http

⑦ HTTP에 대해 설명해주세요.

- HTTP는 Hypertext Transfer Protocol의 약자로 웹 통신을 위한 응용 프로토콜의 한 종류입니다.
- HTTP는 웹 서버와 클라이언트의 통신에 사용됩니다.
- HTTP는 요청과 응답으로 구성됩니다.
- HTTP는 상태가 없는 프로토콜입니다. 따라서 상태를 유지해야 하는 경우 쿠키와 세션 등의 기술을 활용합니다.
- HTTP는 헤더와 바디로 구분됩니다. 헤더는 HTTP Method, 요청 서버 주소, 컨텐츠 타입 등의 정보를 담고 있습니다.

⑦ http는 stateless한데 3way handshake는 상태 기반 아닌가요?

HTTP와 TCP의 3-way handshake는 서로 다른 레벨에서 동작하는 프로토콜이며, 각각의 특성이 다릅니다.

1. HTTP (HyperText Transfer Protocol):

- HTTP는 애플리케이션 레벨의 프로토콜로, 웹 서버와 클라이언트 간의 요청과 응답을 처리합니다.
- HTTP는 "stateless"하다고 설명됩니다. 이는 각 요청과 응답이 독립적이라는 의미로, 서버는 이전에 수행된 요청에 대한 정보를 기억하지 않습니다. 따라서, 각 요청은 모든 필요한 정보를 포함해야 합니다.
- 쿠키나 세션과 같은 메커니즘을 통해 상태 정보를 유지할 수 있지만, 이는 HTTP 프로토콜의 기본적인 특성 위에 구축된 것입니다.

2. TCP (Transmission Control Protocol):

- TCP는 전송 레벨의 프로토콜로, 데이터의 신뢰성 있는 전송을 보장합니다.
- TCP 연결을 시작할 때, 3-way handshake라는 과정을 거칩니다. 이 과정은 클라이언트와 서버 간에 연결을 초기화하고, 양쪽 모두 데이터 전송을 위한 준비가 되었음을 확인하는 것입니다.
- 3-way handshake는 상태 기반의 과정입니다. SYN, SYN-ACK, ACK 패킷을 주고받으며, 이 과정을 통해 연결의 상태를 "ESTABLISHED"로 변경합니다.

⑦ HTTP/3에 대한 특징 간략히 설명해주세요.

1. TCP대신 QUIC 프로토콜을 사용합니다. QUIC 프로토콜은 UDP 기반으로 동작하며 빠른 연결 속도를 제공합니다.
2. 개선된 패킷 재전송 매커니즘으로 패킷 손실에 대한 대응이 빠릅니다.
3. IP가 바뀌어도 연결 유지가 가능합니다. 특히 동영상 서비스를 이용하거나, 와이파이 변경 등의 상황에 효과적으로 작용합니다.

⑦ QUIC에 대해 간략하게 설명해주세요.

1. **UDP 기반**: QUIC는 TCP가 아닌 UDP 위에서 구현됩니다. 이를 통해 연결 설정에 필요한 지연 시간을 줄이고, 네트워크 변경에 더 유연하게 대응할 수 있습니다.
2. **빠른 연결 설정**: QUIC는 연결 설정을 위해 3-way handshake 대신 더 효율적인 방법을 사용하여, 특히 이미 연결된 서버와의 재연결 시에는 거의 즉시 연결을 설정할 수 있습니다.
3. **내장된 보안**: QUIC는 기본적으로 TLS 암호화를 사용하여 데이터를 보호합니다. 이로 인해 중간자 공격 (Man-in-the-Middle Attack)에 대한 보호를 강화하고, 연결 설정 시의 보안 절차를 간소화합니다.
4. **향상된 오류 제어**: QUIC는 패킷 손실이 발생했을 때 전체 스트림이 차단되는 TCP의 "head-of-line blocking" 문제를 해결합니다. 이는 각 스트림이 독립적으로 동작하기 때문에 가능합니다.
5. **동적인 혼잡 제어**: QUIC는 네트워크 혼잡 상황을 더 잘 감지하고 대응할 수 있는 동적인 혼잡 제어 메커니즘을 포함하고 있습니다.
6. **멀티플렉싱**: QUIC는 여러 개의 독립적인 스트림을 동일한 연결 위에서 동시에 운영할 수 있습니다. 이로 인해 웹 페이지의 여러 리소스를 효율적으로 로드할 수 있습니다.
7. **향상된 연결 이동**: 사용자의 네트워크 환경이 변경될 때 (예: Wi-Fi에서 모바일 데이터로 전환) QUIC 연결은 더 빠르게 복구되거나 이동될 수 있습니다.

https

⑦ HTTPS

HTTPS는 "HyperText Transfer Protocol Secure"의 약자로, 웹 트래픽을 암호화하여 전송하는 프로토콜입니다. 기본적으로 HTTP의 기능 위에 SSL/TLS 프로토콜을 사용하여 보안 계층을 추가함으로써 동작합니다. 이로 인해 데이터의 기밀성, 무결성, 그리고 인증이 보장됩니다. HTTPS의 주요 특징 및 장점은 다음과 같습니다:

1. **기밀성**: HTTPS는 웹 트래픽을 암호화하여 중간자 공격자 (Man-in-the-Middle attacker)가 데이터를 도청하더라도 실제 내용을 알아볼 수 없게 합니다.
2. **무결성**: 데이터가 송신자로부터 수신자까지 전송되는 도중에 변경되거나 손상되지 않았음을 보장합니다.
3. **인증**: 사용자는 웹 사이트가 그것을 주장하는 서버에 의해 제공되었음을 확신할 수 있습니다. 이는 특히 사용자가 웹 사이트에 개인 정보나 신용 카드 정보를 입력할 때 중요합니다.
4. **SEO (Search Engine Optimization) 장점**: 여러 검색 엔진들은 HTTPS를 사용하는 웹사이트에 더 높은 순위를 부여하는 경향이 있습니다.
5. **신뢰성**: 사용자들은 주소창에 표시되는 자물쇠 아이콘을 통해 사이트가 안전하다는 것을 알 수 있습니다. 이로 인해 사용자의 신뢰도가 향상될 수 있습니다.

HTTPS를 구현하기 위해서는 웹 서버에 SSL/TLS 인증서를 설치해야 합니다. 이 인증서는 인증서 기관 (Certificate Authority, CA)에 의해 발급되며, 웹 서버의 신원을 확인하고 통신을 암호화하는 데 사용됩니다.

최근에는 웹의 보안을 강화하기 위한 움직임의 일환으로 많은 웹사이트와 웹 브라우저가 HTTPS를 기본으로 사용하거나 적극 권장하고 있습니다.

SSL이란?

- 보안 소켓 계층(Secure Sockets Layer, SSL)
- 컴퓨터 네트워크에 통신 보안을 제공하기 위해 설계된 암호 규약
- SSL은 웹사이트와 브라우저(혹은, 두 서버) 사이에 전송된 데이터를 암호화하여 인터넷 연결 보안을 유지하는 표준 기술임. 그래서 전송된 데이터는 중간에서 누군가 훔쳐 낸다고 하더라도 데이터가 암호화되어있기 때문에 해독할 수 없음

SSL 인증서

- SSL(Secure Sockets Layer) 인증서는 웹사이트의 신원을 확인하는 디지털 인증서
- SSL인증서는 보안통신을 하기 위한 전자 파일입니다. SSL인증서를 서버에 설치함으로써 SSL 프로토콜을 사용하여 보안 통신을 할 수 있게 되는 것
- 증명서는 신뢰할 수 있는 제 3 자 기관에 의해 발행되는 것이기 때문에 서버나 클라이언트가 실재하는 사실을 증명

TLS

- 전송 계층 보안(Transport Layer Security, TLS)
- TLS은 SSL이 점차 보편화되면서 국제 표준화 기구(IETF)에서 이를 관리하기 시작했는데, 이 때 SSL 표준화 되면서 TLS라는 이름으로 변경

Restful

Restful

- REST 아키텍처 스타일을 만족하는 서비스를 뜻하는 말
- REST는 Representational State Transfer의 약자로 리소스와 URI, 행위와 HTTP Method를 대응하여 자원에 대한 표현을 통해 통신하는 아키텍처 스타일을 말한다.
- REST의 특징
  1. 서버-클라이언트 구조 : 서버는 API 제공, 클라이언트는 요청에 대한 처리로 역할을 구분할 수 있다.
  2. Stateless : 각각의 요청에 대해 독립적이다. 서버의 부하를 줄일 수 있다는 장점을 갖는다.
  3. Cacheable : 캐시 기능을 사용할 수 있어, 반복된 요청을 효율적으로 처리 가능하다.
  4. Uniform Interface : URI와 리소스가 대응되고, HTTP Method를 통해 동작을 정의한다. 이를 통해 각 메시지를 통해 직관적으로 이해 가능하다.
  5.

REST의 장단점

장점

1. 별도의 인프라를 구축할 필요가 없다.
2. 클라이언트와 서버의 역할을 분리할 수 있다.
3. 플랫폼 독립적이다.
4. 직관적인 사용이 가능하다.

단점

1. 표준이 존재하지 않는다.
2. HTTP Method인 CRUD에 대한 간단한 행위만 정의 가능하다.
3. RDBMS

왜 REST API여야만 하는가

- 시스템을 완전히 통제할 수 있다면 REST에 시간을 낭비하지 말라
- Uniform Interface는 기본적으로 비효율적
- 표준화된 형식으로 데이터를 전달
- 상황에 따라 최적이 아닐 수 있음

우아한 테크톡 - REST-API
Interview Question for Beginner
그런 REST API 로 괜찮은가?

HATEOAS

- Hypermedia As The Engine of Application State의 약자로 REST Api를 사용하는 **클라이언트가 전적으로 서버와 동적인 상호작용이 가능하도록 하는 것**을 의미
- **Hypermedia (링크)를 통해서 애플리케이션의 상태 전이가 가능해야 한다.**
- **또한 Hypermedia (링크)에 자기 자신에 대해한 정보가 담겨야 한다.**