Main Program

stepperMoter.py

메인 프로그램

```
from flask import Flask, render_template, send_from_directory, Response, send_file,
request, redirect, url_for
from flask_socketio import SocketIO
import argparse, logging, logging.config, conf
import os
from gpioControl import GpioControl
from time import sleep
from usbwebcamera import UsbWebCamera
from power import PowerStatus
from listenSpeech import ListenSpeech
from capture import capture_and_save
from postgresql import KKMsgDao
import json
from flask_sqlalchemy import SQLAlchemy
from models import db
from models import User
from flask import session
from flask_wtf.csrf import CSRFProtect #csrf
from form import RegisterForm, LoginForm



app = Flask(__name__)
app.config['SECRET_KEY'] = '17xxxx21q'

socketio = SocketIO(app)
archive_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'archive')

logging.config.dictConfig(conf.dictConfig)
logger = logging.getLogger(__name__)

gpio = GpioControl()
power = PowerStatus()
listenspeech = ListenSpeech()

usbcamera = UsbWebCamera(video_source=0)
usbcamera.start()

postgresql = KKMsgDao()
```

```
@app.after_request
def add_header(r):
    """
    Add headers to both force latest IE rendering or Chrome Frame,
    and also to cache the rendered page for 10 minutes
    """
    r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    r.headers["Pragma"] = "no-cache"
    r.headers["Expires"] = "0"
    r.headers["Cache-Control"] = "public, max-age=0"
    return r

@app.route('/', methods=['GET', 'POST'])
def mainpage():
    userid = session.get('userid',None)
    form = LoginForm() #로그인폼
    if userid is None:
        return render_template('login.html', form=form)
    else:
        msgList = postgresql.getMsg()
        return render_template("index.html", msgList=msgList, userid=userid)


@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm() #로그인폼
    try:
        if form.validate_on_submit(): #유효성 검사
            logger.debug('{}가 로그인 했습니다'.format(form.data.get('userid')))
            session['userid']=form.data.get('userid') #form에서 가져온 userid를 세션에 저장
            msgList = postgresql.getMsg()
            return render_template("index.html", msgList=msgList,
userid=form.data.get('userid')) #성공하면 main.html로
    except ValueError as ve:
        logger.debug('로그인 실패 ' + str(ve))
        return render_template('login.html', form=form, message=str(ve))

@app.route('/register', methods=['GET','POST'])  #겟, 포스트 메소드 둘다 사용
def register():   #get 요청 단순히 페이지 표시 post요청 회원가입-등록을 눌렀을때 정보 가져오는것
    form = RegisterForm()
    if form.validate_on_submit(): # POST검사의 유효성검사가 정상적으로 되었는지 확인할 수 있다. 입력 안 한것들이 있는지 확인됨.
    #비밀번호 = 비밀번호 확인 -> EqulaTo

        user = User()  #models.py에 있는 user
        user.userid = form.data.get('userid')
        user.username = form.data.get('username')
        user.email = form.data.get('email')
        user.password = form.data.get('password')
```

```
        print(user.userid,user.password)   #회원가입 요청시 콘솔창에 ID만 출력 (확인용, 딱히 필요없음)
        db.session.add(user)   # id, name 변수에 넣은 회원정보 DB에 저장
        db.session.commit()   #커밋

        return "가입 완료" #post요청일시는 '/'주소로 이동. (회원가입 완료시 화면이동)

    return render_template('register.html', form=form)


@app.route("/index.html")
def index():
    logger.debug("Requested /")

    userid = session.get('userid',None)
    if userid is None:
        return render_template('login.html', form=form)

    msgList = postgresql.getMsg()
    return render_template("index.html", msgList=msgList)

@app.route("/video_usb_feed")
def video_usb_feed():
    userid = session.get('userid',None)
    if userid is None:
        return Response(None,
            mimetype="multipart/x-mixed-replace; boundary=frame")
    return Response(genusb(usbcamera),
        mimetype="multipart/x-mixed-replace; boundary=frame")

def genusb(usbcamera):
    logger.debug("Starting USB stream")
    while True:
        frame = usbcamera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/png\r\n\r\n' + frame + b'\r\n')

@app.route("/temperature")
def temperature():
    content = os.popen("vcgencmd measure_temp").readline()
    content = content.replace("temp=", "")
    powerstatus = power.getPowerStatus()
    return Response(content+"["+powerstatus+"]", mimetype='text/xml')


''' ##### Control Camera Section ##### '''
@app.route("/msg", methods=['GET', 'POST'])
def msg():
     if request.method == 'POST':
        message = request.form['textinput']
```

```
        if not message :
             reString = "No received message"
        else :
             reString = listenspeech.speech(message)
        print("Get Message = " + reString)
     return reString

@app.route("/up", methods=['GET', 'POST'])
def up():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveUp(int(moveOrder))
        print("Move Up ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/down", methods=['GET', 'POST'])
def down():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveDown(int(moveOrder))
        print("Move Down ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/left", methods=['GET', 'POST'])
def left():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveLeft(int(moveOrder))
        print("Move Left ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/right", methods=['GET', 'POST'])
def right():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.moveRight(int(moveOrder))
        print("Move Right ..." + moveOrder)
    return Response(reString, mimetype='text/html')
```

```
''' ##### Achive File Section ##### '''
@app.route("/usbcapture")
def captureusb():
    logger.debug("Requested USBCAM capture")
    im = usbcamera.get_captureFrame()
    capture_and_save(im)
    return render_template("send_to_init.html")

@app.route('/archive')
def archive():
    return render_template('archive.html')

def get_type(filename):
    name, extension = os.path.splitext(filename)
    return 'video' if extension == '.mp4' else 'audio' if extension == '.wav' else
'audio' if extension == '.mp3' else 'photo'

@app.route('/archive/<string:filename>')
def archive_item(filename):
    name, extension = os.path.splitext(filename)
    type = get_type(filename)
    return render_template('record.html', filename=filename, type=type)

@app.route('/archive/delete/<string:filename>')
def archive_delete(filename):
    os.remove(archive_path + "/" + filename)
    return redirect(url_for('archive'))

@app.route('/archive/play/<string:filename>')
def archive_play(filename):
    return send_file('archive/' + filename)

def get_records():
    records = []

    for filename in sorted(os.listdir(archive_path), reverse=True):
        if not filename.startswith('.'):
            type = get_type(filename)
            size = byte_to_mb(os.path.getsize(archive_path + "/" + filename))
            record = {"filename": filename, 'size': size, 'type': type}
            records.append(record)

    return records

def byte_to_mb(byte):
    mb = "{:.2f}".format(byte / 1024 / 1024)
    return str(mb) + " MB"

app.jinja_env.globals.update(get_records=get_records)
```

```
''' ##### Achive File Section ##### '''
def messageReceived(methods=['GET', 'POST']):
    logger.debug('message was received!!!')

@socketio.on('my event')
def handle_my_custom_event(jsonMsg, methods=['GET', 'POST']):
    jsonObj = json.loads(str(jsonMsg).replace("'", "\""))
    #logger.debug('User name =' + str(jsonObj.get("user_name")))
    username = jsonObj.get("user_name")
    message = jsonObj.get("message")
    clientIP = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
    if username and message and clientIP:
        #logger.debug(">>> INSERT")
        postgresql.insMsg(username, message, clientIP)
        socketio.emit('my response', jsonMsg, callback=messageReceived)
    else:
        logger.debug("NONE")

''' ##### Firing File Section ##### '''
@app.route('/firing', methods=['GET', 'POST'])
def kkr_fire():
    gpio.fire()
    logger.debug('message KK Fired !!!')
    return Response('Fired !', mimetype='text/html')


@app.route("/favorit.ico")
def favorit_ico():
    logger.debug("Requested favorit.ico image")
    filename = "favorit.ico"
    return send_file(filename)


if __name__=="__main__":

  #데이터베이스---------
    basedir = os.path.abspath(os.path.dirname(__file__)) #현재 파일이 있는 디렉토리 절대 경로
    dbfile = os.path.join(basedir, 'db.sqlite') #데이터베이스 파일을 만든다

    app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///' + dbfile
    app.config['SQLALCHEMY_COMMIT_ON_TEARDOWN'] = True #사용자에게 정보 전달완료하면 teadown.
그 때마다 커밋=DB반영
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False #추가 메모리를 사용하므로 꺼둔다

#    db = SQLAlchemy() #SQLAlchemy를 사용해 데이터베이스 저장
    db.init_app(app) #app설정값 초기화
    db.app = app #Models.py에서 db를 가져와서 db.app에 app을 명시적으로 넣는다
```

```
    with app.app_context():
        db.create_all() #DB생성
    logger.debug("Starting VisionK server")
    socketio.run(app, log_output=True, host='0.0.0.0', port=8081, debug=True,
use_reloader=False)
```

Revision #3
★Created 2023-06-10 12:50:39 UTC by Hyeon Su Ryu
Updated 2024-03-17 11:51:27 UTC by Hyeon Su Ryu