

# 웹 서버 프로그래밍 시 주의할 점

📅 Date @2021/11/04

옛날(싱글) 게임

온라인 게임

클라이언트?

Http

'클라이언트'를 의심하라

요청 검증 - 파일 업로드

클라이언트 요청

SQL Injection

DoS 공격

XSS

'서버'단도 의심

부하 테스트

응답 시간

데이터 신뢰성

DB 신뢰

트랜잭션

logging

서버 테스트 시나리오

서버 프로그래밍 요약

## 옛날(싱글) 게임

개발자는 "language"를 이용해서 궁극적으로 메모리에 저장하는 값을 제어해가며 원하는 기능을 만들어 낸다.

치팅 컨트롤 프로그램이 무수히 등장해 각종 데이터를 변환할 수 있었다.

## 온라인 게임

클라이언트의 데이터와 서버가 알고 있는 데이터 무엇이 우선 시 될까? 이유는?

치팅과 같은 불법적인 접근을 제한하기 위해서 서버 데이터가 우선 시 된다.

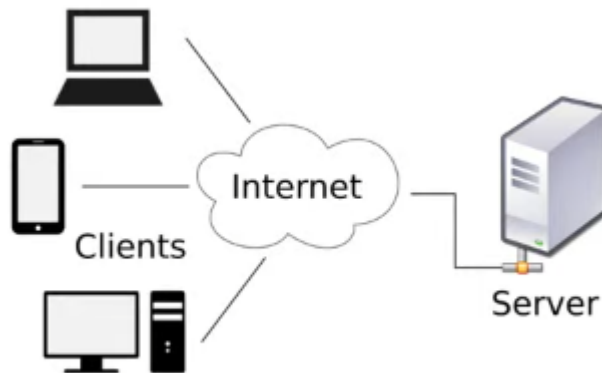
## 클라이언트?

서버의 능동적인 대처가 필요하다.

개발자가 어떤 의도로 만들었던, 클라이언트가 그렇게 써줄 가능성은 없다.

따라서 대처해야한다.

# Http



- Hyper Text Transfer Protocol
- Stateless
- Http 서버는 불특정 다수의 요청을 전제로 한다.



## Stateless 프로토콜의 단점은 무엇이며 이를 해결하기 위한 방법은?

클라이언트 구분이 불가능하다. 따라서 세션이나 쿠키 등의 다양한 해결책을 이용하게 된다.

## '클라이언트'를 의심하라

- 사용자 인증
- 서버 중심의 견고한 로직 처리
- 사용자 요청의 유효성 검증
- Sql Injection, XSS 방어

## 요청 검증 - 파일 업로드

: 실제 사례 "DIGITAL Hyundai Card"

- 게시판 파일 업로드 기능
- '.jsp' 파일 업로드 허용 : 데이터 필터링이 필요한 이유

- '.jsp' 파일 내에 특정 테이블 조회 기능을 구현해 정보를 빼돌렸다.... (ㄷㄷ)

## 클라이언트 요청

: 앞서 순서를 진행한 후에 해야하는 기능에 접근을 제한을 줘야한다.

→ 앞 순서를 진행해 Token 을 보내 서버에서 확인해주는 제한이 가능하다.

## SQL Injection

- 클라이언트의 입력 값은 DB query > where 조건문의 일부로 사용

```
select * from USERS where id = 'user1' and pw = 'pw1';
select * from USERS Where id = '' or 1 = 1 -- ' and pw = 'pw1';
```

## DoS 공격

- 최소한의 장치를 마련할 수 있다.

<https://velog.io/@damiano1027/Nginx-DoS-DDoS-%EA%B3%B5%EA%B2%A9-%EB%B0%A9%EC%96%B4-%EC%84%A4%EC%A0%95>

## XSS

- Cross-Site Scripting

<script> 악의적인 해커의 url로 사용자 주요 정보 전송 </script> 등의 다양한 내용들을 "정규표현식" 등으로 제한해줄 수 있다.

## '서버'단도 의심

- 1이 아닌 N개의 클라이언트의 요청이 올 것이다.
- 적절한 응답 시간과 데이터의 신뢰성을 갖고 있어야한다.

- 트랜잭션 처리 여부를 고민하자.
- 서비스 및 콘텐츠 사용 권한을 신경쓰자
- 주요 데이터를 암호화해 탈취를 조심하자

## 부하 테스트

- 1이 아닌 N개 클라이언트의 요청
- 적절한 응답 시간과 데이터 신뢰성
- 'JMeter'와 같은 도구를 활용하여 상품 검색 부하 테스트를 진행하자

## 응답 시간


- 요청-응답 시간을 줄이기 위해 캐시 layer 적용을 고려하자.
- Redis 등

## 데이터 신뢰성

- Multi-thread 환경에서의 concurrency

### ArrayList (Java Platform SE 8 )

The iterators returned by this class's `andListIterator` methods are fail-fast: if the list is structurally modified at any time after the iterator is created, in any way except through the iterator's own `next` or `previous` methods, the iterator will throw a `ConcurrentModificationException`.

 <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>

## DB 신뢰

- SELECT ... FOR UPDATE 문 등으로 명시적 lock 부여
- 특정 row에 대해 동시 트랜잭션이 발생하여 데이터 정합성이 깨지는 것을 방지하자.

## 트랜잭션

- 기능을 하나 진행하더라도 방법이 여러가지가 존재하는 경우가 있다.
- 예 ) 결제 = 문화상품권 + 신용카드 + 쇼핑몰 포인트
- 하나가 실패하더라도 전체 트랜잭션을 실패하도록 롤백시켜줄 수 있는 단계를 꼭 넣어야 한다.



개발자가 하는 일이란게 뭐 있나요? 돌려보고 생각대로 안 움직이면 디버깅 걸고 한 줄 한 줄 확인하고 사용하는라이브러리(제품)가 있다면 로그 잘 보고 하나씩 하나씩 해결하는 것 말고 없습니다.

## logging

- 서버가 작동하면서 발생하는 주요 이슈들을 기록
- 단순 콘솔 출력보다는 주요 logger(library)를 사용
- 서버 운영 시 발생한 이슈 추적의 시작점

## 서버 테스트 시나리오

- 복수개의 요청은 수행해봐야 함
- 눈에 보이지 않는 비기능요소에 대한 테스트 시나리오 작성
- 테스트할 내용을 안다는 것 자체로 SW역량 인증

## 서버 프로그래밍 요약



### ‘너’를 의심

- 서버 중심의 로직 처리
- 코드값등의 유효성 검증
- 사용자 인증
- Sql injection, XSS 방어



### ‘나’를 의심

- 적절한 응답 시간
- 데이터 신뢰성
- 트랜잭션 처리 여부
- 리소스 사용 권한
- 주요 데이터 암호화



### 발생한 일 기록

- 효율적인 logging 구축
- 주요 액티비티 로깅
- 익셉션, 에러 로깅

- 인증/인가 모두 고민하자