概述

这是一个基于 Flask 框架的smarthomehub的 Web API 服务。该系统允许用户通过 HTTP 请求来管理智能家居设备，包括添加设备、删除设备、列出设备、执行设备命令、显示设备状态以及计算总能耗的功能。

环境要求

Python 3.10 版本、OOP_work.py

代码结构与模块说明

导入模块

OOP_work（Smarthomehub的核心功能），flask（Flask 框架），os（文件操作），json（JSON数据处理）

初始化

python

```
app = Flask(__name__)#Flask实例
hub = SmartHomeHub()#SmarthomeHub实例
data = 'data.json' #当前目录存储文件地址
```

API实现

/add #添加设备

python

```
@app.route('/add', methods=['POST'])
def add_device():
    try:
        devices = request.get_json()#接受JSON文件并读取
        report = ""#初始化返回字段
        if not devices:
            return 'there is not json file 400'
        with open(data, "r") as fs:#读取本地文件并准备修改
            old_date = json.load(fs)
        for device in devices["hub"]:
#将新设备加载进hub和本地数据中，识别JSON文件的”hub”键。
            report += hub.controller.add_device(device) + "\n"
            old_date["hub"].append(device)
        with open(data, "w") as f:#保存文件
            json.dump(old_date, f, indent=4)
        return report
#报错：1.没有文件 2.文件格式不为JSON 3.其他错误
    except FileNotFoundError:
        return "error: not find file"
    except json.JSONDecodeError:
        return "error: unable to parse the file as valid JSON"
```

```
    except Exception as e:
        return str(e) + " 400"
```

/total_amount #获取设备数量

python

```
@app.route('/total_amount', methods=['GET'])
def total_amount():
    return
str(hub.controller.total_amount())#调用hub.controller中的方法，返回device的数量
/remove #移除设备
@app.route('/remove', methods=['POST'])
def remove_device():
    try:
        devices = request.get_json() #接受读取的JSON文件
        if str(devices['password']) != '114514': #注意：弱密码及其保存结构
            return 'fail to connect with correct password'
        with open(data, "r") as fs: #读取本地文件
            old_date = json.load(fs)
        ids = devices["id"]
        success = ""
        error = ""
        for _id in ids:
#从hub中删除device，从本地文件传入的数据中删除device
            if _id in hub.controller.devices:
                hub.controller.remove_device(_id)
                #识别”hub”字段
                for num, device in enumerate(old_date["hub"]):
                    if _id in device: #顺序检索
                        del old_date["hub"][num]
                        break #停止循环
                success += str(_id) + " "
            else:
                error += f"{_id} is not exist\n"
        with open(data, "w") as f: #保存
            json.dump(old_date, f)
#返回删除成功和失败对应的device
        return success + "is deleted\n" + error
#报错：1.没有文件 2.文件格式不为JSON 3.其他错误
    except FileNotFoundError:
        return "error: not find file"
    except json.JSONDecodeError:
        return "error: unable to parse the file as valid JSON"
    except Exception as e:
        return str(e) + ' 400'
```

/list #获取设备完整数据

python

```
@app.route('/list', methods=['GET'])
def list_devices():
    return hub.controller.list_devices() #从hub.controller中返回device的字符串
```

/execute/<string:device_id> #改变设备的开关状态

python

```
@app.route('/execute/<string:device_id>', methods=['GET'])
def execute_command(device_id):
    try: #尝试获取指令
        command = request.args.get("command")
        #将id和指令传输到execute中，并返回操作信息
        return hub.controller.execute_command(device_id, command)
    except Exception as e:
        return str(e) + "400"
```

display_status #获取设备状态

python

```
@app.route('/display_status', methods=['GET'])
def display_status():
    return hub.display_status() #从hub中返回设备名字和状态
```

/total_energy_usage 返回所有设备的能耗总和

python

```
def total_energy_usage():
return str(hub.total_energy_usage())    #返回hub的方法中返回的数字转化后的字符
```

/schedule 接受并加载预设时间

python

```
@app.route('/schedule',methods=['POST'])
def schedule():
    total=request.get_json()
    for t,di in total.items():
        for _id,command in di.items():
            hub.schedule_task(device_id=_id,command=command,time=t)
    return 'finish covering'
```

auto_switch 函数，用于自动检测并自动改变设备

python

```
def auto_switch():
    try:
        with open('time.json', 'r') as f:
            data_time = json.load(f)
        now=time.strftime('%H:%M',time.localtime(time.time()))
        if isinstance(data_time,dict):
            for dt in data_time.keys():
                if now==dt:
                    for _id,command in data_time[now].items():
                        hub.controller.execute_command(_id,command)
        time.sleep(60)
    except Exception as e:
        return str(e)
```

主程序

#加载或创建JSON文件作为本地数据，并加载其中的device信息。运行Flask应用，开启调试。子进程开启时间检测和自动开关设备。

python

```
if __name__=='__main__':
    #copying
    shutil.copy(data,"./old data/")
    #loading
    if not os.path.exists('./data.json'):
        with open(data,'w') as fs:
            json.dump({"hub":[]},fs,indent=4)
    else:
        with open(data,'r') as fs:
            datas=json.load(fs)
            for d in datas["hub"]:
                hub.controller.add_device(d)
    #run
    _auto_switch=multiprocessing.Process(target=auto_switch)
    app.run(debug=True)
```

错误处理

在每个 API 路由中，都有相应的异常处理机制，捕获 `FileNotFoundError`、`json.JSONDecodeError` 等异常，并返回相应的错误信息。

对于其他未预料到的异常，返回异常信息和状态码 `400`。

注意事项

密码 `114514` 是硬编码的，建议在实际应用中使用更安全的密码管理方式。

在生产环境中，建议关闭 Flask 的调试模式。