1. HTTP와 HTTPS
2. WAS
3. DB서버

HTTP와 HTTPS

- HTTP(Hypertext transfer protocol)

인터넷에서 웹 서버와 사용자의 인터넷 브라우저 사이에 문서를 전송하기 위해 사용되는 통신규약.
HTTP를 이용하면 사용자는 다양한 응용 프로그램에 접근하여 텍스트 / 그래픽 / 애니메이션을 볼 수 있다.

- 프로토콜

정보기기간에 정보를 주고받을 때, 이를 원활하게 하기 위하여 정한 여러 가지 통신규칙과 방법. 즉, 통신규약을 의미

- 통신규약

상호간의 전달방식, 통신방식, 주고받는 자료의 형식, 오류 검출방식 등에 대하여 정하는 것.
컴퓨터의 기종이 다르면 통신규약도 다르기 때문에 표준 프로토콜(TCP/IP) 을 설정하여 통신망을 구축해야 한다.

- HTTPS(Hypertext Transfer protocol over Secure Sockey Layer, HTTP over TLS, HTTP over SSL)

HTTP의 보안이 강화된 버전.

- TLS(Transport Layer Security)

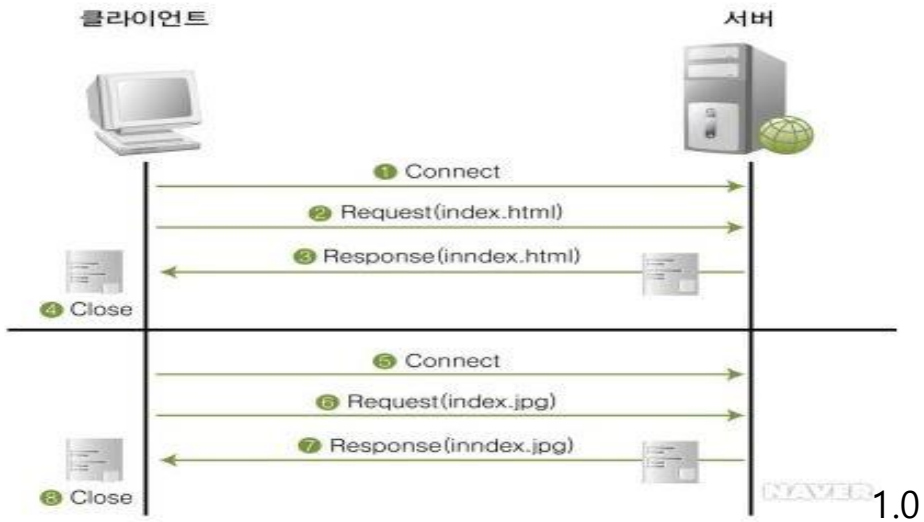
컴퓨터 네트워크에 통신 보안을 제공하기 위해 설계된 암호 규약 프로토콜.

- SSL(Secure Sockets Layer)

인터넷에서 데이터를 안전하게 전송하기 위한 인터넷 통신 규약 프로토콜.

HTTP 0.9 버전과 그 이후

0.9버전



1.0 이후 버전

0. 클라이언트가 웹 브라우저를 이용해 서버에 연결을 요청하면, 서버는 클라이언트에 대해 서비스를 준비한다.

1. 서버가 준비 상태가 된다.
2. 클라이언트는 읽고자 하는 문서를 서버에 요청한다.
3. 서버는 클라이언트가 요청한 문서를 전송하고
4. 연결을 끊는다.
5. 1.0 버전 이후로 한번의 connect 후에 Request / Response 를 반복할 수 있게 되었다.

*http 0.9 1.0 1.1

https://developer.mozilla.org/ko/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP

*http 1.1 버전과 2.0 버전의 차이

- https://medium.com/@shlee1353/http1-1-vs-http2-0-%EC%B0%A8%EC%9D%B4%EC%A0%90-%EA%B0%84%EB%8B%A8%ED%9E%88-%EC%82%B4%ED%8E%B4%EB%B3%B4%EA%B8%B0-5727b7499b78

HTTP Transaction

- 요청을 보내고(request) 응답을 받는(response) 과정이며, 연결 당 하나의 트랜잭션을 수행한다.

- 웹 브라우저는 서버와 TCP connection을 맺는다.
- 웹 브라우저는 서버에 HTTP 요청을 보낸다.

- 서버는 클라이언트와 TCP connection을 맺는다.
- 서버는 클라이언트에 HTTP 응답을 보낸다.

- HTTP Transaction
- https://m.blog.naver.com/PostView.nhn?blogId=agapeuni&logNo=60155279671&proxyReferer=https:%2F%2Fwww.google.com%2F

HTTP Request

HTTP Request는 웹 서버에 데이터를 요청하거나 전송할 때 보내는 패킷(데이터 블록)으로, 주로 GET, POST와 같은 메소드를 사용한다.

• GET 방식

- GET 방식은 가장 일반적인 HTTP Request 형태로 데이터가 HTTP Request Message의 Header 부분의 URL(Uniform Resource Locator)에 담겨서 전송된다.
- GET 방식은 데이터가 주소 입력란에 표시되기 때문에, 최소한의 보안도 유지되지 않는, 보안에 매우 취약한 방식이다.

• POST 방식

- POST 방식은 URL에 요청 데이터를 기록하지 않고 HTTP 헤더에 데이터를 전송하기 때문에 GET 방식에서의 ? 가 존재하지 않는다.
- POST 방식은 내부의 구분자가 각 파라미터(이름과 값)를 구분하며, 서버가 각 구분자에 대한 내용을 해석하여 데이터를 처리하기 때문에 GET 방식에 비해 상대적으로 처리 속도가 늦다.
- POST 방식에서는 인수 값을 URL을 통해 전송하지 않기 때문에 다른 이가 링크를 통해 해당 페이지를 볼 수 없다.

• 기타 방식

*https://terms.naver.com/entry.nhn?docId=3431904&cid=58437&categoryId=58437&expCategoryId=58437

HTTP Response

HTTP Response는 클라이언트의 HTTP Request에 대한 응답 패킷이다. 서버에서 쓰이고 있는 프로토콜 버전, Request에 대한 실행 결과 코드 및 간략한 실행 결과 설명문(OK 등)에 대한 내용이 담겨 있다.

- 1xx : 정보
- 2xx : 성공
  - 200 : OK. 요청 성공
  - 201 : Created. 생성 요청 성공
  - 202 : Accepted. 요청 수락(처리 보장X)
  - 204 : 성공했으나 돌려줄게 없음
- 3xx : 리다이렉션
  - 300 : Multiple choices. 여러 리소스에 대한 요청 결과 목록
  - 301,302,303 : Redirect. 리소스 위치가 변경된 상태
  - 304 : Not Modified. 리소스가 수정되지 않았음
- 4xx : 클라이언트 오류
  - 400 : Bad Request. 요청 오류
  - 401 : Unauthorized. 권한없음
  - 403 : Forbidden. 요청 거부
  - 404 : Not Found. 리소스가 없는 상태
- 5xx : 서버 오류
  - 500 : Internal Server Error. 서버가 요청을 처리 못함
  - 501 : Not Implemented. 서버가 지원하지 않는 요청
  - 503 : Service Unavailable. 과부하 등으로 당장 서비스가 불가능한 상태

교차 출처 리소스 공유
(Cross-Origin Resource Sharing, CORS)

- HTTP 요청은 Cross-Site HTTP Requests가 가능하다.
- <img> 태그로 다른 도메인의 이미지 파일을 가져오거나, <script> 태그로 다른 도메인의 라이브러리를 가져오는 것이 가능하다.
- 하지만 <script></script>로 둘러싸여 있는 스크립트에서 생성된 Cross-Site HTTP Requests 는 Same Origin Policy를 적용 받기 때문에 Cross-Site HTTP Requests가 불가능하다.
- 이를 해결하기 위해 W3C(World Wide Web Consortium)에서 CORS라는 이름의 권고안이 나오게 되었다.
- CORS : 추가 HTTP 헤더를 사용하여, 한 출처에서 실행 중인 웹 애플리케이션이 다른 출처의 선택한 자원에 접근할 수 있는 권한을 부여하도록 브라우저에 알려주는 체제.

* https://developer.mozilla.org/ko/docs/Web/HTTP/CORS

- Same Origin Policy : 어떤 출처에서 불러온 문서나 스크립트가 다른 출처에서 가져온 리소스와 상호작용하는 것을 제한하는 중요한 보안 방식. 잠재적으로 해로울 수 있는 문서를 분리함으로써 공격 경로를 줄이는데 도움을 준다.

* https://developer.mozilla.org/ko/docs/Web/Security/Same-origin_policy

HTTPS

- HTTP는 TCP와 직접 통신 (기본 포트 80)
- HTTPS에서 HTTP는 SSL과 통신하고, SSL이 TCP와 통신한다. (기본 포트 443)
- SSL을 사용한 HTTPS는 암호화와 안전성 보호를 이용할 수 있다.
- 왜 모든 웹 페이지에서 HTTPS를 사용하지 않을까?

HTTP : 평문 통신 : 암호화 되지 않은 전송 또는 저장된 데이터.

HTTPS : 암호화 통신 : CPU 혹은 메모리 같은 리소스가 많이 필요하다. 따라서 서버 한 대당 처리할 수 있는 Requests가 줄어들기 때문이다.

HTTPS 보안 작동 방식

0. 웹 브라우저가 https URL 주소를 만나면,

1. 웹 서버에 80번 포트가 아닌 443번 포트로 TCP 연결을 맺고

2. 바이너리 포맷으로 된 보안 매개변수를 교환(handshake)하고,

3. 그와 관련된 HTTPS 명령들이 작동됨.

*TLS SSL HandShake

https://kthan.tistory.com/entry/TLS-%ED%95%B8%EB%93%9C%EC%89%90%EC%9D%B4%ED%81%AC-Handshake-%ED%94%84%EB%A1%9C%ED%86%A0%EC%BD%9C-%EB%B6%84%EC%84%9D

WAS

- WAS(Web Application Server) : DB 조회나 다양한 로직 처리를 요구하는 동적인 컨텐츠를 제공하기 위해 만들어진 서버.
- HTTP를 통해 컴퓨터나 장치에 애플리케이션을 수행해주는 미들웨어(소프트웨어 엔진)이다.
- Web Container 혹은 Servlet Container 라고 불린다.
- Container : JSP, Servlet을 실행시킬 수 있는 소프트웨어.
- JSP : HTML 코드 안에 Java 코드,
  JSP가 수정된 경우 재배포할 필요 없이 WAS가 알아서 처리한다.

- Servlet : Java 코드 안에 HTML 코드 (하나의 클래스)
  Servlet이 수정된 경우 전체 코드를 업데이트하고, 다시 컴파일한 후 재배포 해야 한다.

*JSP Servlet

https://gmlwjd9405.github.io/2018/11/04/servlet-vs-jsp.html

WAS의 역할과 주요기능

- WAS = Web Container + Web Server

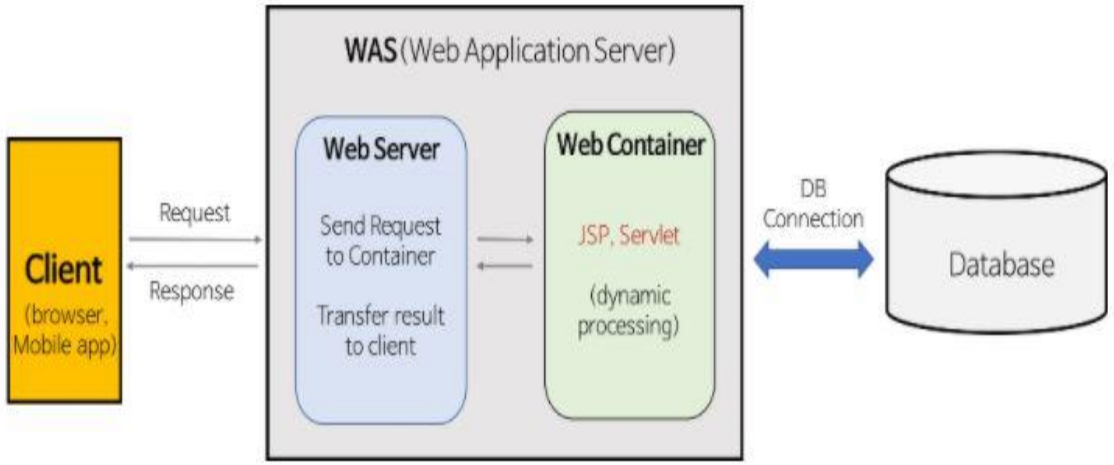
ex) http://tomcat.apache.org/ / Jeus / Web sphere 등

역할

- 분산 트랜잭션, 보안, 메시징, 쓰레드 처리 등의 기능을 처리하는 분산 환경에서 사용된다.
- 주로 DB 서버와 같이 수행된다.

주요기능

- 프로그램 실행 환경과 DB 접속 기능 제공
- 여러 개의 트랜잭션(논리적인 작업 단위) 관리 기능
- 업무를 처리하는 비즈니스 로직 수행



Web Server & WAS

두 서버를 통합하지 않는 이유

1) 기능을 분리하여 서버 부하 방지
Web Server : 정적 컨텐츠만 처리
WAS : 동적 컨텐츠만 처리
복합적으로 사용할 경우 데이터 처리로 인한 부하가 커지고, 수행속도가 느려진다.

2) 보안 강화
SSL에 대한 암복호화 처리에 Web Server를 사용한다.

3) Load Balancing 강화
Fail over, Fail back 처리에 유리
Fail over : 서버에 이상이 생겼을 때 예비 시스템으로 자동 전환되는 기능
Fail back : 서버에 장애가 발생하기 전의 상태로 되돌리는 처리

*WAS
https://gmlwjd9405.github.io/2018/10/27/webserver-vs-was.html

Data Base

Data Base 특징

1) 독립성 (하위 단계의 데이터 구조가 변경되더라도 상위 단계에 영향을 미치지 않음)

2) 무결성 (잘못된 데이터 발생 하는 경우의 수 방지 =>유효성 검사)

3) 보안성 (인가된 사용자들만 자원에 접근)

4) 일관성 (연관된 정보 논리적 구조로 관리)

5) 중복 최소화 (자료 중복, 데이터 중복 문제 해결)

- 무결성 면접 질문
- https://kadamon.tistory.com/21

관계형 DB

행과 열을 가진 DB에서 한 곳의 데이터와 다른 곳의 데이터를 연결하기 위해
한 쪽의 데이터에 다른 쪽 데이터의 위치를 저장하는 방식

- 장점

1) 범용성 / 고성능
2) 데이터의 일관성
3) SQL(Structured Query Language) 지원 : 데이터 관리 및 접근

- 단점

1) 대량의 입력 처리
2) 컬럼 확장의 어려움

비관계형 DB

데이터 간의 관계를 만들 수 있지만, 제한이 없다.
데이터의 저장 및 검색을 위한 특화된 메커니즘을 제공한다.

- 장점

1) 대용량 데이터
2) Cloud Computing
3) 빠른 읽기/쓰기 속도
4) 유지 보수 및 확장에 용이

- 단점

1) 데이터가 여러 컬렉션에 중복되어 있기 때문에,
   데이터 수정 시 모든 컬렉션에서 수행해야한다.

*관계형 비관계형 DB

https://siyoon210.tistory.com/130

DB & Storage

- DB : 2차원 데이터 형태인 칼럼(Column)과 로우(Row)로 구성되는 테이블형 데이터가 담긴다.
- Storage : 파일 형태가 되면 무엇이든 담을 수 있다.

즉, Storage는 파일 형태가 되면 무엇이든 담을 수 있지만,
DB는 애플리케이션 서버를 통해 가공해서 담아야 한다.

DB Server

다른 컴퓨터 프로그램이나 컴퓨터에 데이터베이스 서비스를 제공하는 데이터베이스 응용프로그램을 사용하는 서버이다.

단일 서버

WAS DB Server 분리

WAS와 DB Server의 분리

DB Server를 분리하는 가장 큰 이유는 WAS에서 처리할 부분이 DB Server 에 비해 상대적으로 많기 때문이다.

단일 서버로 구현된 경우 WAS의 성능을 높이려면, DB Server 까지 함께 높여야 한다. => 가격 / 성능의 비효율성

WAS와 달리 DB는 하드웨어보다 메모리에 영향을 많이 받으므로, 성능을 높이기 위한 기술적 방법이 다르다.

분리된 서버의 장점

1) 확장성
병렬 구조의 Scale out 방식의 서버 증설을 하기 위함. 자원을 WAS와 공유하지 않음.

2) 보안성
WAS 가 해킹 당해도 DB Server는 별도의 제한적 권한이 있다.

3) 기타
WAS 와 DB Server 사이에 Load Balancer를 두고 여러 개의 Web Server를 분산하여 사용.
WAS와 DB의 Scale out을 각각 다르게 설정, 사용할 수 있다.

Scale out : 접속된 서버의 수를 늘려 처리 능력을 향상 시킴.
*scale out / scale up 비교
https://m.blog.naver.com/islove8587/220548900044