클라이언트와 서버는 http 프로토콜을 활용하여 통신을 하는데, 클라이언트가 서버에 Request 를 보내면 서버는 그에 맞는 Response 를 보냅니다. 이 과정에서 요청과 응답이 끝나면 연결이 유지되지 않고 종료되어 connectionless 한 형태를 지니고 있습니다.

그리고 http 는 서버에서 클라이언트가 보냈던 이전 기록들을 가지고 있지 않기 때문에 stateless 의 성격을 지니고 있습니다. 서버는 클라이언트로부터 요청이 오면 그저 요청에 대한 응답을 보내주고 끝나기 때문에 무슨 클라이언트 인지, 그리고 이전에 통신을 했던 클라이언트인지에 대한 정보를 전혀 모릅니다.

이러한 http 프로토콜의 특성을 극복하기 위해 쿠키와 세션을 활용하여 각각의 클라이언트를 식별하고, 일정 시간 동안 클라이언트의 접속을 유지할 수 있게 됩니다.

먼저 쿠기에 대해서 살펴보면 쿠키는 클라이언트인 브라우저에 저장 되는 텍스트 파일입니다. key 와 value 의 쌍 값 형태로 저장되고 쿠키 파일에는 쿠키 이름, 쿠키 값, 만료시간, 전송할 도메인명, 전송할 경로, 보안연결여부, httpOnly 여부로 구성되어 있습니다. 하나의 도메인은 최대 20 개의 쿠키를 가질 수 있고, 최대 4KB 까지 저장이 가능합니다.

쿠키의 사용 목적으로는 크게 3 가지로 나눠볼 수 있습니다. 첫째는 세션 관리입니다. 클라이언트의 로그인, 사용자 이름, 접속 시간, 장바구니 등의 정보를 저장합니다. 둘째는 개인화입니다. 쿠키를 활용하여 사용자마다 개인화된 정보를 보여주는데 사용합니다. 마지막으로는 트래킹입니다. 사용자의 행동과 패턴을 분석하고 기록하여 이를 다른 부분에서 활용합니다.

쿠키의 작동 원리에 대해서 살펴보겠습니다. 먼저 클라이언트는 서버에 로그인 요청을 보냅니다. 그러면 서버는 클라이언트가 보낸 로그인 요청을 토대로 아이디와 비밀번호를 검사하여 유효성을 확인하고, Response 헤더에 set-cookie 라는 속성을 통해 쿠키를 추가하여 응답을 보냅니다. 클라이언트는 이후 서버에 Request 를 전송할 때 자동으로 Request 헤더에 쿠키 정보를 포함하여 전송함으로써 쿠키를 활용하게 됩니다.

마지막으로 쿠키의 종류에 대해서 설명드리겠습니다. 총 4 개의 쿠키로 나눠볼 수 있는데 Session cookie, Persistent cookie, Secure cookie, Third-party cookie 가 있습니다. Session cookie 는 쿠키의

만료시간을 지정하고 클라이언트의 메모리에만 저장하며, 브라우저를 종료하면 자동으로 삭제되는 쿠키입니다. 이를 활용한 예시로는 로그인 상태 유지, 장바구니 유지가 있습니다. 한마디로 세션을 구현하는데 활용하는 쿠키입니다. Persistent cookie 는 브라우저의 종료 여부와 상관없이 장기간동안 유지되는 쿠키입니다. 이는 자동 로그인, 사이트 방문 기록 저장, 최근 본 상품 저장 등에서 활용됩니다. Secure cookie 는 HTTPS 프로토콜에서만 사용하는 쿠키이며, 기존 쿠키는 텍스트 파일 형식으로 따로 보안이 적용되지 않은 취약점이 있습니다. 이를 보완하여 쿠키 정보를 암호화하여 주고 받는 쿠키입니다. Third-party cookie 는 방문한 도메인과 다른 도메인의 쿠키입니다. 이를 활용한 예시로는 쿠팡에서 검색했던 제품이 다른 웹사이트의 광고란에 뜨는 것을 예시로 들 수 있습니다.

이번에는 세션에 대해서 살펴보겠습니다.
세션은 일정 시간 동안 같은 클라이언트로부터 들어오는 일련의 요구를 하나의 상태로 인식하고, 그 상태를 유지시키는 기술입니다. 웹서버에 웹 컨테이너의 상태를 유지하기 위한 정보를 저장하며, 세션을 사용할 때 사용되는 쿠키가 앞서 설명드렸던 Session Cookie 입니다. 각 클라이언트에 대해 고유한 세션 ID 를 부여함으로써, 클라이언트를 구분하고 클라이언트와 서버의 접속을 유지하는데 사용됩니다. 세션은 저장 데이터가 클라이언트 쪽이 아닌 웹 서버에 저장되므로 웹서버 용량이 제공하는 한에서 거의 무제한으로 사용할 수 있습니다. 그리고 데이터가 웹서버에 저장되므로 클라이언트의 브라우저에 저장되는 쿠키보다 보안적으로 띄어납니다.

세션의 작동 원리에 대해서 설명드리겠습니다. 먼저 클라이언트가 서버에 로그인 요청을 보내게 되면, 서버는 클라이언트 로그인 요청의 유효성 검증을 실시한 후, 고유한 Session ID 를 생성하여 웹서버에 저장합니다. 그리고 클라이언트에 Response 를 보낼 때 헤더에 session ID 를 쿠키에 추가하여 전송합니다. 클라이언트는 이후 서버에 요청을 보낼 때 전달 받았던 session ID 를 쿠키에 넣어서 Request 를 전송합니다. 서버에서는 Request 헤더 안의 session ID 를 웹서버에서 유효성 검증을 실시한 후 요청을 처리 및 응답합니다.

세션의 적용 사례를 보게 되면 먼저 사용자 로그인 상태 유지가 있습니다. 예시로 네이버 웹사이트를 들어보면, 메인 페이지에서 로그인을 하면 웹서버는 세션을 생성하고 사용자 정보를 저장해둡니다. 그리고 사용자가

네이버 쇼핑이나, 이메일 등의 다른 페이지로 이동하게 되면 로그인은 유지된 상태로 해당 서비스를 바로 사용할 수 있게 됩니다. 이 과정에서 클라이언트의 Session Cookie 와 웹서버에 저장된 Session ID 를 활용하여 사용자의 로그인 상태를 계속해서 유지할 수 있게 됩니다. 두번째로는 장바구니 기능이 있습니다. 사용자가 로그인하지 않아도 장바구니에 상품을 담을 수 있도록 세션을 활용합니다. 사이트를 새로고침하거나 다른 페이지로 이동해도 장바구니의 데이터를 유지할 수 있습니다. 마지막으로 사용자 맞춤 설정 저장입니다. 사용자가 다크 모드나 언어 설정 등을 선택하면 세션에 저장하여 유지 가능하게 해줍니다.

아래의 도표처럼 쿠키와 세션의 차이점을 비교해 보았습니다.

마지막으로 세션과 쿠키를 모두 사용하는 이유가 무엇일까요?
세션이 쿠키에 비해 보안적으로 우수하지만, 웹서버에 저장되므로 서버 자원에 한계점이 존재하고, 서버를 통해 데이터를 접근하기 때문에 속도 저하의 문제가 있습니다. 이에 반해 쿠키는 클라이언트 브라우저에 저장되므로 접근 속도가 빨라서, 쿠키와 세션을 적절히 병행하여 사용하면 웹서버 자원의 낭비를 방지하며, 웹사이트의 속도와 성능을 높일 수 있습니다.

이상으로 발표를 마치며 질문을 받겠습니다.