쿠키쓰기(생성).

먼저 response메시지를 조작한다.

Response.writeHead() 는 헤드를 작성하는 코드이다. 첫번째 인자로 성공이나 에러 실패등의 메시지(200,404 등) 두번째에 객체가 들어간다.

쿠키를 key-value 형식으로 넣으면 된다.

예제에는 중복되는 같은 Set-Cookie키를 중복해서 두번사용하는데 이때 value를 배열로 주면 된다.

Response.writeHead(200,{‘Set-Cookie’ :[‘yummy\_cookie=choco’,’tasty\_cookie=strawberry’]});

웹브라우저는 request 헤더에 저장된 쿠키값을 cookie라는 헤더값을 통해 서버로 전송

생성 이후 개발자 도구의 네트워크 탭에서 response header, request header 등에서 확인 가능하며,

네트워크탭의 cookies 탭에서도 확인이 가능하다.

application탭에서는 쿠키를 삭제할 수 있다.

쿠키 읽기

console.log(request.headers.cookie); 를 통해 쿠키를 읽어볼 수 있으나 복수의 쿠키가 들어올 경우 복잡한 형태로 반환된다.

함수를 통해 제어할 수 있으나 모듈을 설치해 다뤄보도록 한다.(npm)

설치

Npm install -s cookie

Var cookie = require(‘cookie’);

Cookie 모듈은 parse 메소드를 통해 nodejs가 반환한 문자를 객체화 하여 다루기 쉽게 해준다.

Var cookies = cookie.parse(var cookies = {};//조건문 안에서 만들면 밖에서 못쓰므로 미리 만듬

if(request.headers.cookie!=undefined)//parse가 undifined를 수용못해서 미리 체크해야함

{

cookies=cookie.parse(request.headers.cookie);

} request.headers.cookie);

Console.log(cookies.yummy\_cookie)

쿠키의 지속

Session cookies – 웹 브라우저가 켜져있는 동안 유효함. 껏다키면 사라짐

Permanent cookies – 영구지속. 껏다 켜도 남아있음

Permanent cookies

cookie값에 속성으로 Max-Age 나 Expires를 설정하면 된다.

Expire – 절대수치 정해진 날 까지

Max-Age – 상대수치 현재시점을 기준으로 얼마간 살아있는지

response.writeHead(200,{'Set-cookie':[

        'yummy\_cookie=choco',

        'tasty\_cookie=strawbarry',

        `Permanent=cookies; Max-Age=${60\*60\*24\*30}`//1달

    ]});

Secure

'Secure=Secure;Secure'

개발자 도구로 보면 response에는 나오는데 request의 헤더에는 표시되지 않는다.

이 쿠키는 https 상에서만 보이기 때문 (현시점까지는 http환경이다)

https는 http+보안 이기 때문에 쿠키를 훔쳐가는 사태를 방지해준다

HttpOnly – 웹 브라워저와 웹서버가 통신할 때만 쿠키를 알 수 있다.

대충 도중에 javascript로 중간에서 장난 못치게 하려고 쓴다. 정도

Path

지금까지 예제에서의 쿠키는 최상위에 존재.

특정 디렉토리에서만 쿠키가 활성화되도록 할 때 사용

'Path=Path;Path=/cookie'

이렇게 path옵션을 지정하면 해당 디렉토리와 하위 디렉토리에서만 해당 쿠키가 활성화됨.

Domain

‘Domain=Domain;Domain=O2.org’ (== .O2.org)

O2.org의 어떤 서브도메인에서도 살아남는다.

Path는 어떤 경로에서 동작할 것인지를 제한

Domain은 어떤 도메인에서 동작할 것인지를 제한한다.

예제에서의 쿠키에서 id와 비밀번호를 가지고 있는 것은 교육용이므로 실사용은 지양해야한다