搭建开发环境

1、什么是虚拟环境?

虚拟环境是一个包含特定版本依赖包的开发的环境。

virtualenv 虚拟环境的管理工具，可以创建多个互不干扰的开发环境，库将安装到各自的目录下，不会和其他环境共享。

由于 virtualenv 用起来有点麻烦，virtualenvwrapper 对它进行了封装，让它更好用，我们使用 wrapper 提供的命令，但是实际工作都是 virtualenv 做的。

2、虚拟环境安装

Window 10 平台

pip 升级

python -m pip install --upgrade pip

Virtualenv 安装

pip install virtualenv

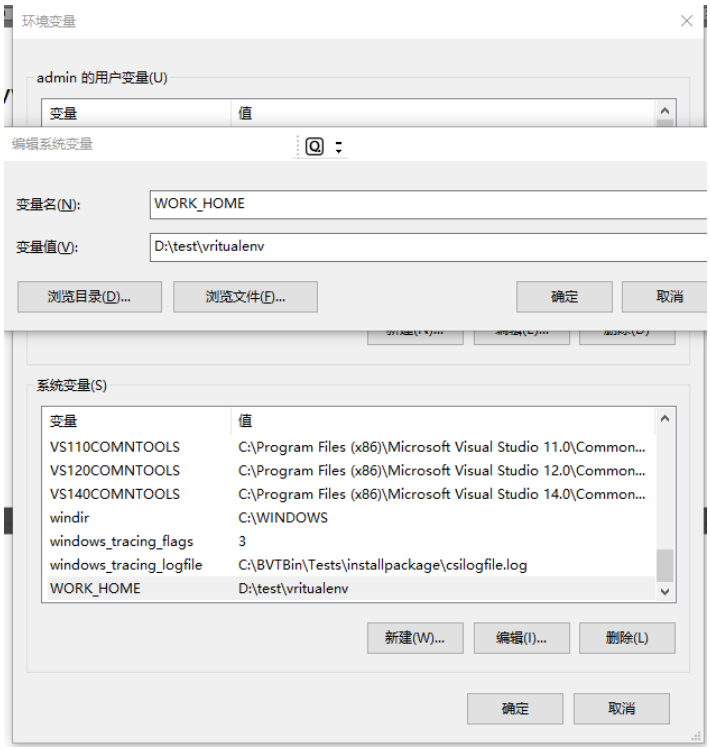
virtualenvwrapper 安装

pip install virtualenvwrapper-win

设置 WORK_HOME 环境变量

默认路径：C:\Users\admin\Envs

WORKON_HOME = D:\test\vritualenv



Ubuntu 平台

pip 安装

sudo apt install python3-pip

pip 升级

sudo python3 -m pip install --upgrade pip

Virtualenv 安装

sudo python3 -m pip install virtualenv

virtualenvwrapper 安装

sudo python3 -m pip install virtualenvwrapper

打开~/.bashrc 文件:

sudo gedit ~/.bashrc

在结尾添加:

export WORKON_HOME=$HOME/.virtualenvs

export PROJECT_HOME=$HOME/workspace

source /usr/local/bin/virtualenvwrapper.sh

然后执行:

source ~/.bashrc

将设置在文件中的配置信息马上生效,而不需要经过重启。

所有的虚拟环境，都位于/home/.virtualenvs 目录下

报错：/usr/bin/python: No module named virtualenvwrapper

原因：Ubuntu 安装了 2.7 和 3.x 两个版本的 python,在安装时使用的是 sudo pip3 install virtualenvwrapper

在运行的时候默认使用的是 python2.x,但在 python2.x 中不存在对应的模块。

解决办法：增加此环境变量:

VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3

注意

在 ubuntu 下以点开头命名的文件和文件夹是隐藏的，如果需要修改它们，如何看见

进入自己主目录，按 ctrl+h.就能看见以点号开头的隐藏文件

3、virtualenvwrapper 操作

- 创建：mkvirtualenv [虚拟环境名称]
- 删除：rmvirtualenv [虚拟环境名称]
- 进入：workon [虚拟环境名称]
- 退出：deactivate

Requests 库

简介

Requests Python 编写，基于 urllib，自称 HTTP for Humans（让 HTTP 服务人类）

特性：支持 HTTP 连接保持和连接池，支持使用 cookie 保持会话，支持文件上传，支持自动确定响应内容的编码，支持国际化的 URL 和 POST 数据自动编码。

使用更简洁方便，比 urllib 更加 Pythoner

开源地址：https://github.com/kennethreitz/requests

中文文档 API： http://docs.python-requests.org/zh_CN/latest/index.html

安装

pip install requests

基本 GET 请求

```
import requests
response = requests.get("http://www.baidu.com/")
# 也可以这么写
#response = requests.request("get", "http://www.baidu.com/")
# 查看响应内容，response.content 返回的字节流数据
print(response.content)
print(response.content.decode('utf8'))
# 查看响应内容，response.text 返回的是 Unicode 格式的数据
print(response.text)
# 查看完整 url 地址
print(response.url)
# 查看响应头部字符编码
print(response.encoding)
# 查看响应码
print(response.status_code)
```

添加 headers 和 查询参数

如果想添加 headers，可以传入 headers 参数来增加请求头中的 headers 信息。如果要将参数放在 url 中传递，可以利用 params 参数。

```
import requests

kw = {'wd':'长城'}
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36"}
# params 接收一个字典或者字符串的查询参数，字典类型自动转换为 url 编码，不需要 urlencode()
response = requests.get("http://www.baidu.com/s?", params = kw, headers = headers)
print(response.text)
print(response.encoding)
```

基本 POST 请求（data 参数）

1. 最基本的 GET 请求可以直接用 post 方法

```
response = requests.post("http://www.baidu.com/", data = data)
```

2. 传入 data 数据

对于 POST 请求来说，一般需要为它增加一些参数。可以利用 data 参数传参。

```
import requests


if __name__ == "__main__":
    #对应上图的 Request URL
    #Request_URL = 'http://fanyi.youdao.com/translate_o?smartresult=dict&smartresult=rule'
    Request_URL = 'http://fanyi.youdao.com/translate?smartresult=dict&smartresult=rule'
    #创建 Form_Data 字典，存储上图的 Form Data
    Form_Data = {}
    Form_Data['i'] = 'Tom'
    Form_Data['from'] = 'AUTO'
    Form_Data['to'] = 'AUTO'
    Form_Data['smartresult'] = 'dict'
    Form_Data['client'] = 'fanyideskweb'
    Form_Data['salt'] = '1526796477689'
    Form_Data['sign'] = 'd0a17aa2a8b0bb831769bd9ce27d28bd'
    Form_Data['doctype'] = 'json'
    Form_Data['version'] = '2.1'
    Form_Data['keyfrom'] = 'fanyi.web'
    Form_Data['action'] = 'FY_BY_REALTIME'
    Form_Data['typoResult'] = 'false'
    head = {}
    #写入 User Agent 信息
    head['User-Agent'] = 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/65.0.3325.181 Safari/537.36'

    response = requests.post(Request_URL, data=Form_Data,headers=head)
    print(response)
    print(response.text)
    translate_results = response.json()
    # #找到翻译结果
    translate_results = translate_results['translateResult'][0][0]['tgt']
    # #打印翻译信息
    print("翻译的结果是：%s" % translate_results)
```

代理（proxies 参数）

如果需要使用代理，你可以通过为任意请求方法提供 proxies 参数来配置单个请求：

```
import requests

import requests

# 根据协议类型，选择不同的代理
proxies = {
   "http": "http://27.184.124.29:8118",
}

response = requests.get("http://www.baidu.com", proxies = proxies)
print(response.text)
```

Cookies 和 Sission

Cookies

如果一个响应中包含了 cookie，那么我们可以利用 cookies 参数拿到：

```
import requests
response = requests.get("https://www.baidu.com/")
# 7. 返回 CookieJar 对象：
cookiejar = response.cookies
# 8. 将 CookieJar 转为字典：
cookiedict = requests.utils.dict_from_cookiejar(cookiejar)
print(cookiejar)
print(cookiedict)
```

运行结果：

```
<RequestsCookieJar[<Cookie BDORZ=27315 for .baidu.com/>]>
{'BDORZ': '27315'}
```

Sission

在 requests 里，session 对象是一个非常常用的对象，这个对象代表一次用户会话：从客户端浏览器连接服务器开始，到客户端浏览器与服务器断开。

会话能让我们在跨请求时候保持某些参数，比如在同一个 Session 实例发出的所有请求之

间保持 cookie 。

实现人人网登录

```
import requests

# 1. 创建 session 对象，可以保存 Cookie 值
ssion = requests.session()
# 2. 处理 headers
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36"}
# 3. 需要登录的用户名和密码
data = {"email":"mr_mao_hacker@163.com", "password":"alarmchime"}
# 4. 发送附带用户名和密码的请求，并获取登录后的 Cookie 值，保存在 ssion 里
ssion.post("http://www.renren.com/PLogin.do", data = data)
# 5. ssion 包含用户登录后的 Cookie 值，可以直接访问那些登录后才可以访问的页面
response = ssion.get("http://www.renren.com/410043129/profile")
# 6. 打印响应内容
print(response.text)
```

处理 HTTPS 请求 SSL 证书验证

Requests 也可以为 HTTPS 请求验证 SSL 证书：

要想检查某个主机的 SSL 证书，你可以使用 verify 参数（也可以不写）

```
import requests
response = requests.get("https://www.baidu.com/", verify=True)
# 也可以省略不写
# response = requests.get("https://www.baidu.com/")
print(response.text)
```

如果 SSL 证书验证不通过，或者不信任服务器的安全证书，则会报出 SSLError，比如 12306：

```
import requests
response = requests.get("https://www.12306.cn/mormhweb/")
print(response.text)
```

报错：

SSLError: HTTPSConnectionPool(host='www.12306.cn', port=443): Max retries exceeded with url: / (Caused by SSLError(CertificateError("hostname 'www.12306.cn'…