실습10
Client-side Dynamic Web Service

2022년 가을학기 컴퓨터네트워크 02분반
데이터네트워크연구실
문현수, 이영석
2022. 11. 18.
munhyunsu@cs-cnu.org

실습 시작 전...

실습8 Web Service Deployment 피드백

- 제출률: 84.62% (22/26)
  - e-mail 질문: 17건
  - 디스코드 질문: 2건
  - 연구실 방문: 1건

- 설문조사 주요 내용

fastapi를 nginx docker 서버로 연결할 때 root_path를 설정하면 docs에 접근할 수 있었습니다. 하지만 uvicorn의 포트로 연결할 때는 안되는데 (8080/docs는 되는데 8889/docs는 안된다) 둘 다 되게하는 법이 있을까요?

로컬호스트 환경이지만 직접 Nginx 설정을 간략하게나마 해보는 경험이 좋았습니다.

구조를 정말 쉽게 알 수 있어서 많이 배울 수 있었습니다.

정말 쉬운거였는데 리눅스랑 환경이 다른걸 몰라서 오래걸렸네요. 개발자도구 열어볼 생각을 조금 더 일찍했으면 더 빨리 풀었을 것 같습니다.

이번 과제는 계속 시도하다가 결국 구현하지 못하였는데 답안이 궁금합니다.

실습9 Server-side Dynamic Web Service 피드백

- 제출률: 53.85% (14/26)
  - e-mail 질문: 30건
  - 디스코드 질문: 3건
  - 연구실 방문: 0건

- 설문조사 주요 내용

구현양이 상당히 많았던 것 같아요ㅠㅠ
구현 페이지가 많은 것에 비해 나와있는 코드는 적어서 시간이 많이 걸렸습니다 :(

과제들이 학생들을 스스로 생각하게 만듭니다.

지난 주차 코드를 어디서부터 어디까지 써야하는지 모르겠다.
urllib, flask 를 처음 봤는데 어떤식으로 써야하는지 모르겠다.
실행을 어떻게 해야할지 모르겠다.
Flask 사용법에 대한 설명이 부족해서 어디서부터 어디까지 해야할지 잘 모르겠다.
Flask 에서 FastAPI 서버 간의 통신을 어떻게 해야할지 잘 모르겠다.

엔진엑스의 이런 리버스프록시 사용기회를 제공해주셔서 정말 감사합니다. 혹시 cors관련해서도 사용기회를 얻을 수 있을까요

웹과 연동하는 방식의 과제라 재미있었다.

html, js코드도 짜야되서 평소보다 어려웠던것 같습니다

좀 어렵긴한데 재밌어요

이 수업덕분에 어느정도 완성도 있는 앱정도는 만들 수 있을 거 같습니다. 그런데 제 컴퓨터안에서만 통신하기때문에 배포를 한다면 막막한데요 혹시 aws라던가 실제 배포를 할 때 서버를 올리는 방법에 대해서도 이 수업에서 경험할 수 있을까요. 감사합니다.

node.js를 배우고 싶습니다.. 개인적으로 이영석 교수님의 분반을 선택한 이유는 강의 계획서에서 node.js를 주로 다룬다는 점이 마음에 들어서 선택을 했습니다. 그러나 아직도 node.js를 시작하지 못해서 너무 아쉽습니다.. 언제쯤부터 nodejs로 실습을 하는지 알 수 있을까요?

post는 Request로 하는 것이 아닌가요?

지난주까지 우리가 배운 것과 이번주에 배울 것

어제의 나

1. Wireshark 라는 프로그램의 존재 인지
2. Docker 활용 Web server (nginx) 띄우기
3. HTTP server 는 어떤 역할을 하는가?
4. HTTP server load test
5. MPM HTTP server
6. RESTful API server
7. App server and database integration
8. Web and App service deployment
9. Server-side dynamic web service

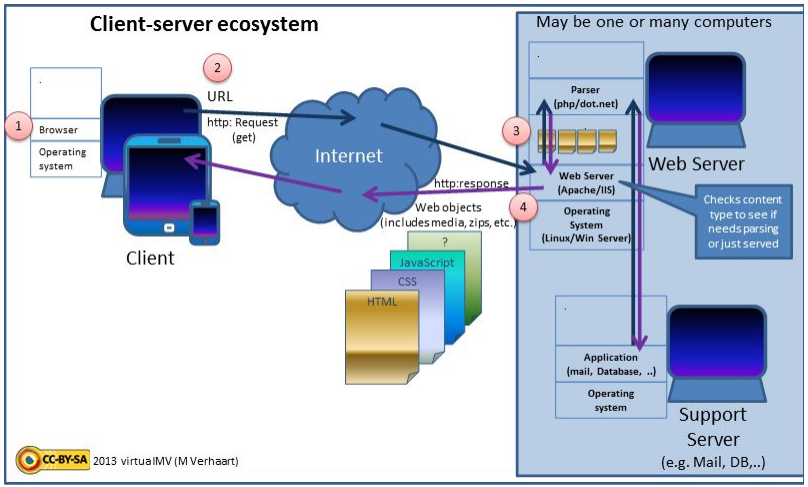
오늘의 나

- Client-side Dynamic Web Service

(복습)
Dynamic web page

Dynamic web page [링크]

- 서버측 동적 웹
  - 서버에서 사용자에 따라 전달하는 웹 문서 생성
- 클라이언트측 동적 웹
  - 클라이언트에서 사용자 입력에 따라 웹 문서 변경 (추가 요청 포함)

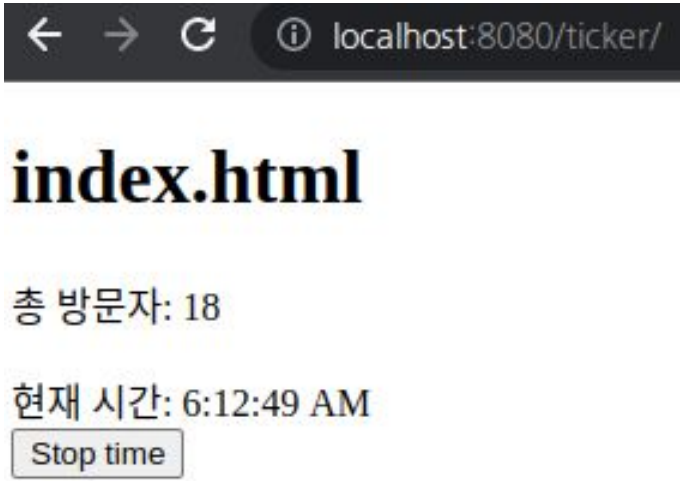


실습)
Simple Dynamic Web using JavaScript

현재 시간도 표시하는 웹 페이지 (지난 실습 포함)

동적 웹을 만들기 위한 요구 사항 (확장성)

- 데이터베이스 연결
  - 데이터베이스에클라이언트로그 작성
- 클라이언트 로그 수를 웹페이지에 표시
  - 매번 접속할 때마다 Counting
- HTML Template 구조 및 원리 이해!
  - App 서버에서 어떻게 동적 웹 페이지를 생성하는가?
- 클라이언트 레벨에서 어떻게 동적으로 웹 페이지를 렌더링하는가?
  - javascript 관찰 [참고]



- 샘플 코드는 dir 디렉터리 데이터를 서빙하므로 주의

```
dir/
├── index.html
└── myscript.js

0 directories, 2 files
```

동적 웹 HTML (Template)

HTML Template

- {{ COUNTER }} 와 같은 문법 체크!
- App 서버에서는 해당 문법을 어떻게 처리할까?

```
<!DOCTYPE html>
<html>
<body>

<h1>index.html</h1>

<p>총 방문자: {{ COUNTER }}</p>

<div id="timer">
</div>

</body>
<script src="/ticker/myscript.js"></script>
</html>
```

```
<!DOCTYPE html>
<html>
<body>

<h1>index.html</h1>

<p>총 방문자: 18</p>

<div id="timer">
</div>

</body>
<script src="/ticker/myscript.js"></script>
</html>
```

참고) myscript.js

- 시간 템플릿

```
const myInterval = setInterval(myTimer, 1000);

function myTimer() {
  const date = new Date();
  document.getElementById('timer').innerHTML = '현재 시간: ' +
date.toLocaleTimeString() + '<br><button
onclick="myTimerStop()">Stop time</button><br>';
}

function myTimerStop() {
  clearInterval(myInterval);
}
```

Python 3 HTTP Server (HTTP Handler)

```
import os
import http
import http.server
import socketserver
import sqlite3

PREFIX = ''

FLAGS = _ = None
DEBUG = False

EXT = {'.html': 'text/html;charset=utf-8',
       '.js': 'text/javascript;charset=utf-8'}

CONN = sqlite3.connect('./Log.db')
CUR = CONN.cursor()
CUR.execute('''CREATE TABLE IF NOT EXISTS Log (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                ip TEXT,
                request TEXT);''')
CONN.commit()


class MyHTTPDaemon(http.server.HTTPServer):
    allow_reuse_address = True


class MyHTTPRequestHandler(http.server.BaseHTTPRequestHandler):
    def __init__(self, *args, **kwargs):
        self.rootdir = FLAGS.rootdir
        self.conn = CONN
        self.cur = CUR
        super().__init__(*args, **kwargs)

    def do_GET(self):
        if DEBUG:
            print(f'Command: {self.command}')
            print(f'Path: {self.path}')
            print(f'Headers: {self.headers}')

        self.cur.execute('''INSERT INTO Log (ip, request)
                            VALUES (?, ?);''', (self.client_address[0], self.requestline))
        self.conn.commit()

        if not self.path.startswith(PREFIX):
            return None
        self.path = self.path[len(PREFIX):]
        if self.path == '/':
            path = 'index.html'
        else:
            path = self.path[1:]
        path = os.path.join(self.rootdir, path)
        if DEBUG:
            print(f'Joined path: {self.rootdir} {self.path} {path}')
        if not os.path.exists(path):
            self.send_error(http.HTTPStatus.NOT_FOUND, 'File not found')
        else:
            ext = os.path.splitext(path)[-1].lower()
            self.send_response(http.HTTPStatus.OK)
            self.send_header('Content-Type', EXT[ext])
            with open(path, 'rb') as f:
                self.cur.execute('''SELECT MAX(id) FROM Log;''')
                counter = self.cur.fetchone()[0]
                body = f.read()
                body = body.replace('{{ COUNTER }}'.encode('utf-8'), f'{counter}'.encode('utf-8'))
                self.send_header('Content-Length', len(body))
                self.end_headers()
                self.wfile.write(body)
```

```
def main():
    print(f'Parsed arguments: {FLAGS}')
    print(f'Unparsed arguments: {_}')

    with MyHTTPDaemon((FLAGS.host, FLAGS.port),
                      MyHTTPRequestHandler) as httpd:
        try:
            print(f'Start server {httpd.server_address}')
            httpd.serve_forever()
        except KeyboardInterrupt:
            print(f'Stop server {httpd.server_address}')
            httpd.shutdown()


if __name__ == '__main__':
    import argparse

    parser = argparse.ArgumentParser()
    parser.add_argument('--debug', action='store_true',
                        help='Debug message')
    parser.add_argument('--host', default='0.0.0.0', type=str,
                        help='IP address')
    parser.add_argument('--port', default=8888, type=int,
                        help='Port number')
    parser.add_argument('--rootdir', default='./dir', type=str,
                        help='Root directory')

    FLAGS, _ = parser.parse_known_args()
    DEBUG = FLAGS.debug

    main()
```

실습)
Client-side Dynamic Web Server

JavaScript

- 웹 그룹이 HyperText Markup Language (HTML), Cascading Style Sheets (CSS) 이후 세번째로 선택한 기술 [링크]
  - 네번째는 WebAssembly
- HTML <script> 태그내에 위치
  - Client-side script

- HTML, DOM 등을 수정할 수 있음
- 서버에 요청을 보낼 수 있음
- 연산도 할 수 있음

- 지속적으로 개발 및 개선되고 있는 중 [링크]

Retrieve Data from Server using JavaScript

- 새로고침을 하지 않아도 자동으로 새로운 글을 보여줌

- 지난주까지의 과제 결과 활용!

힌트) newsfeed.js

- setInterval() 로 10초마다 새로운 데이터 요청
- 수신 성공시 json 으로 변환한 후 div 생성
- 새로고침에 비하여 장점은?

```
async function fetchRequestWithError() {
  try {
    const url = `http://localhost:8080/talkie/api/talks/`;
    const response = await fetch(url);
    if (response.status >= 200 && response.status < 400) {
      const data = await response.json();
      for (var key in data) {
        ndiv = document.createElement('div');
        ndiv.innerHTML = `<h3> ${data[key]['title']} </h3><p>
${data[key]['content']}</p><hr>`;
        pdiv = document.getElementById('notes');
        pdiv.appendChild(ndiv);
      }
    } else {
      console.log(`${response.statusText}: ${response.status} error`);
    }
  } catch (error) {
    console.log(error);
  }
}

fetchint = setInterval(fetchRequestWithError, 10*1000);
```

과제)
Client-side Dynamic Web Service

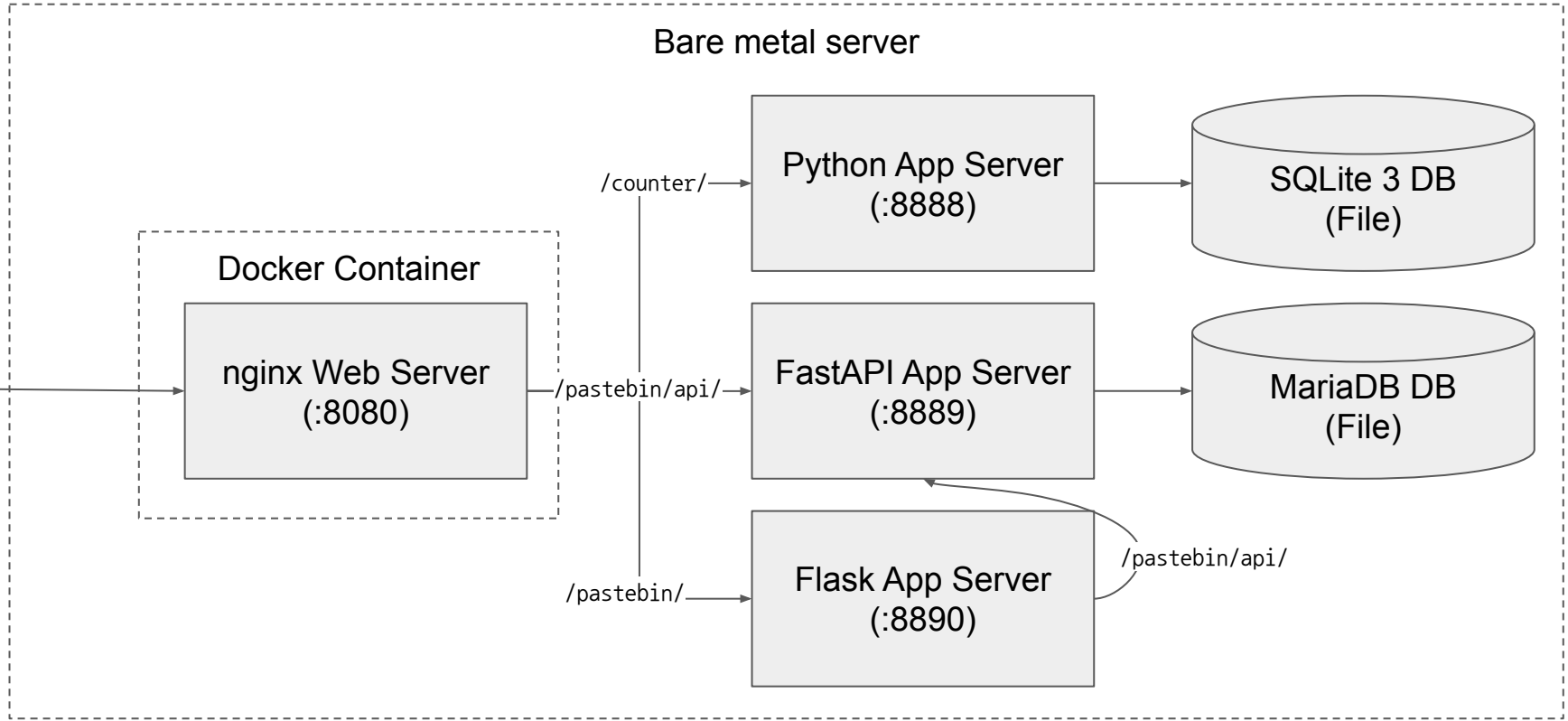
PasteBin/Talks News Feed Service

- 우리가 만들어 둔 PasteBin RESTful API 서비스의 News Feed 제공
  - PasteBin 을 Talks 와 같은 다른 이름으로 변경해도 됨!
  - RESTful API, Web service, Database, Web server 활용 ⇒ Micro service

Endpoints

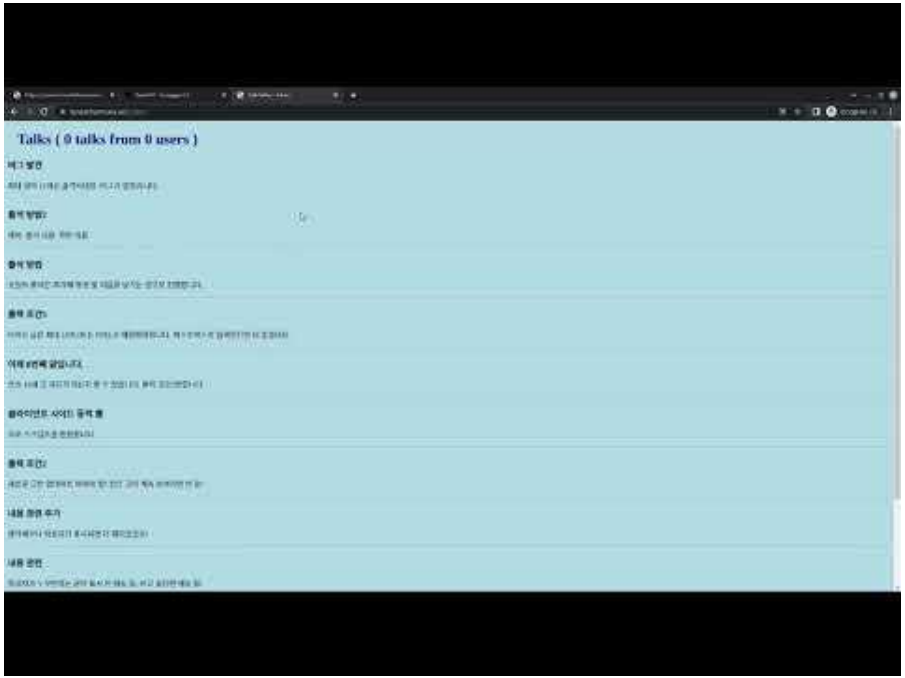
- 8080/counter ⇒ 일일 접속자 카운트 웹
- 8080/pastebin/api ⇒ FastAPI
- 8080/pastebin ⇒ Flask using RESTful API
  - index.html ⇒ news feed 제공

그림으로 보는 과제 구조


Bare metal server
Docker Container
nginx Web Server
(:8080)
/counter/
/pastebin/api/
/pastebin/
Python App Server
(:8888)
FastAPI App Server
(:8889)
Flask App Server
(:8890)
SQLite 3 DB
(File)
MariaDB DB
(File)
/pastebin/api/

과제 구현 조건

- 같은 글을 계속 적지 않아야 함!
  - 힌트) id, skip 활용!
- SNS 처럼 최근 글을 상단에 리프레시할 것! (내림차순)
- 표시할 글을 최대 10개 (변경 가능)만 표시할 것
  - removeChild() 활용



실습10 과제 정리 및 배점

보고서에 포함해야할 것:
→ 각 요소가 완성되어있음을 증명할 것

1. 동적 콘텐츠 업데이트 (1)
2. 새로운 글만 업데이트 (1)
3. 내림차순으로 정렬하여 업데이트 (1)
4. 최대 표시 개수 제한하여 업데이트 (1)
5. 모두 해결 (1)

- 학번-이름.zip 압축
  a. 보고서 (학번-이름.**pdf**)
  b. 소스코드
- e-learning 사이버캠퍼스 제출
- 설문조사: [클릭]

- 제출 기한: 2022. 11. 23. 23:59 (다음주 수요일 자정)
  ⇒ 목요일에 전 주 과제 채점 후 과제에 반영 위함

기타 참고 자료

질문 창구: On/Off-line (Offline 추천)

Online: Discord 활용 + e-mail + Whyrano

- 링크: https://discord.gg/xKFKsgQHqW
- 접속 후 학번-이름 으로 서버 닉네임 변경
  - 계정 닉네임을 변경하지 않아도 됨!
    서버 닉네임만 변경 가능!
- TA가 서버 닉네임, 출석부 대조 후 컴퓨터네트워크 채널 권한 부여
- 24/7 자유롭게 질문

- 이메일: munhyunsu@cs-cnu.org

- ~~Whyrano: 닉네임만 노출되는 질문 서비스~~
  - ~~우리 학과 학생이 서버스 런칭~~
  - ~~1일 1회 이상 모니터링 하겠음!~~

Offline: 연구실 방문

- 공대5호관 633호 데이터네트워크연구실
- Discord / e-mail 로 약속 후 방문
  - 급하면 그냥 와도 되나 바빠서 안 될 때도 있음

- 컴퓨터네트워크 이외의 질문도 가능