2장

HTTP는 상태를 유지하지 않는 프로토콜

- HTTP는 상태를 계속 유지하지 않는 스테이트리스(Stateless) 프로토콜
- 리퀘스트와 리스폰스를 교환하는 동안에 상태(Status)를 관리하지 않는다.
- 과거에 교환했던 리퀘스트와 리스폰스의 상태를 관리하지 않음

`많은 데이터를 매우 빠르고 확실하게 처리하는 범위성(Scalability)을 확보하기 위해서 이와 같이 간단하게 설계`
`서버의 CPU나 메모리 같은 리소스의 소비를 억제`

- 그러나, 로그인 상태를 유지해야 하는 경우가 발생
- `쿠키(Cookie)`라는 기술이 도입
- 쿠키로 인해 HTTP를 이용한 통신에서도 상태를 계속 관리할 수 있게 되었다.

쿠키를 사용한 상태 관리

- 쿠키는 리퀘스트와 리스폰스에 쿠키 정보를 추가해서 클라이언트의 상태를 파악하기 위한 시스템
- 쿠키는 서버에서 리스폰스로 보내진 Set-Cookie라는 헤더 필드에 의해 쿠키를 클라이언트에 보존
- 다음 번에 클라이언트가 같은 서버로 리퀘스트를 보낼 때, 자동으로 쿠키 값을 넣어서 송신한다.
- 서버는 클라이언트가 보내온 쿠키를 확인해서 어느 클라이언트가 접속했는지 체크하고 서버 상의 기록을 확인해서 이전 상태를 알 수 있다.

서버에 임무를 부여하는 HTTP 메소드

GET : 리소스 획득

- 리퀘스트 URI로 식별된 리소스를 가져올 수 있도록 요구

POST : 엔티티 전송

- 엔티티를 전송하기 위해 사용

PUT : 파일 전송

- 파일을 전송하기 위해서 사용

- FTP에 의한 파일 업로드와 같이, 리퀘스트 중에 포함된 엔티티를 리퀘스트 URI로 지정한 곳에 보존하도록 요구
- 단, HTTP/1.1 PUT 자체에는 인증 기능이 없어 누구든지 파일을 업로드 가능하다는 보안상의 문제도 있어서 일반적인 웹 사이트에서는 사용되지 않는다.
- 웹 애플리케이션 등에 의한 인증 기능과 짝을 이루는 경우
- REST(Representational State Transfer)와 같이 웹끼리 연계하는 설계 양식을 사용할 때 이용

HEAD : 메시지 헤더 취득

- GET과 같은 기능이지만 메시지 바디는 돌려주지 않는다.
- 유효성과 리소스 갱신 시간을 확인하는 목적 등으로 사용

DELETE : 파일 삭제

- PUT과 반대로 동작
- 단, HTTP/1.1 PUT 자체에는 인증 기능이 없어 누구든지 파일을 삭제 가능하다는 보안상의 문제도 있어서 일반적인 웹 사이트에서는 사용되지 않는다.
- 웹 애플리케이션 등에 의한 인증 기능과 짝을 이루는 경우
- REST(Representational State Transfer)와 같이 웹끼리 연계하는 설계 양식을 사용할 때 이용

OPTIONS : 제공하고 있는 메소드의 문의

- 리퀘스트 URI로 지정한 리소스가 제공하고 있는 메소드를 조사하기 위해 사용

TRACE : 경로 조사

- Web 서버에 접속해서 자신에게 통신을 되돌려 받는 루프백(loop-back)을 발생
- 리퀘스트를 보낼 때 Max-Frowards라는 헤더 필드에 수치를 포함
- 서버를 통과할 때마다 수치가 감소
- 수치가 0이 된 곳을 끝으로, 리퀘스트를 마지막으로 수신한 곳에서 상태 코드 200 OK 리스폰스를 반환
- 클라이언트는 TRACE 메소드를 사용함으로써, 리퀘스트를 보낸 곳에 어떤 리퀘스트가 가공되어 있는지 등을 조사할 수 있다.
- 프록시 등을 중계하여 오리진(Origin) 서버에 접속할 때 그 동작을 확인하기 위해서 사용

- 단, TRACE 메소드는 거의 사용되지 않는데다 크로스 사이트 트레이싱(XST)과 같은 공격을 일으키는 보안 상의 문제도 있기 때문에 보통은 사용되고 있지 않다.

CONNECT : 프록시에 터널링 요구

- 프록시에 터널 접속 확립을 요함
- TCP 통신을 터널링 시키기 위해서 아용
- 주로 SSL이랑 TLS등의 프로토콜로 암호화 된 것을 터널링 시키기 위해 사용

지속 연결(Persistent Connections)

- HTTP 초기에는 통신을 한 번 할 때마다 TCP에 의해 연결과 종료를 할 필요가 있었다.
  - 초기 당시에는 작은 사이즈의 텍스트를 보내는 정도였기 때문에 문제 X
- 인터넷이 발전한 요즘 이미지가 포함된 문서들이 대다수
  - HTML 문서에 포함되어 있는 이미지를 획득하기 위해서 여러 리퀘스트를 송신

    → TCP 연결과 종료를 하게 되는 쓸모없는 일이 발생되어 통신량이 증가

`위와 같은 문제를 해결하기 위해 지속 연결이란 방법이 고안`

지속 연결이란 ?

- 어느 한 쪽이 명시적으로 연결을 종료하지 않는 이상 `TCP 연결을 계속 유지`
- TCP 커넥션의 연결과 종료를 반복되는 오버헤드를 감소하여 `서버에 대한 부하가 경감`
  - 지속 연결은 HTTP/1.1에서부터 표준 동작

파이프라인화

- 지속 연결은 `여러 리퀘스트`를 보낼 수 있도록 파이프라인(HTTP Pipelining)화를 가능하게 한다.
- 이전에는 리퀘스트를 송신 후에 리스폰스를 수신할 때까지 기다린 뒤에 리퀘스트를 발행

  → 리스폰스를 기다리지 않고 바로 다음 리퀘스트를 발송 가능