

프로젝트 명세서

Client-Side Rendering vs Server-Side Rendering

목차

1. 개요	3
2. SPA, MPA 접속	3
3. CSR vs SSR	5
4. 패킷 구조	5
5. MPA	6
6. SPA	7
7. 과제	8
8. 심화 과제	9
9. 산출물 제출	10

1. 개요

Vue 와 React 는 대표적인 Single page application (이하 SPA) 프레임워크로 Client Side Rendering(이하 CSR) 방식으로 뷰(View)를 만든다. 이와 대조적으로 PHP 는 Multi page application(이하 MPA) 으로 Server Side Rendering(이하 SSR) 방식으로 뷰를 만든다. 본 과제의 목적은 그것의 정확한 의미를 이해하는데 있다.

2. SPA, MPA 접속

SPA 와 MPA로 개발된 두 개의 웹 서비스가 있다. 그리고 그 각각은 a, b 두 개 페이지를 제공한다. 아래 주소에 각각 접속해보자. (접속이 안될 경우 방화벽에 막혀 있을 수 있다. 접속하는 Client 의 네트워크 환경을 점검하라)

MPA	http://13.124.193.201:8500/a
	http://13.124.193.201:8500/b
SPA	http://13.124.193.201:8200/a
	http://13.124.193.201:8200/b



〈접속 시 출력된 화면〉

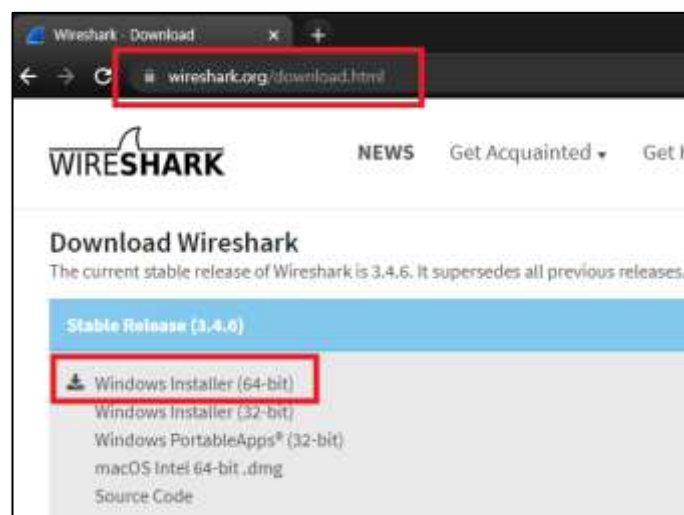
각 웹 서비스에서 페이지 A와 B를 전환해 보면, 브라우저에 보이는 화면의 차이는 없어 보인다. 하지만 내부 동작은 다르다.

3. CSR vs SSR

1. CSR: 브라우저에서 js(java script)에 의해 뷰(HTML 코드)을 동적으로 생성한다. 그래서 페이지 전환이 SSR 보다 상대적으로 빠르다. 대신 최초 접속 시, 모든 페이지의 js와 static 파일(HTML 코드, image 등)를 가져와야 한다. 그래서 최초 접속 시 로딩은 SSR에 비해 늦다.
2. SSR: 웹 서버에서 뷰를 생성한다. 페이지가 전환될 때 마다, 클라이언트가 서버에 뷰를 요청하고, 서버는 그것을 생성 후 클라이언트에게 보낸다. 그래서 뷰 전환 속도가 CSR에 비해 상대적으로 늦다. 그리고 페이지 전환 요청이 잦아지면 CSR에 비해 서버에 더 많은 부하를 준다.

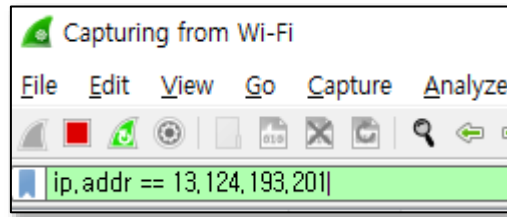
4. 패킷 구조

SPA, MPA에서 페이지 전환 시, 주고받는 패킷의 구조 다르다. 그 구조를 확인하여 SPA와 MPA는 어떻게 다른지 알 수 있다. 다음과 같이 네트워크 패킷을 캡처하기 위해 Wire Shark를 설치하자.



<Wire Shark Download>

아래와 같이 필터를 설정하고 패킷을 캡처하자.



<IP Filter 설정>

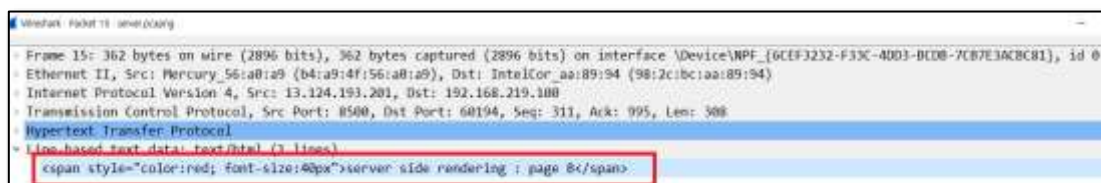
5. MPA

브라우저로 MPA의 웹 서비스에 접속해보자. 그리고 페이지 a와 b를 번갈아 전환해보자. 아래와 같이 Wire Shark로 페이지를 전환할 때 마다 클라이언트와 서버 간에 주고받는 패킷이 생성되는지 확인할 수 있다.

1	0.0.	192.168.219.100	13.124.193.201	TCP	54 61095 → 8200 [FIN, ACK] Seq=1 Ack=1 Win=3768 Len=0
2	0.0.	192.168.219.100	13.124.193.201	TCP	54 57547 → 8200 [FIN, ACK] Seq=1 Ack=1 Win=312 Len=0
3	0.0.	192.168.219.100	13.124.193.201	TCP	54 54612 → 8200 [FIN, ACK] Seq=1 Ack=1 Win=6107 Len=0
4	0.0.	192.168.219.100	13.124.193.201	TCP	60 60194 → 8500 [SYN] Seq=0 Win=64240 Len=0 MSS=3840
5	0.0.	13.124.193.201	192.168.219.100	TCP	54 8200 → 54612 [RST] Seq=1 Win=0 Len=0
6	0.0.	13.124.193.201	192.168.219.100	TCP	54 8500 → 60194 [SYN, ACK] Seq=0 Ack=1 Win=26868 Len=0
7	0.0.	13.124.193.201	192.168.219.100	TCP	54 8200 → 53547 [RST] Seq=1 Win=0 Len=0
8	0.0.	13.124.193.201	192.168.219.100	TCP	54 8200 → 61885 [RST] Seq=1 Win=0 Len=0
9	0.0.	192.168.219.100	13.124.193.201	TCP	54 60194 → 8500 [ACK] Seq=1 Ack=1 Win=11112 Len=0
10	0.0.	192.168.219.100	13.124.193.201	HTTP	551 GET /a HTTP/1.1
11	0.0.	13.124.193.201	192.168.219.100	TCP	54 8500 → 60194 [ACK] Seq=1 Ack=898 Win=28812 Len=0
12	0.0.	13.124.193.201	192.168.219.100	HTTP	362 HTTP/1.1 200 OK (text/html)
13	0.0.	192.168.219.100	13.124.193.201	TCP	54 60194 → 8500 [ACK] Seq=1 Ack=1 Win=11112 Len=0
14	1.7.	192.168.219.100	13.124.193.201	HTTP	551 GET /b HTTP/1.1
15	1.7.	13.124.193.201	192.168.219.100	HTTP	362 HTTP/1.1 200 OK (text/html)
16	1.7.	192.168.219.100	13.124.193.201	TCP	54 60194 → 8500 [ACK] Seq=595 Ack=619 Win=130560 Len=0

<Packet capture>

다음으로 서버에서 온 응답을 더블 클릭하여 자세히 보자. 다음과 같이 응답에 뷰(HTML 코드)가 포함된 것이 보이는가?



<Packet capture>

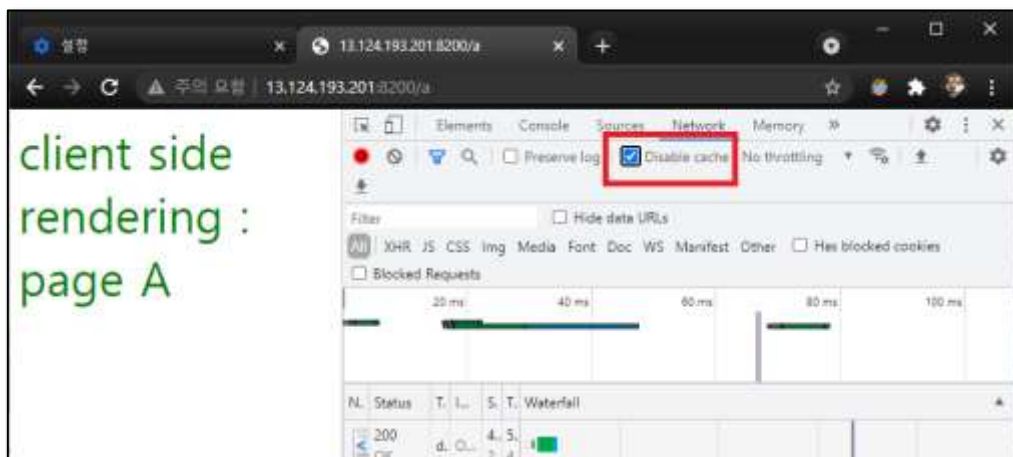
6. SPA

브라우저로 SPA의 웹 서비스에 접속해보자. 아래와 같이 Wire Shark에서, 최초 접속 시, js 및 static 파일을 다운 받는지 확인하라.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0	192.168.219.100	13.124.193.201	HTTP	551	GET /a HTTP/1.1
2	0.0	13.124.193.201	192.168.219.100	HTTP	497	HTTP/1.1 200 OK (text/html)
3	0.0	192.168.219.100	13.124.193.201	HTTP	445	GET /js/app-8820ce48.js HTTP/1.1
4	0.0	192.168.219.100	13.124.193.201	HTTP	455	GET /js/chunk-vendors.42056474.js HTTP/1.1
5	0.0	13.124.193.201	192.168.219.100	TCP	1514	8200 → 54632 [ACK] Seq=664 Ack=629 Win=328 Len=0 [TCP segment of data stream 0]
6	0.0	13.124.193.201	192.168.219.100	HTTP	1226	HTTP/1.1 200 OK (application/javascript)
7	0.0	192.168.219.100	13.124.193.201	TCP	54	54632 → 8200 [ACK] Seq=664 Ack=1076 Win=0 Len=0
8	0.0	13.124.193.201	192.168.219.100	TCP	1514	8200 → 61885 [ACK] Seq=1 Ack=402 Win=259 Len=0 [TCP segment of data stream 0]
9	0.0	13.124.193.201	192.168.219.100	TCP	1514	8200 → 61885 [ACK] Seq=1461 Ack=402 Win=260 Len=0 [TCP segment of data stream 0]
10	0.0	13.124.193.201	192.168.219.100	TCP	1514	8200 → 61885 [ACK] Seq=2921 Ack=402 Win=260 Len=0 [TCP segment of data stream 0]
11	0.0	13.124.193.201	192.168.219.100	TCP	1514	8200 → 61885 [ACK] Seq=4381 Ack=402 Win=260 Len=0 [TCP segment of data stream 0]
12	0.0	13.124.193.201	192.168.219.100	TCP	1514	8200 → 61885 [ACK] Seq=5841 Ack=402 Win=260 Len=0 [TCP segment of data stream 0]

<Packet capture>

보통 브라우저에는 다운로드 캐시(Download cache) 기능이 있기 때문에, 다운로드 되지 않을 수 있다. 그래서 아래와 같이 설정 후 접속하자. (크롬에서 F12를 누르면 설정 화면이 나온다)



<Disable Cache>

js & static 이 들어간 패킷을 찾아 더블 클릭하여 자세히 살펴보자. 아래와 같이 js 및 static 파일의 내용을 확인하자.

```

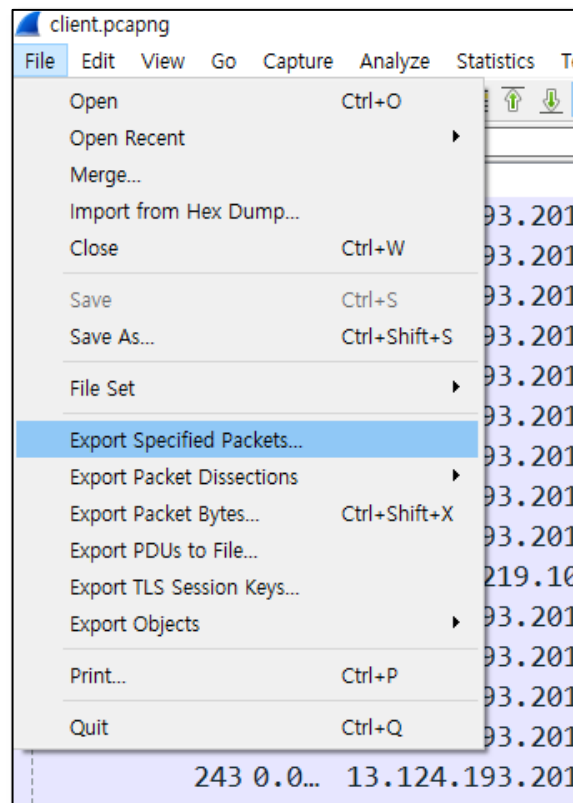
[Next Sequence Number: 2921 (relative sequence number)]
Acknowledgment Number: 402 (relative ack number)
Acknowledgment number (raw): 3169765577
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 269
[Calculated window size: 269]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x7bfa (unverified)
[Checksum Status: Unverified]
Urgent Pointer: 0
[SEQ/ACK analysis]
[Timestamps]
TCP payload (1460 bytes)
[Reassembled PDV in frame: 132]
TCP segment data (1460 bytes)
0010 01 0d 7b fa 00 00 65 29 7b 74 2e 65 78 70 6f 72 ..{...e) {t.expor
0040 74 73 3d 66 75 6e 63 74 69 6f 6e 28 74 29 7b 69 ts=func ion(t){i
0050 66 28 22 66 75 6e 63 74 69 6f 6e 22 21 3d 74 79 f("func ion"-ty
0060 70 65 6f 66 20 74 29 74 68 72 6f 77 20 54 79 70 peof t)t hrow Typ
0070 65 45 72 72 6f 72 28 53 74 72 69 6e 67 28 74 29 eError($ tring(t)
0080 2b 22 20 69 73 20 6e 6f 74 20 61 20 66 75 6e 63 *" is no t a func
0090 74 69 6f 6e 22 29 3b 72 65 74 75 72 6e 20 74 7d tion");r eturn t}
00a0 7d 2c 22 31 63 37 65 22 3a 66 75 6e 63 74 69 6f }, "lc7e" :functio
00b0 6e 28 74 2c 65 2c 6e 29 7b 76 61 72 20 72 3d 6e n(t,e,n) {var r:n
00c0 28 22 62 36 32 32 22 29 2c 6f 3d 72 28 22 69 74 ("b622") ,o=r("it
00d0 65 72 61 74 6f 72 22 29 2c 69 3d 21 31 3b 74 72 erator") ,i=1;tr
00e0 79 7b 76 61 72 20 61 3d 30 2c 73 3d 7b 6e 65 78 y{var a= 0,s={nex
00f0 74 3a 66 75 6e 63 74 69 6f 6e 28 29 7b 72 65 74 t:functi on(){ret
0100 75 72 6e 7b 64 6f 6e 65 3a 21 21 61 2b 2b 7d 7d urn(done :!!a++)}
0110 2c 72 65 74 75 72 6e 3a 65 75 6e 63 74 69 6f 6e ,return: function
  
```

〈Js & static 파일〉

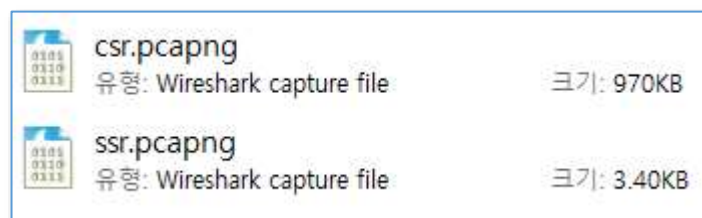
다음으로 페이지 A와 B를 번갈아 전환해도 서버에 요청을 보내지 않는지 확인하라.
페이지를 전환할 때, MPA와는 다르게 Wire Shark에서 패킷이 캡처 되지 않을 것이다.

7. 과제

MPA와 SPA 접속과 화면 전환하면서 발생하는 패킷을 각각 export 하여 제출하라.



〈Export Packet〉



〈Exported Files〉

8. 심화 과제

HTTP 패킷을 자세히 분석해보자. 아래 화면 같이 HTTP의 헤더에서 각 item들이 의미하는 바는 무엇인가?

```

- Hypertext Transfer Protocol
GET /a HTTP/1.1\r\n
Host: 13.124.193.201:8200\r\n
Connection: keep-alive\r\n
Pragma: no-cache\r\n
Cache-Control: no-cache\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.77 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: ko-KR;q=0.9,en-US;q=0.8,en;q=0.7\r\n
\r\n
[Full request URI: http://13.124.193.201:8200/a]
[HTTP request 1/3]
[Response in frame: 2]
[Next request in frame: 1]

```

<HTTP Head>

9. 산출물 제출

https://lab.ssafy.com/s12-study/seasonal_fesw 의 “산출물 제출 가이드” 참조