Python Web Programming

for Raspberry Pi

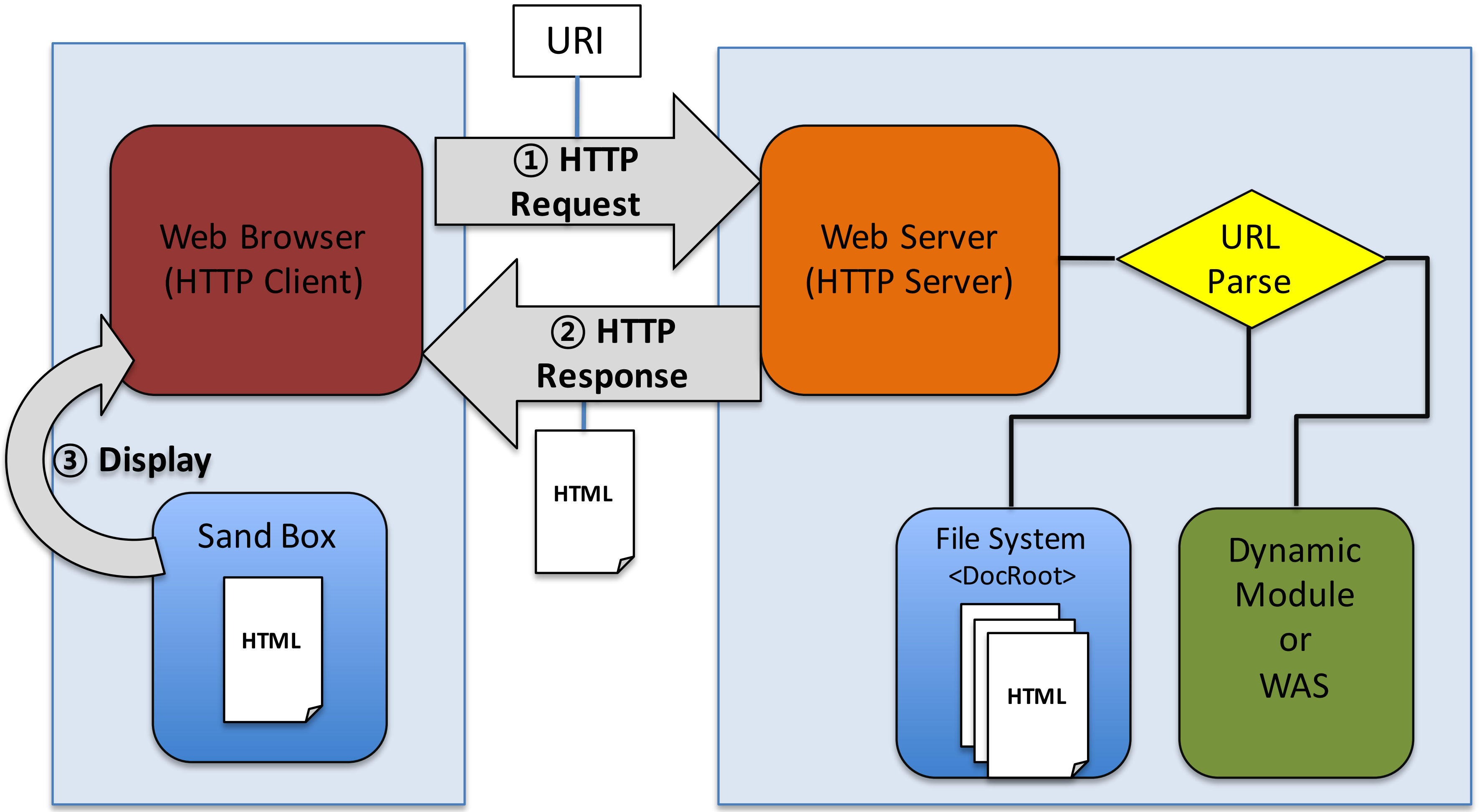
Rev. R610

이세우 (dltpdn@gmail.com)

세부목차

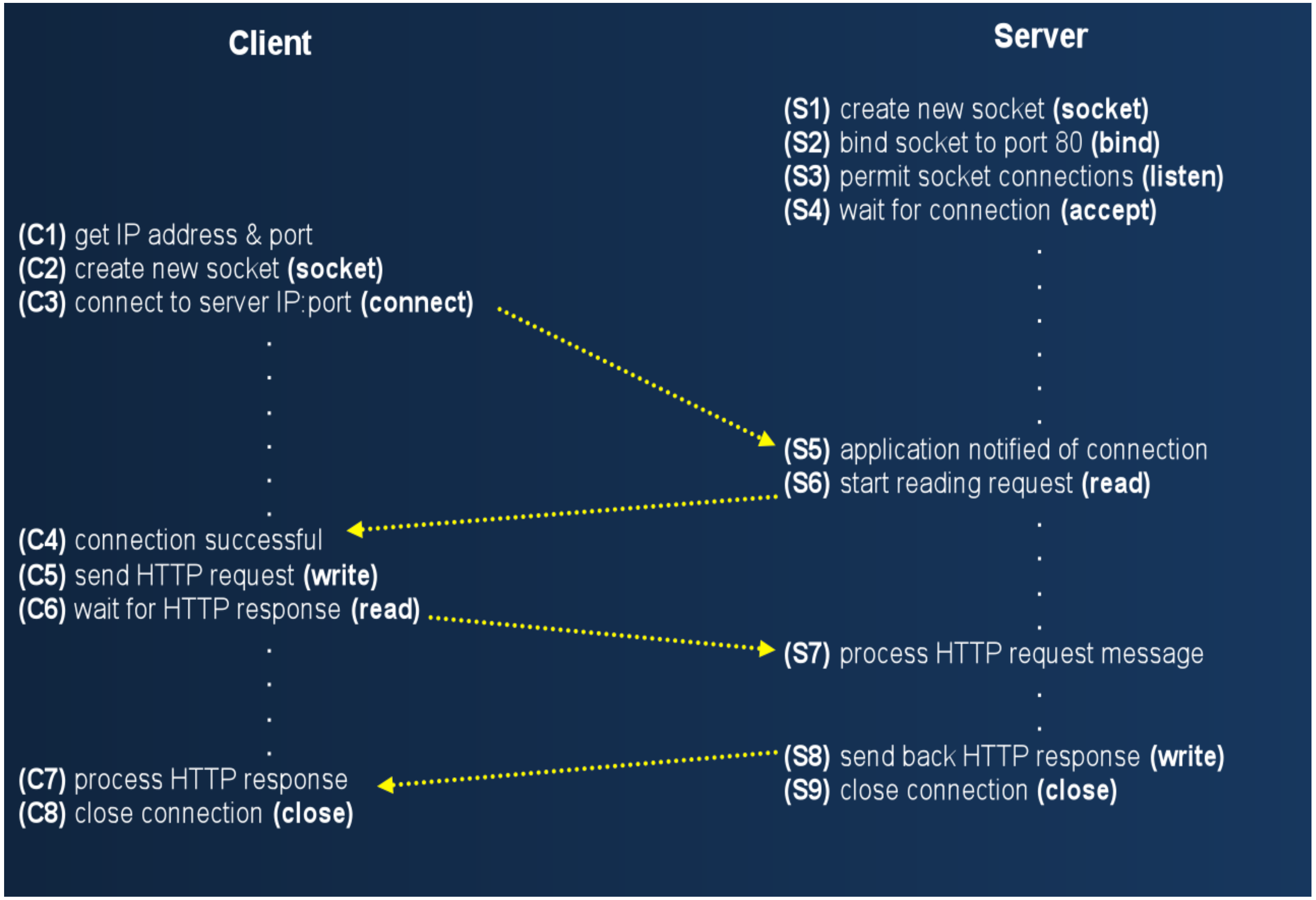
Web Architecture

❖ Web Architecture



Web Architecture

❖ Client/Server Timeline


Client
Server
(S1) create new socket **(socket)**
(S2) bind socket to port 80 **(bind)**
(S3) permit socket connections **(listen)**
(S4) wait for connection **(accept)**
(C1) get IP address & port
(C2) create new socket **(socket)**
(C3) connect to server IP:port **(connect)**
(S5) application notified of connection
(S6) start reading request **(read)**
(C4) connection successful
(C5) send HTTP request **(write)**
(C6) wait for HTTP response **(read)**
(S7) process HTTP request message
(S8) send back HTTP response **(write)**
(S9) close connection **(close)**
(C7) process HTTP response
(C8) close connection **(close)**

세부목차

HTTP

- **Hyper-Text Transfer Protocol over TCP/IP**
- **History**
  - HTTP 0.9 : No Spec Sheet
  - HTTP 1.0 :
    - Fix : 1996' IETF RFC 1945
    - Difference between spec and implementation
    - Added Header, GET Method
  - HTTP 1.1 :
    - Fix : 1997' IEFT RFC 2068,
    - Rev. 1999', RFC 2616(Current Version)
    - Cache Control, connection keep
    - http://tools.ietf.org/html/rfc2616
- **Feature**
  - Connectionless
  - Stateless
  - Request and Response

HTTP

❖ HTTP Request Structure

Division	Example
Request line <request_method><URI><HTTP_Ver>	GET /index.html HTTP/1.1
Request Header (General |Request | Entity Header)* <header_name> : <header_value><CR><LF>	Host : www.example.com:80 User-Agent : Mozilla/5.0 Accept : text/html Accept-Language : en-us Accept-Encoding : gzip, delate Date : Tue, 3 Oct 1974 02:16:00 GMT Connection : keep-alive
An Empty line <CR><LF>	<carriage return>
Optional Message Body	POST Data

HTTP

Request Methods

Request Method	Description
GET	지정된 URL의 정보를 가져온다.
POST	지정된 URL로 Body에 포함된 정보를 제출한다.
PUT	지정된 URL에 저장될 정보를 전달한다.
DELETE	지정된 Resource를 삭제한다.
HEAD	응답 헤더를 요청한다. Response Body가 없는 걸 제외 하면 GET과 동일
OPTIONS	지정된 URL이 지원하는 HTTP methods를 요청
TRACE	Echoes back 수신된 메시지를 다시 전송한다.
CONNECT	Proxy 사용에 예약되어 있다.

HTTP

❖ HTTP Request Example

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:5.0.1) Gecko/20100101
Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: UTF-8,*
Connection: keep-alive
Referer: http://www.google.com/url?sa=t&source=web&cd=1
Cookie: mediaWiki.user.bucket%3Aext.articleFeedback-options=8%3Ashow;  If-Modified-
Since Sat, 13 Aug 2011 19:57:28 GMT
Cache-Control:     max-age=0
```

HTTP

❖ HTTP Response Structure

Division	Example
Response line <HTTP ver><status code><status-message>	HTTP/1.1 200 OK
Response Header (General |Response | Entity Header)* <header_name>:<header_value><CR><LF>	Host : www.example.com:80 User-Agent : Mozilla/5.0 Accept : text/html Accept-Language : en-us Accept-Encoding : gzip, delate Date : Tue, 3 Oct 1974 02:16:00 GMT Connection : keep-alive **Content-Type : text/html;charset=UTF-8**
An Empty line	<CR><LF>, carriage return
Message Body	HTML Contents

HTTP

❖ HTTP Response Example

```
HTTP/1.1 200 OK
Date: Sun, 10 Oct 2011 03:21:12 GMT
Server: Apache/2
Cache-Control: no-store, no-cache, must-revalidate, post-check=0
Content-Encoding:gzip
Connection:close
Content-Type : text/html;charset=UTF-8

<!DOCTYPE html>
<html>
<head>
</head>
<body>
…
… 생략 …
…
```

HTTP

❖ Response Status Code

Range	Status Code	Description
1xx Informational	100	Continue
	101	Switching protocols
2xx Success	200	OK
	201	Created
	202	Accepted
	203	Non-authoritive information
	204	No connect
	205	Reset content
	206	Partial content
	207	Multi-Status(WebDAV)
	226	IM Used

HTTP

❖ Response Status Code

Range	Status Code	Description
3xx Redirection	300	Multiple choices
	301	Moved Permanently
	302	Found(Redirection)
	303	See other
	304	Not Modified
	305	Use proxy
	306	Switch proxy
	307	Temporary Redirect
	308	Resume Incomplete

HTTP

❖ Response Status Code

Range	Status Code	Description
4xx Client Error	400	Bad Request
	401	Unauthorized
	402	Payment required
	403	Forbidden
	404	Not found
	405	Method not allowed
	406	Not Acceptable
	407	Proxy authentication required
	408	Request timeout
	409	Confilct
	410	Cone

HTTP

❖ **Response Status Code**

Range	Status Code	Description
5xx Server Error	500	Internal Server Error
	501	Not Implemented
	502	Bad Gateway
	503	Service Unavailable
	504	Gateway Timeout
	505	HTTP Version not supported
	506	Variant Also negotiates
	507	Insufficient storage (WebDAV)
	509	Bandwidth limit exceeded
	510	Not Extended

HTTP

- Multipurpose Internet Media Extensions Type
- Internet Media Type
- Content-type
- Syntax

```
<type>/<subtype>;[<parameter-name>=<parameter-value>
```

- Example

```
Content-Type : text/html;charset=UTF-8
```

HTTP

❖ Socket Webserver

```
from  socket import *

sock = socket(AF_INET, SOCK_STREAM)
sock.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
sock.bind(('', 8080))
sock.listen(1)
print 'server listening on 8080...'
while True:
   conn, addr = sock.accept()
   req = ''
   while True:
      req += conn.recv(1024)
      if req.endswith('\r\n\r\n'):
         req_line = req.split('\r\n')[0]
         print req_line
         method, url, ver = req_line.split()
         print url
         break
   conn.send("HTTP/1.1 200 OK\nContent-Type:text/html\n\n<h1>Welocome to My server</h1>\n")
   conn.close()
sock.close()
```

HTTP

❖ SimpleHTTPServer

- 현재 디렉토리 List-up 기능을 구현해 놓은 예시 클래스

```
import SimpleHTTPServer
import SocketServer

PORT = 8080
httpd = SocketServer.TCPServer(("", PORT), SimpleHTTPServer.SimpleHTTPRequestHandler)

print "server on%d" %PORT
httpd.serve_forever()
```

HTTP

❖ BaseHTTPServer

- BaseHTTPReuqestHandler를 상속해서 Custom 서버를 구성

```
import BaseHTTPServer
import SocketServer


class MyHandler(BaseHTTPServer.BaseHTTPRequestHandler):
   def do_GET(self):
      self.send_response(200)
      self.send_header('Content-Type', 'text/html')
      self.end_headers()
      self.wfile.write('<h1>Helo! Welcome to My Simple Server</h1>')
      return


PORT = 8080
httpd = SocketServer.TCPServer(("", PORT), MyHandler)

print "server on%d" %PORT
httpd.serve_forever()
```

HTTP

WSGI

- Web Server Gateway Interfacce
- Python을 위한 CGI 표준 규격 (PEP-333)

```
from wsgiref.simple_server import  make_server


def app(env, res):
   print env
   res_body = "<h1>Welcome to WSGI server</h1>"

   status = '200 OK'
   res_header  = [('Content-Type', 'text/html')]
   res(status, res_header)

   return [res_body]

httpd = make_server('localhost', 8080, app)
#httpd.handle_request()
httpd.serve_forever()
```

HTTP

❖ Http Client

```
import urllib

url = 'http://www.google.com'
stream = urllib.urlopen(url)
res = stream.read()
print res
```

세부목차

Flask

Flask

- http://flask.pocoo.org/
- WSGI를 기반으로 하는 micro framework
- Armin Ronacher, Austrian(http://lucumr.pocoo.org/)
- 경량 프레임웍
- 필요에 따라 확장 가능
- route 기능
- Jinja Template

Installation

- pip install flask

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "<h1>Hello flask</h1>"

if __name__ == "__main__":
    app.run()
```

Flask

❖ URL Routing

- 요청 URL/Method에 따른 개별 함수 핸들러 등록

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def root():
   return  '<h1> This is root page</h1><a href="/next">Go next</a>'

@app.route('/next')
def next():
   return  '<h1> This is Next page</h1><a href="/">Go Root</a>'


app.run()
```

Flask

❖ URL Routing Parameter

- 특정 URL의 하위 경로
- REST Style

```
from flask import Flask
app = Flask(__name__)

@app.route('/user/<id>', methods=['GET'])
def show_user(id):
   print id
   if id == "abc":
      return 'User id is %s, name is %s' %(id, 'Lee')
   elif id == "xyz":
      return 'User id is %s, name is %s' %(id, 'Kim')
   else:
      return 'No User id : %s' %id

@app.route('/post/<int:post_id>')
def show_post(post_id):
   return 'Post id : %d' %post_id

app.run()
```

Flask

❖ Static File

- route에 등록하지 않은 단순 파일 서비스

```
from flask import Flask

app = Flask(__name__)

@app.route('/<path:path>')
def static_file(path):
   return app.send_static_file(path)

@app.route('/')
def root():
   return "<h1>this is main page</h1>"


@app.route('/aaa.html')
def abc():
   return "<h1>this is abc.html</h1>"

app.run(port=8888)
```

Flask

❖ Parameter 수집

- GET 방식
  - from flask import request
  - request.args.get('name')
  - request.values['name']

```
from flask import Flask, request

app = Flask('my')

@app.route('/get_param', methods=['GET'])
def get_param():
   #id= request.args.get('id')
   id = request.values['id']
   pwd = request.args.get('pwd')
   return  'id: %s , pwd : %s' %(id, pwd)

@app.route('/')
def root():
   return '<a href="/get_param?id=abc&pwd=1234">get_param</a>'

app.run()
```

Flask

❖ **Parameter 수집**
- POST 방식
  - from flask import request
  - request.form[‘name’]
  - request.values[‘name’]

```
from flask import Flask, request, redirect
app = Flask(__name__)

@app.route('/')
def root():
  return  redirect('/static/form.html')

@app.route('/post_param', methods=['POST'])
def post_param():
#    id = request.form['id']
    id = request.values['id']
    pwd = request.form['pwd']
    return 'ID: %s, PWD:%s' %(id, pwd)

app.run()
```

Flask

❖ Parameter 수집

- POST 방식
- static/form.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
	<h1>Post Param Test</h1>
	<form action="/post_param" method="POST">
		<label>ID:</label><input name="id" type="text"/><br/>
		<label>PWD:</label><input name="pwd" type="text"/><br/>
		<input type="submit" value="전송"/>
	</form>
</body>
</html>
```

Flask

❖ Template

- Jinja2 (http://jinja.pocoo.org/docs/dev/)
  - 기본 템플릿 엔진
  - 정적인 HTML 파일에 데이타 합성
- render_template()

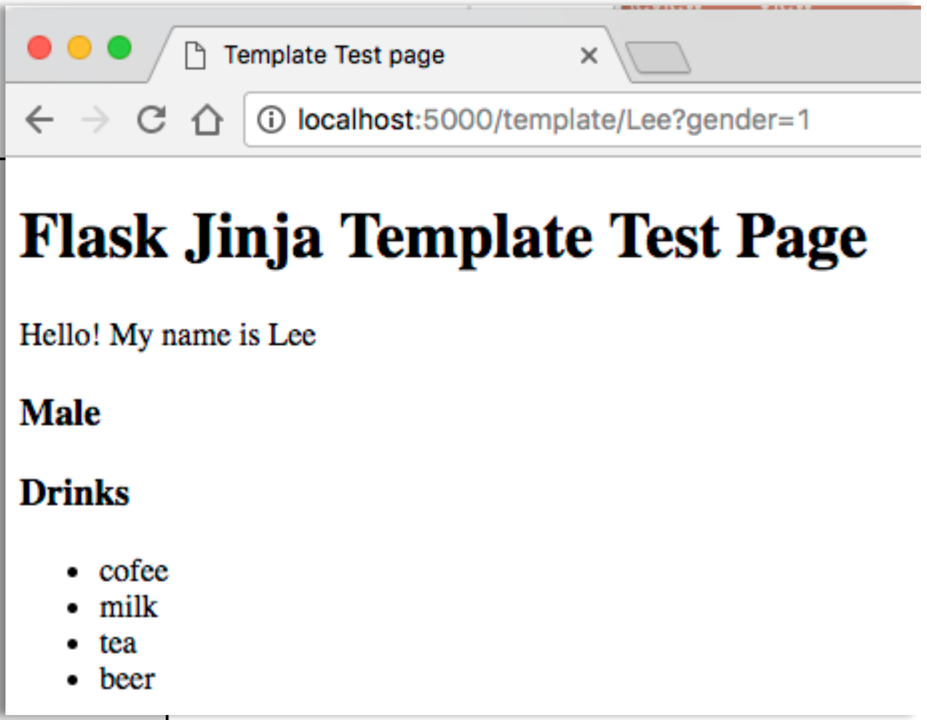
```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/template')
@app.route('/template/<name>')
def template_test(name=None):
    gender = request.args.get('gender')
    drinks = ['cofee', 'milk', 'tea', 'beer']
    return render_template('test.html',name=name, gender=gender, drinks=drinks)

app.run()
```

Flask

❖ Template

- templates/test.html

```
<!DOCTYPE html>
<html>
<head>
<title>Template  Test page</title>
</head>
<body>
        <h1>Flask Jinja Template Test Page</h1>
        <p> Hello! My name is {{name}}</p>
        {% if gender=='1' %}
                <h3>Male</h3>
        {% elif  gender=='0' %}
                <h3>Female</h3>
        {% endif %}
        <h3>Drinks</h3>
        <ul>
                {% for item in drinks %}
                <li>{{item}}</li>
                {% endfor %}
        </ul>
</body>
</html>
```



Flask Jinja Template Test Page

Hello! My name is Lee

Male

Drinks

- cofee
- milk
- tea
- beer

Flask

Session

- session.secret_key

```
from flask import Flask, session, redirect, url_for, escape, request
app = Flask(__name__)
@app.route('/')
def index():
   if 'username' in session:
      return 'Logged in as %s' % escape(session['username'])
   return 'You are not logged in'

@app.route('/login', methods=['GET', 'POST'])
def login():
   if request.method == 'POST':
      session['username'] = request.form['username']
      return redirect(url_for('index'))
   return '''
         <form action="" method="post">
                <p><input type=text name=username>
                <p><input type=submit value=Login></form>
     '''
```

```
@app.route('/logout')
def logout():
    session.pop('username', None)
   return redirect(url_for('index'))

app.secret_key = 'A0Zr98j/3yX R~XHH!jmN]LWX/,?RT'
app.run()
```

세부목차

WebSocket

❖ Flask-Socket.IO

- https://flask-socketio.readthedocs.io/
- Flask와 함께 사용할 수 있는 Socket.io
  - http://socket.io/
- 설치
  - pip install flask-socketio
  - pip install gevent or pip install eventlet
- 주요 코드
  - from flask_socketio import SocketIO, send
  - socketio = SocketIO(app)
  - socketio.run(app)
  - @socketio.on('message')
    def handle_message(msg) :
  - send('message', broadcast=True)
  - socket.send('message') : 외부에서 사용
  - emit('event name', 'message', broadcast=True)

Websocket

❖ websocket.py

```
from flask import Flask, redirect, request
from flask_socketio import SocketIO, send

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret'
socketio = SocketIO(app)

@app.route('/')
def main():
   return redirect('/static/websocket.html')

@socketio.on('message')
def handle_message(msg):
   print 'recv:', msg
   send(msg, broadcast=True)
```

```
@app.route('/notify', methods=['GET'])
def msg():
   param = request.values['param']
   socketio.send("nofity : " + param)
   return 'ok'

if __name__ == '__main__':
  #  app.run()
   socketio.run(app)
```

Websocket

❖ static/websocket.html

```
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<style type="text/css">
#log{
        width : 500px; height : 400px;
        border: 1px solid #000;
        overflow: auto;
}
</style>
<script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.min.js"></script>
<script type="text/javascript" charset="utf-8">
window.onload = function(){
        var id = document.querySelector('#id');
        var btn_conn = document.querySelector('#btn_connect');
        var msg = document.querySelector('#msg');
        var btn = document.querySelector('#btn_send');
        var log = document.querySelector('#log');
        var btn_ajax = document.querySelector('#btn_ajax');
        var param = document.querySelector('#param');
```

Websocket

❖ static/websocket.html <계속>

```
      var socket = null;
      btn_conn.onclick = function(){
            socket = io.connect('http://' + document.domain + ':' + location.port);
        socket.on('connect', function() {
            console.log('ws connect.');
          socket.send(id.value + " login.");
        });
        socket.on('message', function(data){
            var p = document.createElement('p');
            p.textContent = data;
            log.appendChild(p)
        });
      }
btn.onclick = function(){
      socket.send(id.value + ":" + msg.value);
};
```

Websocket

❖ **static/websocket.html <계속>**

```
btn_ajax.onclick = function(){
        xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function(){
                if(xhr.readyState == 4){
                                console.log(xhr.responseText);
                }
        };
        xhr.open('GET', '/notify?param=' + param.value);
        xhr.send();
   };

}
</script>
</head>
<body>
```

Websocket

❖ static/websocket.html <계속>

```
<input id="id"/><button id="btn_connect">connect</button><br/>
<input id="msg" />
<button id="btn_send" >send</button>
<div id="log"></div>
<input id="param"/><button id="btn_ajax">Ajax</button>
</body>
</html>
```

세부목차

Web with GPIO

❖ static/websocket.html <계속>

```
<script type="text/javascript">
window.onload = function(){
        var btn_led_on = document.querySelector('#btn_led_on');
        var btn_led_off = document.querySelector('#btn_led_off');
        btn_led_on.onclick= function(){
                var url = '/operate/led?val=on';
                sendAjax(url);
        }
        btn_led_off.onclick= function(){
                var url = '/operate/led?val=off';
                sendAjax(url);
        }
        function sendAjax(url, fn){
                xhr =new XMLHttpRequest();
                xhr.onreadystatechange = function(){
                        if(xhr.readyState == 4){
                                if(fn){
                                        fn(xhr.responseText);
                                }
                        }
                }
                xhr.open('GET', url);
                xhr.send();
        }
```

```
<button id="btn_led_on">Led
On</button>
<button id="btn_led_off">Led
Off</button><br/>
```

Web with GPIO

❖ static/websocket.html <계속>

```
from flask import Flask, request, redirect
import RPi.GPIO as GPIO

app = Flask(__name__)
GPIO.setmode(GPIO.BCM)
pin_led = 23

@app.route('/')
def main():
   return redirect('/static/gpio.html')
@app.route('/operate/<cmd>')
def op(cmd):
   val = request.values['val']
   if cmd ==  "led":
      val = request.values['val']
      print '/operate/', cmd, val
      if val == 'on':
         GPIO.output(pin_led, True)
         print pin_led, 'on'
      elif val == 'off':
         GPIO.output(pin_led, False)
      return 'OK'
```

```
if __name__ == '__main__':
         app.run(host='0.0.0.0')
```