

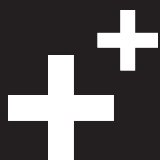
Python开发

NSD DEVOPS

DAY02

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	邮件和JSON
	10:30 ~ 11:20	
	11:30 ~ 12:00	requests模块
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	zabbix编程
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



邮件和JSON

邮件和JSON

邮件编程

SMTP概述

SMTP对象

设置邮件

sendmail方法

SMTP认证

JSON

JSON概述

JSON结构

dumps方法

loads方法

天气预报查询

邮件编程



SMTP概述

- SMTP (Simple Mail Transfer Protocol) 即简单邮件传输协议，使用TCP协议25端口
- 它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式
- python的smtpplib提供了一种很方便的途径发送电子邮件。它对smtp协议进行了简单的封装



SMTP对象

- Python发送邮件，第一步是创建SMTP对象

```
import smtplib
```

```
smtp_obj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

- 创建SMTP对象也可以不给定参数，之后再通过对象的其他方法进行绑定



设置邮件

- 标准邮件需要三个头部信息
 - From : 发件人
 - To : 收件人
 - Subject : 主题

```
from email.mime.text import MIMEText
from email.header import Header
message = MIMEText('Python 邮件发送测试...', 'plain', 'utf-8')
message['From'] = Header("zzg", 'utf-8') # 发送者
message['To'] = Header("root", 'utf-8')   # 接收者
subject = 'Python SMTP 邮件测试'
message['Subject'] = Header(subject, 'utf-8')
```



sendmail方法

- Python SMTP 对象使用 sendmail 方法发送邮件

`SMTP.sendmail(from_addr, to_addrs, msg[, mail_options, rcpt_options])`

- sendmail方法三个必须的参数有：
 - 收件人
 - 发件人
 - 消息主体msg是一个字符串，表示邮件



sendmail方法（续1）

- 将准备好的邮件发送

```
sender = 'from@tarena.com'  
receivers = ['root@localhost']  
smtp_obj = smtplib.SMTP('localhost')  
smtp_obj.sendmail(sender, receivers, message.as_string())
```



案例1：通过本机发送邮件

1. 创建bob和alice帐户
2. 编写发送邮件程序，发件人为root，收件人是本机的bob和alice帐户



SMTP认证

- 如果本机没有SMTP功能，也可以使用第三方的邮件服务器
- 第三方邮件服务器往往需要认证

```
mail_host="mail.tedu.cn"  
mail_user="zzg"  
mail_pass="zzg_pass"  
smtp_obj = smtplib.SMTP()  
smtp_obj.connect(mail_host, 25) # 25 为 SMTP 端口号  
smtp.starttls() # 如果使用证书，打开此注释  
smtp_obj.login(mail_user,mail_pass)  
smtp_obj.sendmail(sender, receivers, message.as_string())
```



案例2：通过互联网服务器发送邮件

1. 通过自己互联网注册的邮箱，为其他同学互联网邮箱发邮件



JSON



JSON概述

- JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式
- 易于人阅读和编写，同时也易于机器解析和生成
- 基于JavaScript Programming Language
- JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）
- 这些特性使JSON成为理想的数据交换语言



JSON结构

- JSON主要有两种结构
 - “键/值” 对的集合：python中主要对应成字典
 - 值的有序列表：在大部分语言中，它被理解为数组

Python	JSON
dict	object
list, tuple	array
str	string
int, float	number
True	true
False	false
None	null



dumps方法

- 对编码后的json对象进行decode解码，得到原始数据，需要使用的json.loads()函数

```
>>> import json
>>> number = json.dumps(100)
>>> json.loads(number)
100
```



loads方法

- 使用简单的json.dumps方法对简单数据类型进行编码

```
>>> import json
>>> json.dumps(100)
'100'
>>> json.dumps([1, 2, 3])
'[1, 2, 3]'
>>> json.dumps({'name': 'zzg'})
'{"name": "zzg"}'
```



天气预报查询

- 搜索 “中国天气网 城市代码查询” ，查找城市代码
- 城市天气情况接口
 - 实况天气获取：<http://www.weather.com.cn/data/sk/城市代码.html>
 - 城市信息获取：
<http://www.weather.com.cn/data/cityinfo/城市代码.html>
 - 详细指数获取：<http://www.weather.com.cn/data/zs/城市代码.html>



案例3：天气预报查询

1. 运行程序时，屏幕将出现你所在城市各区县名字
2. 用户指定查询某区县，屏幕上将出现该区县的当前的气温、湿度、风向、风速等



requests模块

requests模块

```
graph LR; A[requests模块] --> B[requests基础]; A --> C[requests应用]; B --> B1[requests简介]; B --> B2[requests特性]; B --> B3[GET和POST]; B --> B4[其他方法]; B --> B5[请求参数]; B --> B6[设定头部]; B --> B7[发送请求数据]; C --> C1[响应内容]; C --> C2[其他响应内容格式]; C --> C3[响应头]
```

requests基础

requests简介

requests特性

GET和POST

其他方法

请求参数

设定头部

发送请求数据

requests应用

响应内容

其他响应内容格式

响应头

requests基础

requests简介

- Requests是用Python语言编写的、优雅而简单的HTTP库
- Requests内部采用urllib3
- Requests使用起来肯定会比urllib3更简单便捷
- Requests需要单独安装



requests特性

- 支持keep-alive的连接池
- 支持通用的域名以及URL地址
- 支持使用cookie
- 支持使用类似浏览器的SSL验证
- 文件上传、下载



GET和POST

- 通过requests发送一个GET请求，需要在URL里请求的参数可通过params传递

```
r = requests.get(url="", params={}, headers={}, cookies={})
```

- 与GET不同的是，POST请求新增了一个可选参数data，需要通过POST请求传递的body里的数据可以通过data传递

```
r = requests.post(url="", data={}, params={}, file={}, headers={}, cookies={})
```



其他方法

- 其他 HTTP 请求类型：PUT，DELETE，HEAD 以及 OPTIONS使用起来一样简单

```
>>> r = requests.put('http://httpbin.org/put', data = {'key':'value'})  
>>> r = requests.delete('http://httpbin.org/delete')  
>>> r = requests.head('http://httpbin.org/get')  
>>> r = requests.options('http://httpbin.org/get')
```



请求参数

- 当访问一个URL时，我们经常需要发送一些查询的字段作为过滤信息，例如：`httpbin.com/get?key=val`，这里的`key=val`就是限定返回条件的参数键值对
- 当利用python的requests去发送一个需要包含这些参数键值对时，可以将它们传给params

```
payload = {'key1':'value1', 'key2':'value2'}  
r = requests.get('http://httpbin.com/get', params = payload)
```



设定头部

- 用户也可以自己设定请求头

```
url = 'https://api.github.com/some/endpoint'  
headers = {'Accept': 'application/json'}  
r = requests.get(url, headers = headers)
```



发送请求数据

- 有时候，用户需要将一些数据放在请求的body内；这时候，就需要对data传参了（仅POST, DELETE, PUT等方法有该参数，GET没有，因为GET请求没有body）

```
payload = {'key1': 'value1', 'key2': 'value2'}
```

```
r = requests.post('http://httpbin.org/post', data = payload)
```



requests应用



响应内容

- 读取服务器响应的内容

```
>>> r = requests.get('http://www.baidu.com')
>>> r.text
```
- 请求发出后，Requests 会基于 HTTP 头部对响应的编码作出有根据的推测
- 可以找出 Requests 使用了什么编码，并且能够使用 r.encoding 属性来改变它

```
>>> r.encoding
'ISO-8859-1'
>>> r.encoding='utf8'
>>> r.text
```



其他响应内容格式

- 也可以用字节的方式访问请求响应体，尤其是非文本请求（如图片）

```
>>> r.content
```

- Requests 中还有一个内置的 JSON 解码器，助你处理 JSON 数据

```
>>> r.json()
```



响应头

- 可以查看以字典形式展示的服务器响应头

```
>>> r.headers
{
  'content-encoding': 'gzip',
  'transfer-encoding': 'chunked',
  'connection': 'close',
  'server': 'nginx/1.0.4',
  'x-runtime': '148ms',
  'etag': '"e1ca502697e5c9317743dc078f67693f"',
  'content-type': 'application/json'
}
```



案例4：钉钉机器人

1. 创建一个群聊钉钉机器人
2. 编写代码，通过python脚本实现钉钉机器人在群中发送消息
3. 使用json和requests模块



zabbix编程

zabbix编程

Zabbix api简介

部署zabbix

Zabbix api概述

JSON-RPC

API结构

执行请求

使用API

方法参考

工作流程

获取令牌

令牌响应信息

检索主机

获取主机组

创建主机

Zabbix api简介



部署zabbix

- Zabbix是一个基于WEB界面的提供分布式系统监视以及网络监视功能的企业级的开源解决方案
- 能监视各种网络参数，保证服务器系统的安全运营；并提供灵活的通知机制以让系统管理员快速定位/解决存在的各种问题
- 部署方式参见云计算监控课程，不再赘述



Zabbix api概述

- Zabbix API允许你以编程方式检索和修改Zabbix的配置，并提供对历史数据的访问。它广泛用于：
 - 创建新的应用程序以使用Zabbix
 - 将Zabbix与第三方软件集成
 - 自动执行常规任务



JSON-RPC

- Zabbix API是基于Web的API，作为Web前端的一部分提供。它使用JSON-RPC 2.0协议，这意味着两件事：
 - 该API包含一组独立的方法
 - 客户端和API之间的请求和响应使用JSON格式进行编码



API结构

- Zabbix API包含许多方法，这些方法都名义上分组为单组的API
- 每个方法执行一个特定任务。例如，方法 `host.create` 隶属于 `host` 这个API，用于创建新主机
- 历史上，API有时被称为“类”
- 大多数API至少包含四种方法：`get`，`create`，`update` 和 `delete`，分别是检索，创建，更新和删除数据。但是某些API提供一套完全不同的一组方法。



执行请求

- 设置前端后，你就可以使用远程HTTP请求来调用API。为此，需要向 `api_jsonrpc.php` 位于前端目录中的文件发送HTTP POST请求。例如，如果你的Zabbix前端安装在 `http://company.com/zabbix`，那么用HTTP请求来调用 `apiinfo.version` 方法就如下面这样：

```
POST http://company.com/zabbix/api_jsonrpc.php HTTP/1.1
Content-Type: application/json-rpc
```

```
{"jsonrpc":"2.0","method":"apiinfo.version","id":1,"auth":null,"params":{
}}
```



执行请求（续1）

- 请求的 Content-Type 头部必须设置为以下值之一：
 - application/json-rpc
 - application/json
 - application/jsonrequest



使用API

- 通过zabbix提供的API接口，就可以使用python与其交互了

```
url = 'http://zabbix_server/api_jsonrpc.php'  
headers = {'Content-Type': 'application/json-rpc'}
```

```
data = {  
    'jsonrpc': '2.0',  
    'method': 'apiinfo.version',  
    'id': 1,  
    'auth': None,  
    'params': {},  
}  
r = requests.post(url, headers=headers, data=json.dumps(data))  
print(r.json())
```



案例5：获取zabbix版本信息

- 安装zabbix服务器
- 获取zabbix api的url
- 编写python程序，访问zabbix api，取得zabbix版本号



方法参考

工作流程

- 在访问大多数Zabbix中的任何数据之前，需要登录并获取身份验证令牌
- 取得令牌后，访问其他数据只要出示该令牌即可，不需要再进行身份验证
- 通过zabbix api提供的各种方法实现数据的检索、项目的创建等



获取令牌

- 使用 user.login 方法登录并获取身份验证令牌

```
{  
  "jsonrpc": "2.0",  
  "method": "user.login",  
  "params": {  
    "user": "Admin",  
    "password": "zabbix"  
  },  
  "id": 1,  
  "auth": None  
}
```



获取令牌（续1）

- jsonrpc - API使用的JSON-RPC协议的版本; Zabbix API实现JSON-RPC版本2.0
- method - 调用的API方法
- params - 将被传递给API方法的参数
- id - 请求的任意标识符
- auth - 用户认证令牌; 因为我们还没有一个, 它的设置None



令牌响应信息

- 如果你正确提供了凭据，API返回的响应将包含用户身份验证令牌

```
{
  "jsonrpc": "2.0",
  "result": "0424bd59b807674191e7d77572075f33",
  "id": 1
}
```

- 响应对象又包含以下属性：
 - jsonrpc - JSON-RPC协议的版本
 - result - 方法返回的数据
 - id - 相应请求的标识符



案例6：获取令牌

- 编写get_token函数
- 该函数接受zabbix服务器url、用户名和密码作为参数
- 函数返回值为用户令牌token



检索主机

- 有一个有效的用户身份验证令牌，可以用来访问 Zabbix 中的数据
- 使用 host.get 方法检索所有已配置主机的ID，主机名和接口。auth 属性设置为获得的身份验证令牌

```
{
  "jsonrpc": "2.0",
  "method": "host.get",
  "params": {
    "output": [ "hostid", "host"],
    "selectInterfaces": [ "interfaceid", "ip" ]
  },
  "id": 2,
  "auth": "0424bd59b807674191e7d77572075f33"
}
```



获取主机组

- 获取主机组的方法与检索主机一样，只要修改请求数据即可

```
data = {  
    "jsonrpc": "2.0",  
    "method": "hostgroup.get",  
    "params": { "output": "extend", },  
    "id": 2,  
    "auth": "fdeb85b4311f2fa48bb4259b6071ddd2"  
}
```



创建主机

- 创建主机操作与获取信息操作完全一样，只是传递的请求参数不一样而已

```
data = {
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            { "type": 1, "main": 1, "useip": 1, "ip": "192.168.4.1", "dns": "",
"port": "10050" }
        ],
        "groups": [ { "groupid": "2" } ],
    },
    "auth": "5633be08efe9a3d5369cb868bc1dc61d",
    "id": 1
}
```



案例7：创建主机

- 主机192.168.4.10已安装zabbix_agent
- 将该主机添加到zabbix监控的主机中
- 主机属于Linux Servers组



总结和答疑
