

---

# 웹 서버 구축

# 목차

---

## ■ 웹서버

- ◆ 웹서버 개요, 구성, 종류

## ■ 웹서버 설치

- ◆ Apache, lighttpd, bottle 웹서버 및 PHP(옵션) 설치

## ■ 파이썬 웹 프로그래밍

- ◆ HTML, CGI

- ◆ AJAX

- ◆ WSGI

- ◆ Bottle

## ■ 데이터베이스 설치

- ◆ MySQL

- ◆ SQLite

## ■ REST API 설계

- ◆ 스케줄 관리 프로그램

# 웹 서버

## ■ 웹 서버

- ◆ HTTP 프로토콜을 사용해서 클라이언트가 브라우저를 통해서 요청하는 데이터를 제공하는 인터넷 서비스 프로그램

## ■ 경량 웹 서버 기능

- ◆ 제한된 자원을 가진 시스템에 사용할 수 있도록 웹 서버의 기능을 제한하고 최적화할 필요가 있음

### ◆ 기능

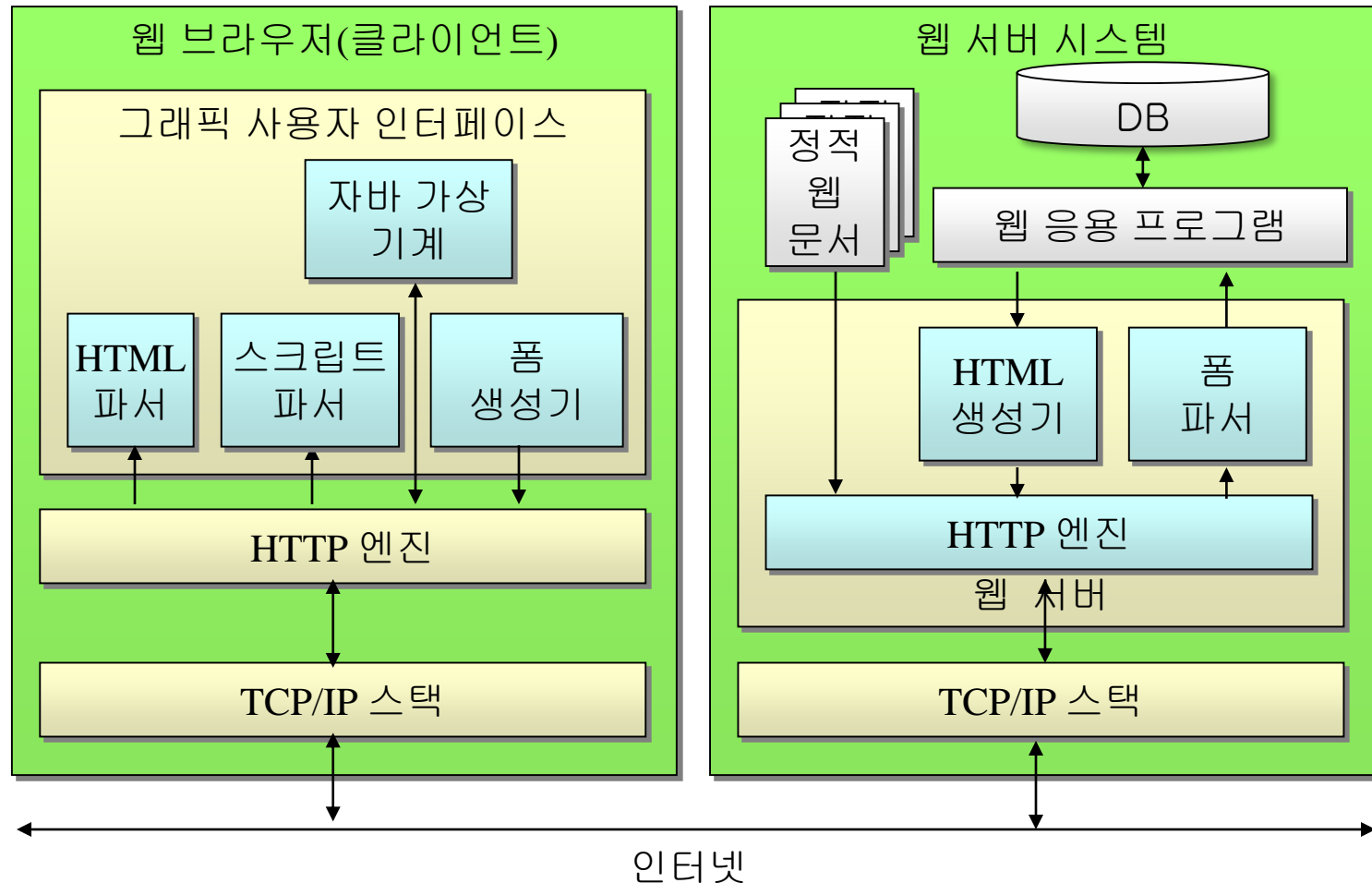
- 제한된 시스템 자원의 효율적인 사용 – 프로세스/메모리 처리 개선
- 라즈베리파이와 쉽게 통합 및 이식성이 높아야 함
- 웹 페이지의 동적 생성 지원 - CGI 및 서버측 스크립트 프로그램
- 설정 가능한 보안 모델 (제거 가능) – SSL, DAA
- 기타 로그 파일, 가상 서버, 인증 서버 등의 범용 웹 서버 기능 제한

### ◆ 응용

- 원격 시스템 감시 - 라즈베리파이의 상태 정보나 자료를 확인
- 원격 제어 - 동적으로 라즈베리파이의 설정 변경

# 웹 서버 개요

## ■ 웹 서버 및 클라이언트의 인터페이스



# 웹 서버

## □ 웹서버 시스템 구성

◆ 보통 HTTP서버+웹응용프레임워크+데이터베이스로 구성

➤ LAMP(Linux+Apache+MySQL+PHP), MAMP, WAMP

◆ HTTP서버

➤ 일반적인 HTTP 서버 사용 가능

이름	사용 플랫폼	라이선스	설 명
Apache	Unix, Linux, Windows, Mac	GPL	- 단일 쓰레드로 동작하는 HTTP 서버. 내부 연결 다중화 - CGI 프로그램, 자동 디렉터리 생성, 자동 압축 해제. - <a href="http://www.boa.org">http://www.boa.org</a>
lighttpd	Unix, Linux, Windows, Mac	BSD	- FastCGI, SCGI, HTTP proxy, WebDAV 지원 - OpenSSL을 통한 SSL, TLS 지원 - <a href="http://www.lighttpd.net">http://www.lighttpd.net</a>
nginx	Unix, Linux, Windows, Mac	BSD	- 동적 웹 페이지를 제공하는 HTTP 서버. - 고부하(동시 10000개 접속)에도 저메모리(~2.5MB) 처리. - <a href="http://nginx.org/">http://nginx.org/</a>

# 웹 서버

## □ 웹서버 시스템 구성

### ◆ 웹 응용 프로그램

- 동적인 웹 콘텐츠를 생성하는 CGI 및 스크립트 프로그램
- 웹 응용 프레임워크를 설치하여 웹 응용 프로그램을 구동

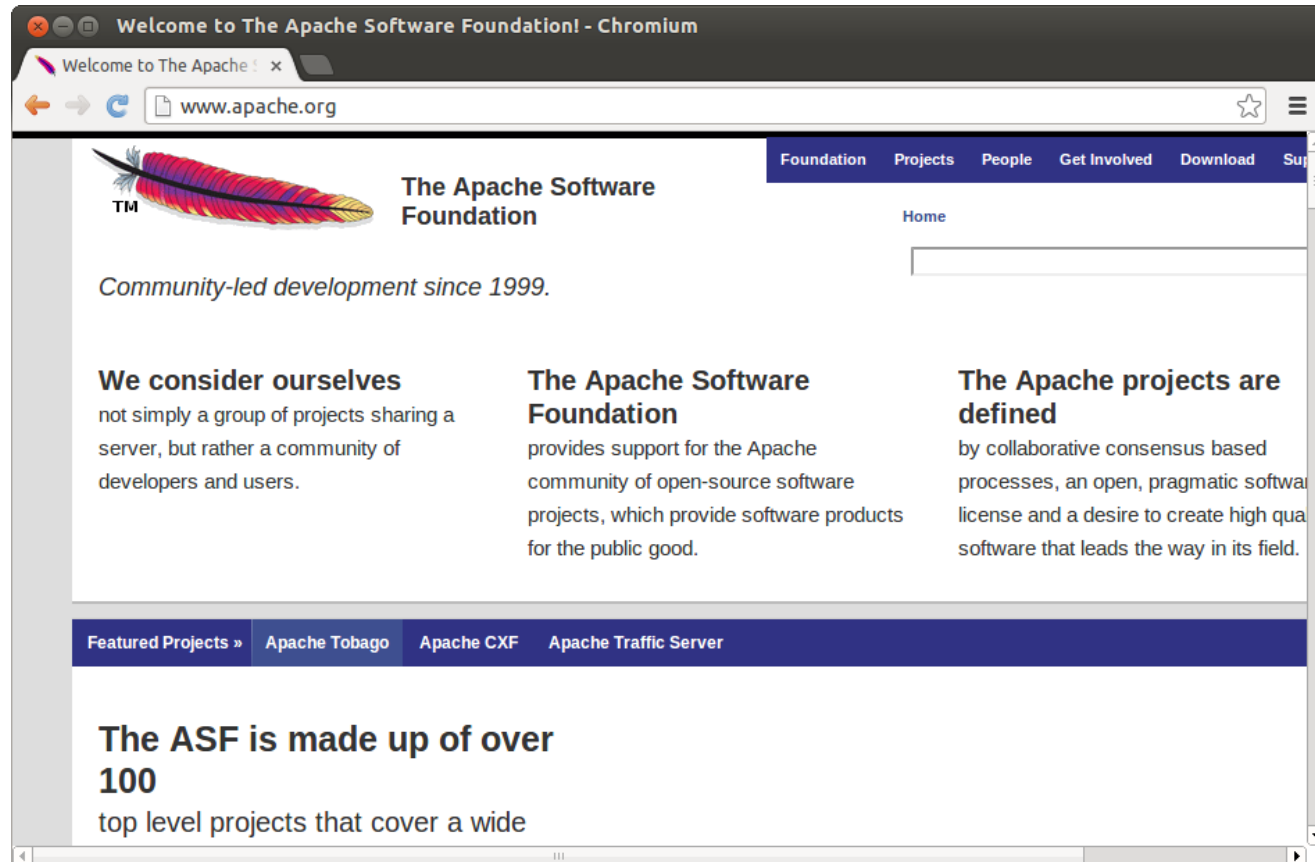
사용 언어	종류
Java	Struts, Wicket, Eclipse RAP, Google Web Toolkit, JSF, JBoss Seam, Spring
Javascript	node.js, SproutCore
Perl	Catalyst, Dancer
PHP	Zend Framework
Python	Bottle, CherryPy, Django, Flask, Grok, Pylons, web2py
Ruby	Camping, Ruby On Rails, Sinatra

### ◆ 데이터베이스 (DB)

- SQL 데이터베이스 등 사용
- MySQL – 오픈소스 SQL DB
- SQLite – 하나의 파일이나 메모리에 데이터베이스를 두는 SQL DB

# Apache

- 세계에서 가장 많이 사용하는 웹서버
- 홈페이지 : <http://www.apache.org>



# Apache 설치

## ■ 패키지 설치

```
pi@raspberrypi ~ $ sudo apt-get update  
pi@raspberrypi ~ $ sudo apt-get install apache2  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:  
  apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common  
  libapr1  
...
```

## ■ 실행

```
pi@raspberrypi ~ $ sudo service apache2 start
```

## ■ 부팅 시 자동 재실행

```
pi@raspberrypi ~ $ sudo update-rc.d apache2 defaults
```



# Apache 설치

## ■ 설정파일

◆ /etc/apache2/apache2.conf 파일

◆ 실제 VirtualHost 의 설정은 /etc/apache2/sites-available/default 파일에 있음

◆ 지시어

이름	내 용
<VirtualHost <ip:port> /> .. </VirtualHost>	웹 서버가 실행되는 가상 호스트의 모든 정보를 저장하는 태그
ServerAdmin	관리자 계정
DocumentRoot	HTML 이 시작될 루트(root) 디렉토리
ScriptAlias <path1> <path2>	가상 경로를 스크립트 서비스 디렉토리로 매핑
<Directory <path> > ... </Directory>	디렉토리 접근 정보를 저장하는 태그 Options, AllowOverride, Order, Allow(접근 허용) 등의 속성 저장
ErrorLog	웹 서버 수행 동안 발생하는 에러 메시지를 저장할 파일.
AccessLog	웹 서버를 접속할 때의 정보를 기록하는 파일.

# Apache 설치

## ◆ /etc/apache2/sites-available/default 파일 예

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/html
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>

    ScriptAlias /cgi-bin/ /var/www/cgi-bin/
```

```
    <Directory "/var/www/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews
        +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    # Possible values include: debug, info, notice, warn,
    error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog ${APACHE_LOG_DIR}/access.log
    combined
</VirtualHost>
```

# Apache 설치

## ■ 소유권 변경

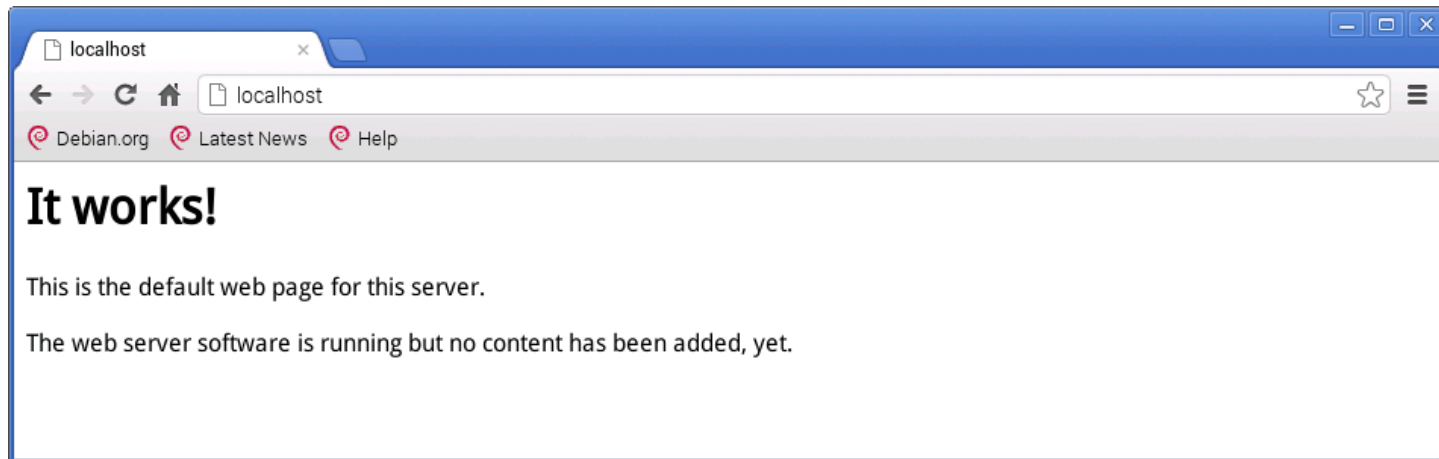
### ◆ 웹사이트 수정 및 접근 용이

```
pi@raspberrypi ~ $ sudo chown www-data:www-data /var/www  
pi@raspberrypi ~ $ sudo chmod -R 775 /var/www  
pi@raspberrypi ~ $ sudo usermod -a -G www-data pi
```

### ◆ 또는

```
pi@raspberrypi ~ $ sudo chown -R pi /var/www  
pi@raspberrypi ~ $ sudo chmod -R 775 /var/www
```

## ■ 실행



# Flask 마이크로 웹 응용 프레임워크 설치

## Python-bottle 패키지 설치

```
pi@raspberrypi ~ $ sudo apt-get install python3-flask
```

### ◆ 또는 최신버전 설치

```
pi@raspberrypi ~ $ sudo apt-get install python3-pip
```

```
pi@raspberrypi ~ $ sudo pip3 install flask
```

## 웹서버 실행

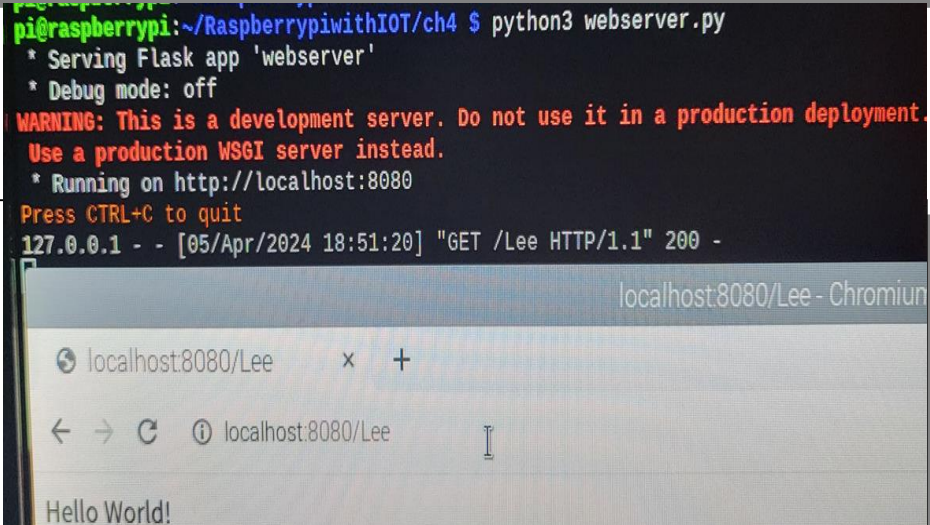
### ◆ Webserver.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/Lee')
def index(name='World'):
    return 'Hello %s!' % name
```

```
pi@raspberrypi ~ $ python webserver.py
Serving Flask app 'webserver'
Listening on http://localhost:8080/Lee
Use Ctrl-C to quit.
```



The screenshot shows a terminal window on a Raspberry Pi with the command `python3 webserver.py` executed. The output indicates that the Flask app 'webserver' is running on `http://localhost:8080`. A warning message states: "WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead." A GET request from `127.0.0.1` at `05/Apr/2024 18:51:20` returns a 200 status. Below the terminal, a Chromium browser window is open at `localhost:8080/Lee`, displaying the text "Hello World!"

# PHP

- 동적인 웹페이지를 만들기 위해 설계된 프로그래밍 언어
- Rasmus Lerdorf 에 의해 1995년 개발
- PHP 처리 방식
  - ◆ Apache 와 같이 PHP 처리 기능을 가진 웹서버가 HTML 문서 내의 PHP 코드를 처리
  - ◆ 근래에는 PHP 코드를 HTML 문서와 분리하여 php-fpm(PHP FastCGI Process Manager) 를 통해 처리
- 홈페이지 : <http://www.php.net>
- PHP 패키지 설치

```
pi@raspberrypi ~ $ sudo apt-get install php5
```

- ◆ **Mysql 연동 패키지 설치**

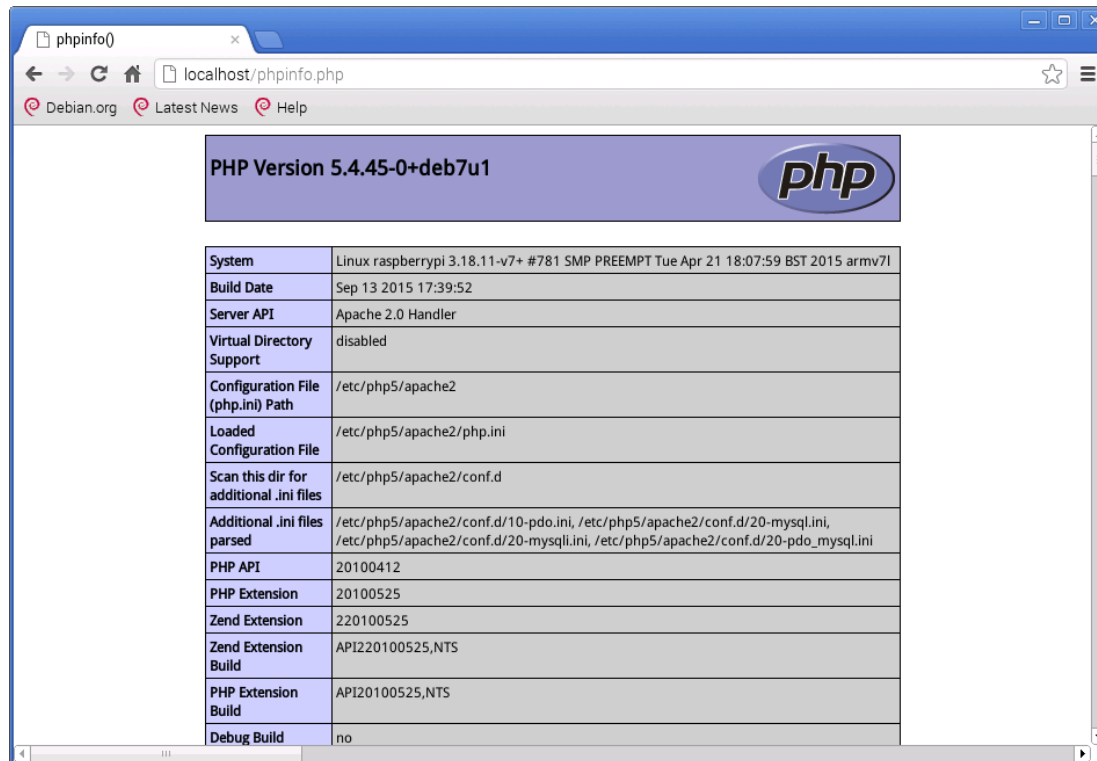
```
pi@raspberrypi ~ $ sudo apt-get install php5-mysql
```

# PHP 설치

## ■ 웹서버 실행

### ◆ /var/www/html/phpinfo.php 파일

```
<?php phpinfo(); ?>
```



System	Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l
Build Date	Sep 13 2015 17:39:52
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysqli.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no

## 4.3 파이썬 웹 프로그래밍

### ■ HTML (HyperText Markup Language)

◆ 하이퍼텍스트 기능을 가진 **html** 문서를 만드는 언어

### ■ HTML 문서

◆ 흔히 태그 (tag) 로 구성

➤ 태그는 보통 시작 부분 **<tag>** 와 끝부분 **</tag>** 으로 구성된 형식

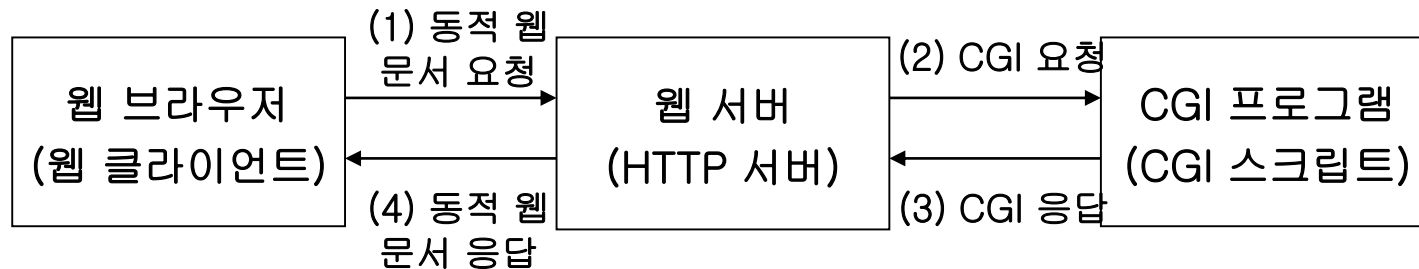
◆ 모든 문서는 **<html>** 로 시작해서 **</html>** 로 끝나며, 문서의 내부  
는 **head** 와 **body** 두 부분으로 구성

```
<html>
<head>
<title>Embedded Webserver</title>
</head>
<body>
<H1>Embedded Webserver</H1><br />
Welcome to Embedded Linux World!
</body>
</html>
```

## 4.3.2 CGI

### ■ CGI (Common Gateway Interface)

- ◆ 웹 서버와 외부 응용 프로그램 사이를 연결하는 표준 규약
- ◆ HTML 의 기능을 보충하여 http text 는 정적인 반면, CGI 는 동적으로 text 정보 변경 가능.
- ◆ CGI 프로그램이 가능한 언어들
  - C/C++, Fortran, PERL, TCL, Any Unix Shell, Java/JavaScript 등





# CGI 구현

## ■ HTML 의 FORM 태그를 통해 수행

## ■ Fill-out Form 을 이용한 CGI

- ◆ 웹 클라이언트 사용자가 어떤 내용을 입력하고 이 내용을 웹 서버 쪽으로 전달하는 문자 입력 상자.
- ◆ 사용자와 등록(registration), 명령(order), 질의(query) 등의 인터페이스를 추가
- ◆ 여러 개의 fill-out form 허용하지만 중첩(nested) 은 안 됨.
- ◆ 형식

```
<FORM ACTION="URL" METHOD=...> ... </FORM>
```

➤ **ACTION** : 웹 서버의 수행 CGI 프로그램

➤ **METHOD** : 웹 서버로 전송하는 방법, GET 과 POST 가 있음

# CGI 구현

## ■ FORM 을 구성하는 입력 태그

### ◆ CGI 의 FORM 내부의 입력 요소 정의 형식

```
<INPUT TYPE=... NAME=... SIZE=... VALUE=...>
```

➤ **NAME** : 입력 요소 구분 이름, **SIZE** : 크기, **VALUE** : 초기값

### ◆ TYPE : 종류

- **TEXT** – 텍스트를 입력할 수 있는 상자 서식
- **PASSWORD** – 텍스트를 입력받지만 내용을 \* 로 보여주는 서식
- **CHECKBOX** – 토글 버튼 서식, 하나를 선택하면 나머지는 해제
- **RADIO** – 토글 버튼 서식, 여러 개를 동시에 선택 가능
- **SUBMIT** – 푸시 버튼으로 FORM의 ACTION 인 CGI 프로그램 수행
- **RESET** – 푸시 버튼으로 현재의 FORM 입력 요소들의 값이 초기화
- **SELECT** – 선택 옵션이 있어서 사용자가 하나를 선택할 수 있음
  - <SELECT NAME=...><OPTION> xxx <OPTION> ...</SELECT>
- **TEXTAREA** – 2 줄 이상의 텍스트를 입력할 수 있는 서식

# CGI 구현

## ■ CGI 프로그램 작성 (C 언어)

### ◆ HTML로부터 CGI로 정보 전달

- 요청 **FORM**의 **METHOD**가 **POST** : 표준 입력으로부터 전달
- 요청 **FORM**의 **METHOD**가 **GET** : 환경변수 **QUERY\_STRING**으로 전달
  - getenv() 함수 사용
  - #include <stdlib.h> .... char \*cl = (char \*) getenv("QUERY\_STRING");

### ◆ 실제 전달되는 값은 "&"로 구분

### ◆ 실제 예)

#### ➤ HTML 문서

```
<form method=get action="cgi-bin/led.cgi">  
<input type="text" name="value" maxlength="4" size="4">  
<input type="submit" name="button" value="input">
```

- **a1**을 입력한 경우 실제 전달되는 값은 **"value=a1&"**

# CGI 구현

## ■ CGI 관련 환경변수

이름	설 명
CONTENT_TYPE	질의 데이터 타입
CONTENT_LENGTH	질의 데이터 길이
HTTP_ACCEPT	클라이언트가 수락한 <b>MIME</b> 타입
HTTP_COOKIE	키와 값으로 구성된 지속 쿠키 값
HTTP_REFERER	참조하는 <b>URL</b>
HTTP_USER_AGENT	요청하는 사용자 정보, 보통 클라이언트 웹 브라우저 이름
PATH_INFO	전달된 추가 경로 정보
QUERY_STRING	질의 문자열
REMOTE_ADDR	원격 클라이언트 <b>IP</b> 주소
REMOTE_HOST	원격 클라이언트 호스트 이름
REQUEST_METHOD	요청 메소드 ('GET' 또는 'POST')
SCRIPT_NAME	<b>CGI</b> 스크립트 프로그램 이름
SERVER_NAME	서버 호스트 이름
SERVER_PORT	서버 호스트 포트
SERVER_PROTOCOL	서버 프로토콜
SERVER_SOFTWARE	서버 소프트웨어 이름과 버전

# CGI 구현

## ■ CGI 프로그램 작성 (Python 언어)

### ◆ cgi, cgitb 패키지 import

- **cgi** – CGI 스크립트 처리 모듈
- **cgitb** – CGI 스크립트 디버깅 모듈

### ◆ FieldStorage 객체 생성

- **HTML FORM** 구문을 통해 전달된 질의 문자열 접근

```
import cgi  
form = cgi.FieldStorage()  
value = form.getValue('value')
```

### ◆ CGI 프로그램 출력은 표준출력(sys.stdout 또는 print 구문) 으로

- **HTTP** 헤더와 **HTML** 데이터로 구성하고 사이클 빈 줄로 구분

```
print 'Content-type: text/html\r'  
print '\r'  
print '<html>'  
print '<head><title>CGI Program</title></head>'  
print '<body><H1>Hello World</H1></body>'  
print '</html>'
```

# CGI 환경 설정

## ■ Apache 웹 서버 환경 설정

### ◆ /etc/apache2/sites-available/default 파일 수정

```
ScriptAlias /cgi-bin/ /var/www/cgi-bin/
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Order allow,deny
    Allow from all
</Directory>
```

### ◆ cgi-bin 디렉토리 생성 및 권한

```
$ sudo a2enmod cgi
$ systemctl restart apache2
$ sudo service apache2 restart
$ mkdir -p /var/www/cgi-bin
$ sudo chmod 775 /var/www/cgi-bin
$ sudo chown www-data:www-data /var/www/cgi-bin
```

```
pi@raspberrypi:~ $ sudo a2enmod cgi
Your MPM seems to be threaded. Selecting cgid instead of cgi.
Module cgid already enabled
pi@raspberrypi:~ $ systemctl restart apache2
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
'apache2.service' 서비스 유닛을 다시 시작하려면 인증이 필요합니다.
Authenticating as: pi, (pi)
Password:
==== AUTHENTICATION COMPLETE ====
pi@raspberrypi:~ $ sudo service apache2 restart
pi@raspberrypi:~ $ $ sudo chmod 775 /var/www/cgi-bin
-bash: $: 명령어를 찾을 수 없음
pi@raspberrypi:~ $ sudo chmod 775 /var/www/cgi-bin
pi@raspberrypi:~ $ sudo chown www-data:www-data /var/www/cgi-bin
pi@raspberrypi:~ $
```

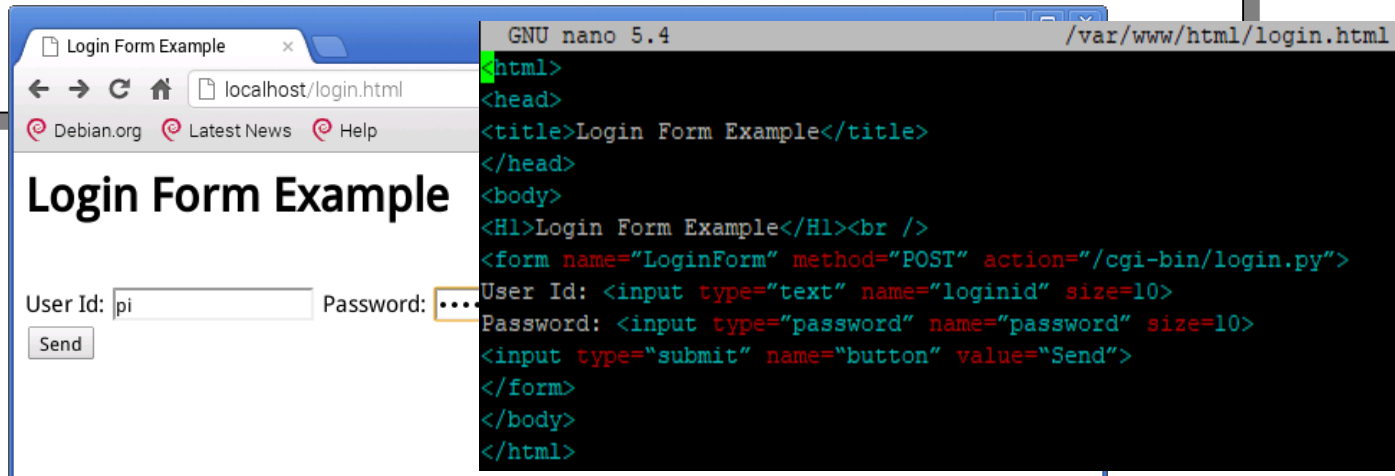
# CGI 프로그램의 예

## 로그인 웹 화면

◆ 사용자 아이디와 비밀번호를 입력받은 HTML 문서

**\$ sudo nano /var/www/html/login.html**

```
<html>
<head>
<title>Login Form Example</title>
</head>
<body>
<H1>Login Form Example</H1><br />
<form name="LoginForm" method="POST" action="/cgi-bin/login.py">
User Id: <input type="text" name="loginid" size=10>
Password: <input type="password" name="password" size=10>
<input type="submit" name="button" value="Send">
</form>
</body>
</html>
```



# CGI 프로그램의 예

## 로그인 화면 프로그램

### 로그인 화면 파이썬 CGI 프로그램 - /var/www/cgi-bin/login.py

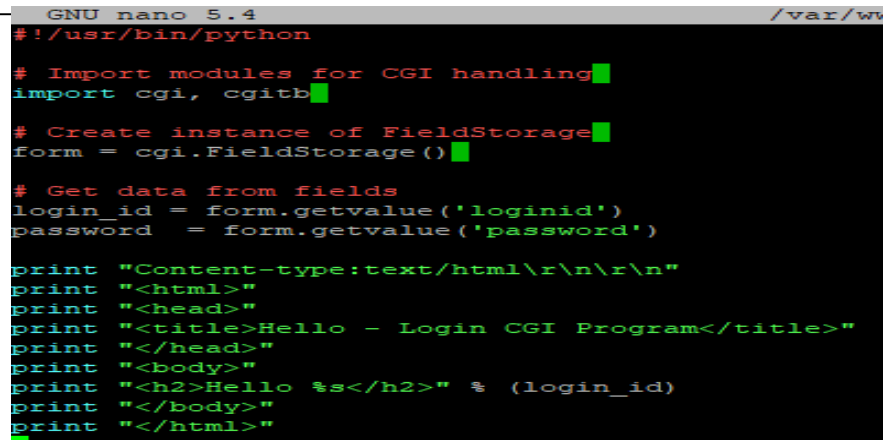
```
#!/usr/bin/python

# Import modules for CGI handling
import cgi, cgitb

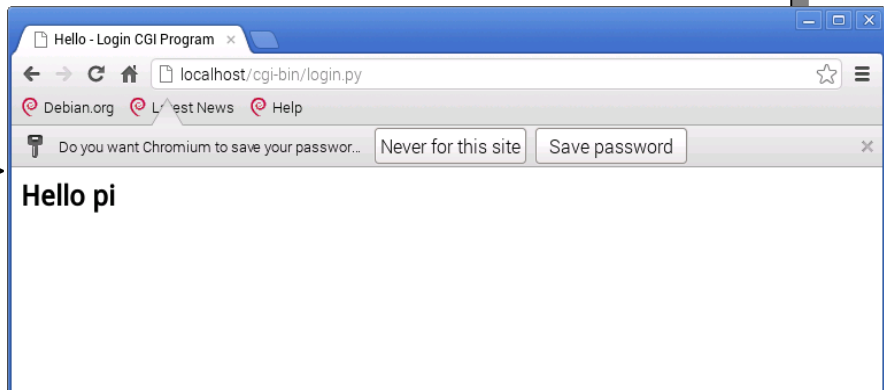
# Create instance of FieldStorage
form = cgi.FieldStorage()

# Get data from fields
login_id = form.getvalue('loginid')
password = form.getvalue('password')

print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Hello - Login CGI Program</title>"
print "</head>"
print "<body>"
print "<h2>Hello %s</h2>" % (login_id)
print "</body>"
print "</html>"
```



```
GNU nano 5.4 /var/www/cgi-bin/login.py
#!/usr/bin/python
# Import modules for CGI handling
import cgi, cgitb
# Create instance of FieldStorage
form = cgi.FieldStorage()
# Get data from fields
login_id = form.getvalue('loginid')
password = form.getvalue('password')
print "Content-type:text/html\r\n\r\n"
print "<html>"
print "<head>"
print "<title>Hello - Login CGI Program</title>"
print "</head>"
print "<body>"
print "<h2>Hello %s</h2>" % (login_id)
print "</body>"
print "</html>"
```



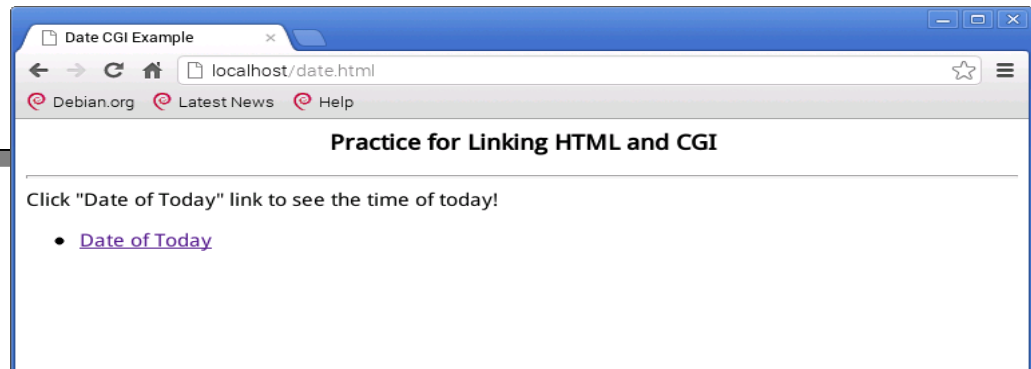


# CGI 프로그램의 예

## ■ 날짜 출력 프로그램

### ◆ CGI 프로그램 호출 HTML 문서 - /var/www/html/date.html

```
<HTML>
<HEAD>
<TITLE>Date CGI Example</TITLE>
</HEAD>
<BODY>
<CENTER>
<H3> Practice for Linking HTML and CGI</H3>
</CENTER>
<HR SIZE=3>
Click "Date of Today" link to see the time of today!<BR>
<UL>
    <LI><a href=/cgi-bin/date.py>Date of Today </a>
</UL>
</BODY>
</HTML>
```



# CGI 프로그램의 예

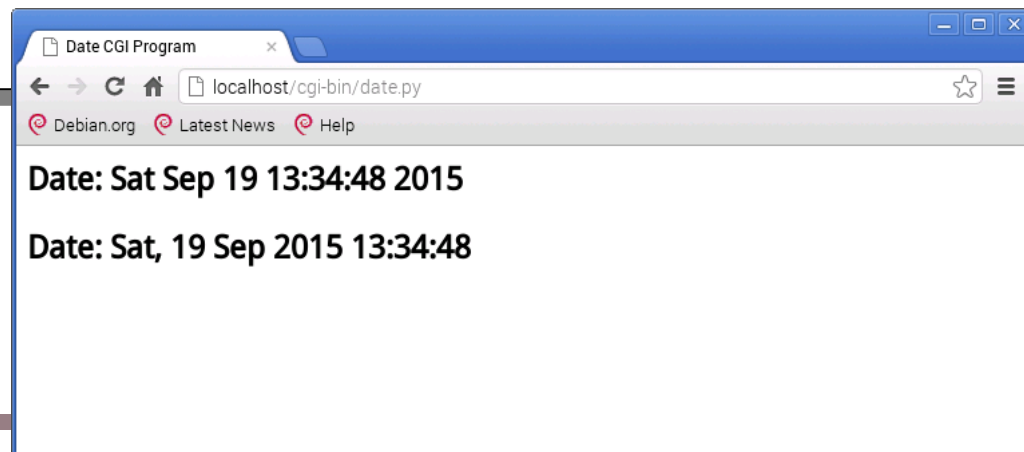
## ■ 날짜 출력 프로그램

### ◆ 날짜 출력 파이썬 CGI 프로그램 - /var/www/cgi-bin/date.py

```
#!/usr/bin/python

# Import modules for CGI handling
import time, datetime

Print "Content-type:text/html\r\n\r\n"
Print "<html>"
Print "<head>"
Print "<title>Date CGI Program</title>"
print "</head>"
print "<body>"
print "<h2>Date: %s</h2>" % time.ctime()
time_format = "%a, %d %b %Y %H:%M:%S %Z"
print "<h2>Date: %s</h2>" % (datetime.datetime.now().strftime(time_format))
print "</body>"
print "</html>"
```



# Flask+web 제어

# 공대선배 라즈베리파이썬 #12 python flask (https://www.youtube.com/watch?v=aEoP15gkarQ)

from flask import Flask # flask 모듈을 불러옴

import RPi.GPIO as GPIO # 라즈베리파이 GPIO 관련 모듈을 불러옴

GPIO.setmode(GPIO.BCM) # GPIO 핀들의 번호를 지정하는 규칙 설정

red\_pin = 14 # 빨간 LED 핀은 라즈베리파이 GPIO 14번핀으로

green\_pin = 15 # 초록 LED 핀은 라즈베리파이 GPIO 15번핀으로

blue\_pin = 18 # 파란 LED 핀은 라즈베리파이 GPIO 18번핀으로

GPIO.setup(red\_pin, GPIO.OUT) # 각각 LED 핀들을 출력으로 설정

GPIO.setup(green\_pin, GPIO.OUT)

GPIO.setup(blue\_pin, GPIO.OUT)

```
pi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $ sudo netstat -tlnp | grep 9999
pi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $
```

app = Flask(\_\_name\_\_) # Flask라는 이름의 객체 생성

@app.route('/') # 기본 주소

def hello(): # 해당 주소에서 실행되는 함수 정의

return "LED 제어를 위해 주소창을 변경하세요" # 반드시 return이 있어야하며, 해당 값을 화면에 보여줌

@app.route('/red\_on') # IP주소:port/red\_on 을 입력하면 나오는 페이지

def red\_on(): # 해당 페이지의 뷰함수 정의

GPIO.output(red\_pin, GPIO.HIGH) # 빨간 LED 핀에 HIGH 신호 인가(LED 켜짐)

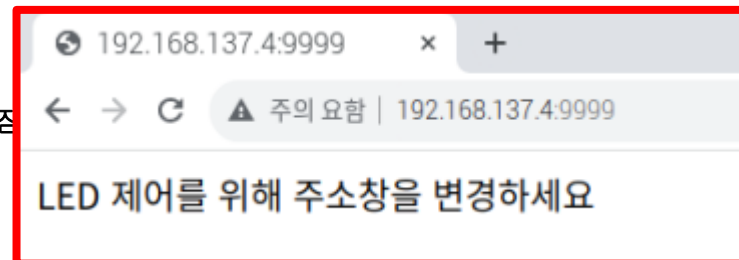
return "red LED on" # 뷰함수의 리턴값

@app.route('/green\_on') # IP주소:port/green\_on 을 입력하면 나오는 페이지

def green\_on(): # 해당 페이지의 뷰함수 정의

GPIO.output(green\_pin, GPIO.HIGH) # 초록 LED 핀에 HIGH 신호 인가(LED 켜짐)

return "green LED on"



# Flask+web 제어

```
@app.route('/blue_on')
```

```
def blue_on():
```

```
    GPIO.output(blue_pin, GPIO.HIGH)
```

```
    return "blue LED on"
```

```
@app.route('/off')      # IP주소:port/off 를 입력하면 나오는 페이지
```

```
def off():              # 해당 페이지의 뷰함수 정의
```

```
    GPIO.output(red_pin, GPIO.LOW) # 각각의 LED핀에 LOW 신호를 인가하여 LED 끄
```

```
    GPIO.output(green_pin, GPIO.LOW)
```

```
    GPIO.output(blue_pin, GPIO.LOW)
```

```
    return "all LED off"
```

```
@app.route('/clean_up')
```

```
def clean_up():
```

```
    GPIO.cleanup()
```

```
    return "clean up"
```

```
pi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $ sudo netstat -tlnp | grep 8080
tcp        0      0 0.0.0.0:8080          0.0.0.0:*           LISTEN      488/motion
pi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $ sudo netstat -tlnp | grep 8081
tcp        0      0 0.0.0.0:8081          0.0.0.0:*           LISTEN      488/motion
pi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $ sudo netstat -tlnp | grep 8082
tcp        0      0 0.0.0.0:8082          0.0.0.0:*           LISTEN      1639/python3
```

```
if __name__ == "__main__": # 웹사이트를 호스팅하여 접속자에게 보여주기 위한 부분
```

```
    app.run(host="192.168.137.4", port = "9999")
```

```
    # host는 현재 라즈베리파이의 내부 IP, port는 임의로 설정
```

```
    # 해당 내부 IP와 port를 포트포워딩 해두면 외부에서도 접속가능
```

```

pi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $ python3 flaskLED.py
/home/pi/RaspberrypiwithIOT/ch4/flask/flaskLED.py:15: RuntimeWarning: This channel is already in use, cont
PIO.setwarnings(False) to disable warnings.
  GPIO.setup(red_pin, GPIO.OUT) # 각각 LED 핀들을 출력으로 설정
/home/pi/RaspberrypiwithIOT/ch4/flask/flaskLED.py:16: RuntimeWarning: This channel is already in use, cont
PIO.setwarnings(False) to disable warnings.
  GPIO.setup(green_pin, GPIO.OUT)
/home/pi/RaspberrypiwithIOT/ch4/flask/flaskLED.py:17: RuntimeWarning: This channel is already in use, cont
PIO.setwarnings(False) to disable warnings.
  GPIO.setup(blue_pin, GPIO.OUT)
* Serving Flask app "flaskLED" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.137.4:9999/ (Press CTRL+C to quit)
192.168.137.4 - - [14/Apr/2024 23:24:12] "GET / HTTP/1.1" 200 -
192.168.137.4 - - [14/Apr/2024 23:24:13] "GET /favicon.ico HTTP/1.1" 404 -
192.168.137.4 - - [14/Apr/2024 23:27:51] "GET /green_on HTTP/1.1" 200 -
192.168.137.4 - - [14/Apr/2024 23:28:21] "GET /clean_up HTTP/1.1" 200 -
^Cpi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $ nano flaskLED.py
pi@raspberrypi:~/RaspberrypiwithIOT/ch4/flask $ python3 flaskLED.py
* Serving Flask app "flaskLED" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.137.4:9999/ (Press CTRL+C to quit)
192.168.137.4 - - [14/Apr/2024 23:32:20] "GET /green_on HTTP/1.1" 200 -

```

