인터넷에서 사용되는 여러 기술 : HTTP 1

조인준
KBS 미디어기술연구소
차장

지난 두 차례의 내용을 통해서 인터넷을 통해 매일 방문하게 되는 웹페이지를 브라우저상에서 표시하기 위한 언어인 HTML(HyperText Markup Language)에 대해 알아보았습니다. 이번 편에서는 HTML을 전달하는 프로토콜인 HTTP(HyperText Transfer Protocol)에 대해 이야기를 해보려 합니다.

그림 1. 클라이언트의 HTTP 요청과 서버의 HTTP 응답

우리가 브라우저를 통해 특정 사이트에 접속하면 웹페이지가 화면에 표시되는데 이는 해당 사이트의 서버에 브라우저가 접속해서 HTML로 된 웹페이지를 전송받아 화면으로 출력해주기 때문에 가능한 것입니다. 이를 조금 더 자세히 표현하면 [그림 1]과같이 브라우저가 클라이언트로 동작하여 웹사이트의 URL을 통해 웹페이지의 HTML을 요청하는 HTTP 요청(HTTP Request)을 해당 웹사이트의 서버로 보내고, 이 요청을 받은 서버가 클라이언트에게 HTTP 응답(HTTP Response)을 통해 웹페이지의 HTML을 회신하면 이를 수신한 브라우저가 화면에 HTML에 기술된 대로 웹페이지를 출력하는 것입니다.

웹페이지 HTML 전송에 대한 예로 이야기를 시작했지만, 일반적으로 표현하면 HTTP는 인터넷에서 데이터를 전송하기 위한 프로토콜입니다. 인터넷의 핵심요소인 HTTP의 특징을 정리하면 아래와 같습니다.

• 클라이언트/서버 모델

HTTP는 클라이언트/서버 모델을 기반으로 동작합니다. 웹브라우저 등의 클라이언트를 통해 웹페이지의 HTML이나 각종 데이터 등을 서버에 요청하고, 서버는 해당 요청을 수신한 후 요청에 대한 처리와 응답을 생성하여 클라이언트에게 회신하는 구조를 바탕으로 동작합니다.

• 비상태 프로토콜(Stateless Protocol)

앞서 HTTP는 요청과 응답으로 동작한다고 말씀드렸습니다. HTTP가 비상태 프로토콜이라고 하는 것은 이 요청들이 서로 독립적이고 이전 요청과 이후 요청이 서로 영향을 주고받지 않는 것을 의미합니다. 사실 너무 추상적인 설명이라서 무슨 말인지 이해하기가 매우 어렵습니다. 그래서 정확하지는 않지만 그래도 무슨 의미인지 느낌이라도 전해 드리기 위해 동전 던지기를 예로 들어보겠습니다. 동전을 반복해서 던질 때 이번에 앞면이 나오든 뒷면이 나오든 다음번의 앞면/뒷면의 결과에 아무 영향을 미치지 않는 것처럼 이전 요청과 현재 요청이 서로에게 어떤 영향도 주고받지 않기에 서버가 클라이언트의 요청들을 기억하여 어떤 맥락을 추적할 필요가 없는 프로토콜이라는 것이 비상태 프로토콜입니다.

아마 많은 독자분께서 고개를 끄덕이시다가 갑자기 다음과 같은 의문에 사로잡히실 것 같습니다. 클라이언트의 요청들이 서로 독립적이고 서버가 이 요청들의 히스토리를 관리하지 않는다면 만약 회원제 인터넷 쇼핑몰 서버가 어떻게 회원 로그인 요청 처리 후 이 상태를 기억하여 로그인 상태에서만 방문 가능한 페이지들에 대한 요청에 응답할 수 있는가? 이와 같은 경우를 대비하여 상태 관리는 쿠키(Cookies)나 토큰(Token) 등을 사용하여 웹 애플리케이션 수준에서 처리됩니다. 쿠키나 토큰에 대해 간략히 설명하기 위해서는 우선 세션(Session)이라는 개념에 대한 설명이 필요합니다.

☑ 세션(Session)

세션은 클라이언트와 서버 사이의 논리적 연결 상태를 말합니다. 이 '논리적'이라는 표현이 모호할 수 있어서 엄밀하지는 않지만 그래도 나름의 설명을 드리자면 A와 B가 우리가 어릴 적 가지고 놀던 실 전화기를 서로의 귀에 대고 이미 물리적으로는 통신 가능한 상태에 있다고 가정합니다. 논리적 연결은 A가 실 전화기에 대고 "이제부터 내가 내일 숙제를 알려줄게!" 라고 이야기하면 B가 "그래!"하고 응답하여 무언가 용건에 대한 상호작용을 시작한 상태가 된 것을 의미합니다. 그리고 A가 숙제를 다 알려주고 "이제 끝!"이라고 했을 때 B가 "그래!" 하면 논리적 연결이 끝난 상태입니다. 이를 클라이언트/서버 관점으로 다시 이야기하면 클라이언트가 로그인을 하면 서버는 클라이언트에 대한 세션을 생성하고, 세션을 구분하기 위한 ID를 쿠키를 통해 클라이언트에게 전달합니다. 클라이언트는 이후 요청을 보낼 때 서버가 발급해 준 세션 ID를 서버에 전송하고 서버는 이를 통해 해당 세션을 식별하고 연결 상태를 관리합니다.

☑ 쿠키(Cookies)

쿠키는 서버가 클라이언트에게 보내는 소용량의 데이터입니다. 소용량이라서 쿠키라는 이름을 붙인 것 같습니다. 쿠키는 서버가 발행하여 응답으로 전달하고 클라이언트는 이를 받아서 저장한 후 이후 요청 때 해당 쿠키를 서버에 전송합니다. 쿠키에는 클라이언트를 식별할 수 있는 정보인 세션 ID, 사용자 ID 및 표시 언어 등 개인화 서비스를 위한 정보 등이 포함될 수 있습니다. 서버는 클라이언트가 보낸 쿠키를 읽어 해당 클라이언트의 상태를 식별합니다. 쿠키는 일반적으로 암호화되지 않은 텍스트 정보로 저장되어 보안 위협에 노출될 가능성이 있습니다.

☑ 토큰(Token)

토큰은 주로 인증 정보를 저장하는 데 사용됩니다. 클라이언트가 로그인을 하면 서버는 토큰을 생성하여 클라이언트에게 응답합니다. 클라이언트는 토큰을 저장하여 이후 HTTP 요청 시 토큰을 포함하여 서버에 보내며 서버는 토큰을 검증하여 클라이언트를 인증하는 방식으로 상태를 관리합니다. 토큰은 일반적으로 서명이나 암호화를 통해 보호될 수 있습니다. 서명된 토큰은 위변조를 방지하고, 암호화된 토큰은 데이터를 안전하게 전송할 수 있습니다.

• TCP/IP 계층 모델의 응용 계층

HTTP는 TCP/IP 계층모델에서 5번째 계층인 응용계층의 프로토콜입니다. 전송계층 프로토콜 위에 올라가게 되며 거의 대부분의 경우 HTTP 전송에는 TCP를 이용하고 있습니다. 다시 말해서 HTTP는 전송에 관한 내용을 전혀 포함하지 않는다는 것입니다. 전송에 관한 것은 모두 TCP 레벨에서 처리가 되고 HTTP는 요청을 보내고 이에 대한 응답을 받는 것에 모든 초점이 맞추어져 있습니다.

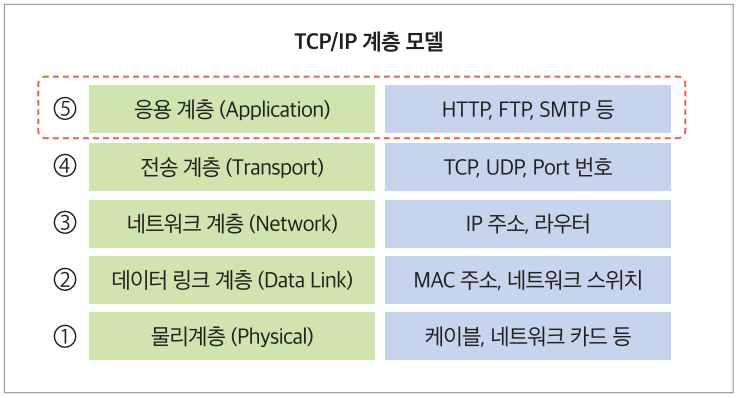

그림 2. TCP/IP 계층 모델에서의 HTTP

• HTTP 메서드(Method)

클라이언트는 HTTP를 사용하여 서버에 요청을 보냅니다. HTTP 메서드는 클라이언트가 서버에 요청의 목적이나 내용을 알리는 방법입니다. 다음은 일반적인 HTTP 클라이언트 메서드입니다.

☑ **GET**

GET 메서드는 서버로부터 데이터를 가져오기 위한 요청에 사용합니다. 클라이언트가 GET 요청을 보내면 서버는 해당 데이터를 반환합니다. GET 요청은 URL을 통해 요청 대상을 지정하고, 필요한 경우 쿼리(Query)의 매개변수를 사용하여 추가적인 정보를 전달할 수 있습니다.

☑ **POST**

POST 메서드는 서버에 데이터를 제공하기 위한 요청에 사용합니다. 주로 신규 데이터를 서버에 등록하거나 파일 업로드와 같은 작업에 사용됩니다. 클라이언트는 POST 요청의 본문(다음 편에서 설명)에 데이터를 포함하여 서버에 전송합니다. 서버는 이를 처리하고 결과에 관한 내용을 응답하며 새로운 데이터가 생성된 URL(Uniform Resource Locator)을 전달하기도 합니다.

☑ **PUT**

PUT 메서드는 서버에 새로운 데이터를 갱신하기 위해 사용합니다. 해당 데이터가 있으면 대체하고 없으면 생성하기도 합니다. 클라이언트는 PUT 요청 본문에 대상이 되는 데이터에 관한 위치 정보를 포함하여 서버에 전송합니다. 서버는 이를 처리하고 결과를 응답으로 반환합니다.

☑ **DELETE**

DELETE 메서드는 서버에서 특정 데이터를 삭제하기 위한 요청을 전송합니다. 클라이언트가 DELETE 요청을 보내면 서버는 해당 데이터를 삭제하고, 삭제 결과를 응답합니다.

위에서 설명한 메서드 외에도 OPTIONS, TRACE, PATCH 등 다양한 HTTP 메서드가 있습니다. 클라이언트는 이러한 메서드들을 사용하여 서버와 통신을 수행하고, 서버의 응답을 처리하여 웹 애플리케이션을 구축하거나 웹에서 데이터를 관리합니다.

• HTTP 상태 코드(HTTP Status Codes)

HTTP 상태 코드(HTTP Status Codes)는 클라이언트가 서버로부터 받는 HTTP 응답의 상태를 나타내는 숫자입니다. 상태 코드는 클라이언트에게 요청의 성공, 실패 및 처리에 관한 다양한 상황을 알려줍니다. 세 자리 숫자로 구성된 상태 코드는 각각의 범위에 따라 의미가 나누어지며 클라이언트는 서버로부터 받은 상태 코드를 분석하여 적절한 조치를 취하거나 사용자에게 상황을 알릴 수 있습니다. 또한, 웹서버 등의 문제 해결에도 유용한 정보를 제공합니다. 일반적인 HTTP 상태 코드는 다음과 같습니다.

☑ **1xx (Informational)**

요청이 수신되었거나 처리 중임을 나타냅니다. 클라이언트의 추가 동작이 필요한 경우도 있을 수 있습니다.

☑ **2xx (Success)**

요청이 성공적으로 처리되었음을 나타냅니다. 대표적으로 200은 OK, 201은 요청한 데이터 생성 성공 등을 나타냅니다.

☑ **3xx (Redirection)**

요청을 완료하기 위해 추가 동작이 필요함을 나타냅니다. 클라이언트는 다른 위치로 원하는 데이터를 요청해야 할 수도 있

습니다. 301은 Moved Permanently로 해당 URI(Uniform Resource Identifier)가 다른 주소로 바뀌었음을 나타내고, 302는 Found로 요청한 URI가 일시적으로 변경되었음을 의미하며, 304는 Not Modified로 클라이언트에게 이전 응답의 캐시 버전을 사용해도 됨을 알려줍니다.

☑ **4xx (Client Error)**

클라이언트 측의 요청에 문제가 있음을 나타냅니다. 매개변수 오류, 인증 실패, 요청 데이터의 누락 등이 해당합니다. 예로 400은 Bad Request로 요청의 매개변수나 형식 등에 문제가 있음을 알리며, 401은 Unauthorized로 클라이언트가 요청한 응답을 받기 위해서는 인증을 해야 한다는 의미이며, 404는 Not Found로 요청한 데이터가 없음을 나타냅니다.

☑ **5xx (Server Error)**

서버에서 요청을 처리하는 동안 오류가 발생했음을 나타냅니다. 예로 500은 Internal Server Error로 웹서버에 문제가 있음을 나타내고, 503은 Service Unavailable로 서버가 기술적 문제 등으로 요청을 처리할 수 없음을 알려줍니다.

• HTTP Headers

HTTP 헤더(HTTP headers)는 HTTP 요청과 응답 메시지에 포함되는 추가 정보입니다. 헤더는 클라이언트와 서버 간의 통신을 보다 세밀하게 제어하고 필요한 메타데이터를 전달하기 위해 사용됩니다. 헤더는 키/값의 Pair 형태로 구성되며, 클라이언트나 서버는 헤더 필드를 통해 원하는 정보를 포함할 수 있습니다. 헤더는 요청과 응답 메시지 모두에 포함되며, 클라이언트와 서버는 이를 해석하여 적절한 동작을 수행하고 추가 정보를 전달합니다. 다음은 HTTP 헤더의 구성 요소에 대한 설명입니다:

☑ **General Header**

요청과 응답 모두에 적용되는 일반적인 헤더입니다.

☑ **Request Header**

클라이언트가 서버로 요청을 보낼 때 사용되는 헤더입니다. 클라이언트는 요청 헤더를 통해 서버에 추가 정보를 전달할 수 있습니다.

☑ **Response Header**

서버가 클라이언트에게 응답을 반환할 때 사용되는 헤더입니다. 서버는 응답 헤더를 통해 클라이언트에게 제공하는 데이터와 관련된 정보를 전달합니다.

☑ **Entity Header**

요청 또는 응답 본문에 대한 정보를 포함하는 헤더입니다. 일반적으로 본문의 크기, 인코딩, 변경 시간 등을 나타냅니다.

지금까지 HTTP의 개요에 관해 설명해 드렸습니다. 다음 편에서는 HTTP 요청 및 응답에 관한 구체적 예를 통해서 HTTP의 동작에 대한 이해를 높이도록 하겠습니다.

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다.
본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.