드는 entity-Body(Content-Length or Transfer-Encoding을 지정하는 부분)를 포함하는데 이때 반드시 Content-Type field에 대한 정보가 있어야 한다. 또한, URL 자리에 *(asterisk)가 있다면 클라이언트에게 목적지에 해당하는 리소스뿐만 아니라 전체 서버에 대해

서 OPTIONS 메소드를 수행한다.

- TRACE

메시지가 프록시를 거쳐서 최종 목적지에 도착할 때까지의 경로를 기록하는 loop-back 메시지를 호출하기 위해 테스트용으로 사용된다.

- PATCH

PUT 메소드와 유사하게 요청된 자원을 수정(Update의 개념)할 때 사용한다. PUT의 경우에는 데이터 전체를 수정하는 의미지만, PATCH는 해당 자원의 일부를 교체하는 역할을 한다.