2. 간단한 프로토콜 HTTP

↗ Books	그림으로 배우는 HTTP & Network Basic

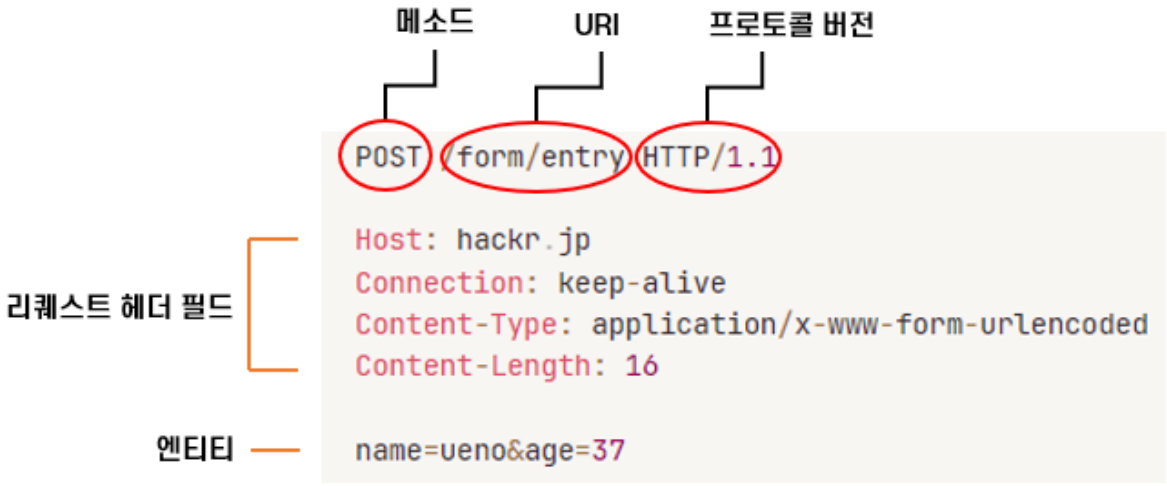
1️⃣ HTTP는 클라이언트와 서버 간에 통신을 한다.

HTTP는 클라이언트와 서버 간에 통신을 하는데 리소스를 요구하는 쪽이 클라이언트, 리소스를 제공하는 쪽이 서버가 된다.

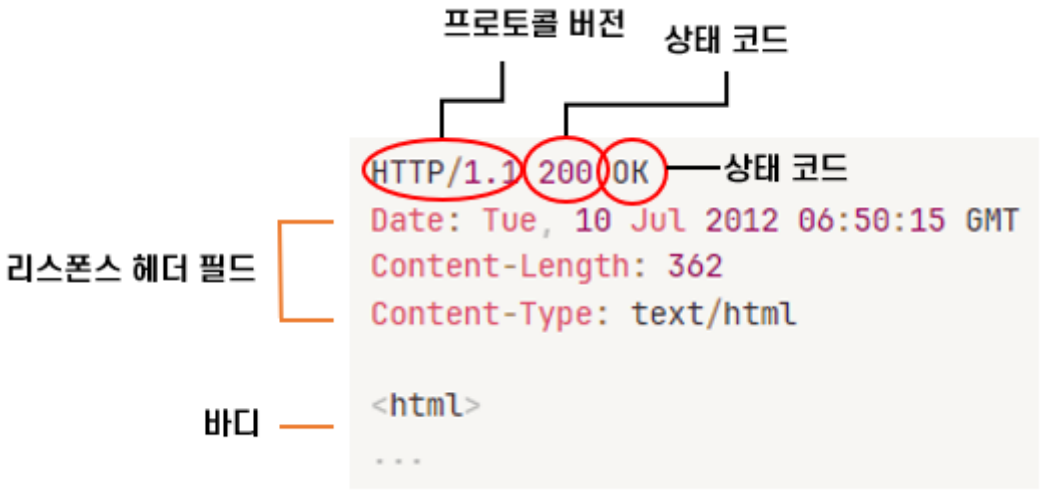
2️⃣ 리퀘스트와 리스폰스를 교환하여 성립

HTTP는 클라이언트로부터 리퀘스트(요청, Request)가 송신되며, 그 결과가 서버로부터 리스폰스(응답, Response)로 되돌아온다.
리퀘스트를 수신하지 않고 리스폰스가 발생하는 경우는 없다.

리퀘스트 메시지는 메소드, URI, 프로토콜 버전, 옵션 리퀘스트 헤더 필드와 엔티티로 구성되어 있다.



리퀘스트를 받은 서버는 처리한 결과를 리스폰스로 클라이언트에 되돌려준다.
리스폰스 메시지는 프로토콜 버전, 상태 코드와 그 상태 코드를 설명한 프레이즈, 옵션의 리스폰스 헤더 필드와 바디로 구성되어 있다.



3️⃣ HTTP는 상태를 유지하지 않는 프로토콜

HTTP는 새로운 리퀘스트가 보내질 때 마다 새로운 리스폰스가 생성된다.
즉, 과거의 리퀘스트나 리스폰스 정보를 전혀 가지고 있지 않는 스테이트리스(stateless) 프로토콜이다.

4️⃣ 리퀘스트 URI로 리소스를 식별

클라이언트는 리소스를 호출할 때 마다 리퀘스트를 송신할 때 리퀘스트 안에 URI를 리퀘스트 URI라고 불리는 형식으로 포함해야 할 필요가 있다.
리퀘스트 URI를 지정하는 방법은 다음과 같다.

- 모든 URI를 리퀘스트 URI에 포함한다.

```
GET http://hackr.jp/index.htm HTTP/1.1
```

- Host 헤더 필드에 네트워크 로케이션을 포함한다.

```
GET /index.htm HTTP/1.1
Host: hackr.jp
```

5️⃣ 서버에 임무를 부여하는 HTTP 메소드

- GET: 리소스 획득
- POST: 엔티티 전송
- PUT: 파일 전송
- HEAD: 메시지 헤더 취득(GET과 같은 기능이지만 메시지 바디는 돌려주지 않는다.)
- DELETE: 파일 삭제
- OPTIONS: 제공하고 있는 메소드의 문의
- TRACE: 경로 조사
  보안 상의 문제가 있어서 보통은 사용하지 않는다.
- CONNECT: 프록시에 터널링 요구

7️⃣ 지속 연결로 접속량을 절약

HTTP 초기 버전에서는 HTTP 통신을 한 번 할 때마다 TCP에 의해 연결과 종료를 할 필요가 있었다.
하지만 리퀘스트를 보낼 때마다 TCP 연결과 종료를 하게 되면 통신량이 늘어난다.

1. 지속 연결

HTTP/1.1과 일부 HTTP/1.0에서는 이 문제를 해결하기 위해 지속 연결(Persistent Connections)이라는 방법을 고안했다.
어느 한 쪽이 명시적으로 연결을 종료하지 않는 이상 TCP 연결을 계속 유지하는 방법이다.
오버헤드를 줄여주기 때문에 서버에 대한 부하가 경감되고 웹 페이지를 빨리 표시할 수 있다는 이점이 있다.

2. 파이프라인화

지속 연결은 여러 리퀘스트를 보낼 수 있도록 파이프라인화를 가능하게 한다.
여러 리퀘스트를 병행해서 보내는 것이 가능하기 때문에 일일이 리스폰스를 기다릴 필요가 없다.

개별 연결보다 지속 연결이 리퀘스트 완료가 빠르고, 지속 연결보다 파이프라인화 쪽이 빠르다.
특히 이 차이는 리퀘스트 수가 늘어날 수록 현저하게 나타난다.

8️⃣ 쿠키를 사용한 상태 관리

인증이 필요한 웹 페이지에서 상태 관리를 하지 않는다면 새로운 페이지로 이동할 때마다 로그인 정보를 보내든지 리퀘스트마다 추가 정보를 붙여 로그인 상태를 관리해야 하는 상황이 발생한다.
스테이트리스 프로토콜이라는 특징은 남겨둔 채, 이러한 문제를 해결하기 위해 쿠키라는 시스템이 도입되었다.

쿠키는 서버에서 리스폰스로 보내진 Set-Cookie라는 헤더 필드에 의해 클라이언트에 보존하게 된다.
다음 번에 클라이언트가 같은 서버로 리퀘스트를 보낼 때, 자동으로 쿠키 값을 넣어서 송신한다.
서버는 쿠키를 확인해서 이전 상태를 알 수 있다.