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

Virtualenv 安装

pip install　virtualenv

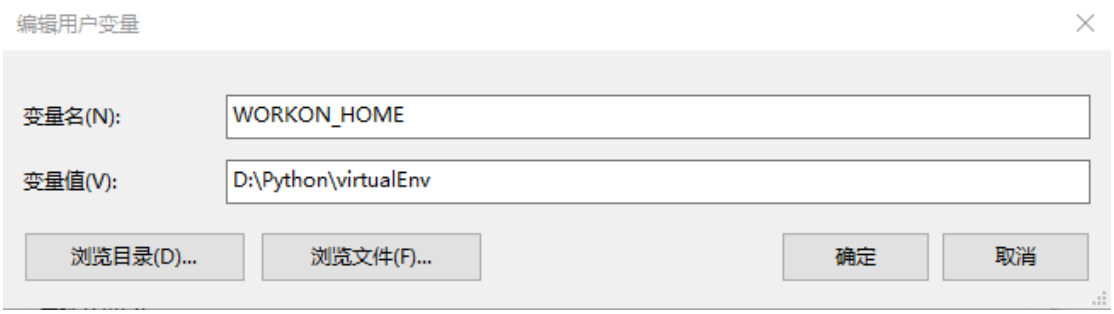
virtualenvwrapper 安装

pip install virtualenvwrapper-win

设置 WORK_HOME 环境变量

默认路径：C:\Users\admin\Envs

WORKON_HOME = D:\ Python\vritualenv



Ubuntu 平台

pip 安装(可选)

sudo apt install python3-pip

pip 升级(可选)

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：/usr/local/bin/virtualenvwrapper.sh 文件增加此环境变量：

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

爬虫简介

什么是爬虫?

是一种按照一定的规则，自动地抓取互联网信息的程序或者脚本。

所谓网页抓取，就是把 URL 地址中指定的网络资源从网络流中读取出来，保存到本地。 在 Python 中有很多库可以用来抓取网页

分类

通用爬虫（General Purpose Web Crawler)、聚焦爬虫（Focused Web Crawler)、增量式爬虫（Incremental Web Crawler)、深层爬虫（Deep Web Crawler）

通用网络爬虫

搜索引擎抓取系统（Baidu、Google、Yahoo 等）的重要组成部分。主要目的是将互联网上的网页下载到本地，形成一个互联网内容的镜像备份。

聚焦爬虫

是"面向特定主题需求"的一种网络爬虫程序，它与通用搜索引擎爬虫的区别在于： ***聚焦爬虫在实施网页抓取时会对内容进行处理筛选，尽量保证只抓取与需求相关的网页信息。***

增量式抓取

是指在具有一定量规模的网络页面集合的基础上，采用更新数据的方式选取已有集合中的过时网页进行抓取，以保证所抓取到的数据与真实网络数据足够接近。进行增量式抓取的前提

是，系统已经抓取了足够数量的网络页面，并具有这些页面被抓取的时间信息。

深度爬虫

针对起始 url 地址进行数据采集，在响应数据中进行数据筛选得到需要进行数据采集的下一波 url 地址，并将 url 地址添加到数据采集队列中进行二次爬取..以此类推，一致到所有页面的数据全部采集完成即可完成深度数据采集，这里的深度指的就是 url 地址的检索深度。

爬虫步骤

网页抓取，数据提取，数据存储

HTTP 协议

HTTP，HyperText Transfer Protocol，是互联网上应用最为广泛的一种网络协议。

是一个基于 TCP/IP 通信协议来传递数据，一个属于应用层的协议

浏览器作为 HTTP 客户端通过 URL 向 HTTP 服务端即 WEB 服务器发送所有请求。Web 服务器根据接收到的请求后，向客户端发送响应信息。

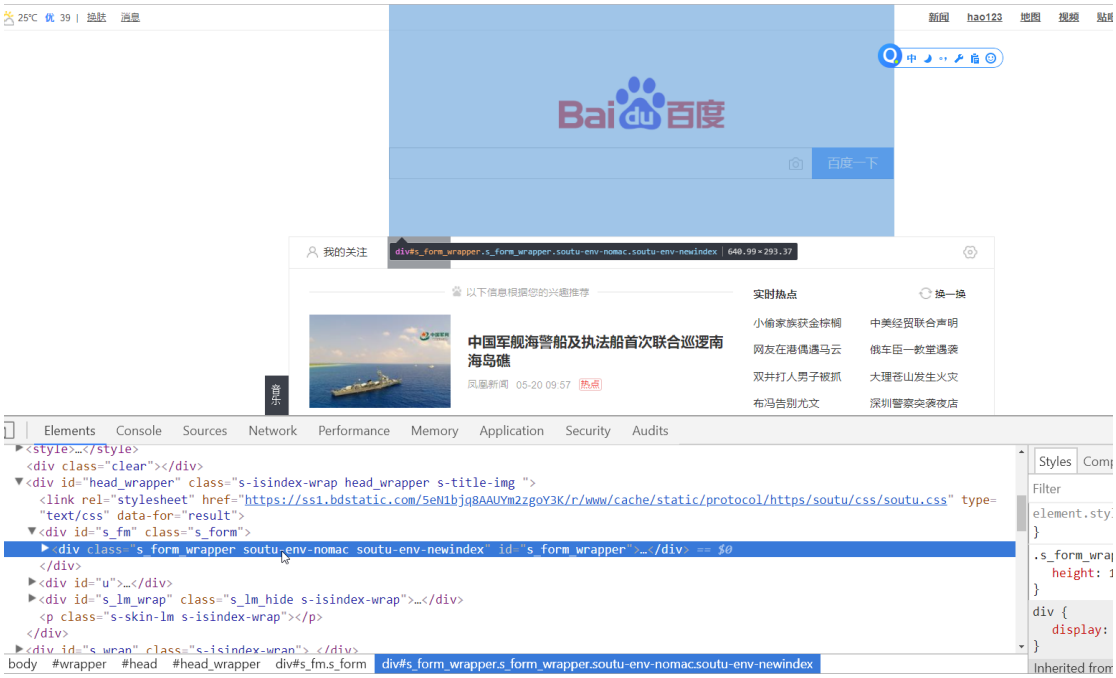
HTTPS（Hypertext Transfer Protocol over Secure Socket Layer）HTTP 的安全版，在 HTTP 下加入 SSL 层。

SSL（Secure Sockets Layer 安全套接层）主要用于 Web 的安全传输协议，在传输层对网络连接进行加密，保障在 Internet 上数据传输的安全。

- `HTTP` 的端口号为 `80`，
- `HTTPS` 的端口号为 `443`

审查元素

可以从浏览器中查看网页代码，例如谷歌浏览器，在任意界面单击右键选择检查，也就是审查元素(不是所有页面都可以审查元素的，例如起点中文网付费章节就不行.)，以百度界面为例，截图如下：



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

基本请求方法

```
requests.get(url)
requests.post(url, data = {'key':'value'})
```

返回响应对象

响应内容

请求的结果，返回 Response 对象。可以从这个对象中获取所有我们想要的信息。
response.content: 返回的字节流数据，可以 decode 解密
response.text： 返回的是 Unicode 格式的数据，Requests 会自动解码来自服务器的内容。大多数 unicode 字符集都能被无缝地解码。
response.url： 查看完整 url 地址
response.status_code： 查看响应码
response.encoding： 查看响应头部字符编码
response.json: JSON 格式响应内容
response.headers： 查看服务器返回的响应头部信息
response.request.headers： 查看发送到服务器的请求头

get 请求响应

```
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)
```

超时

可以告诉 requests 在经过以 **timeout 参数**设定的秒数时间之后停止等待响应。基本上所有的生产代码都应该使用这一参数。如果不使用，你的程序可能会永远失去响应

requests.get(' http://www.baidu.com/', timeout=0.001)

异常

遇到网络问题（如：DNS 查询失败、拒绝连接等）时，Requests 会抛出一个 ConnectionError 异常。

若请求超时，则抛出一个 Timeout 异常。

所有 Requests 显式抛出的异常都继承自 requests.exceptions.RequestException 。

```
import requests

try:
    requests.get(' http://www.baidu.com/', timeout=0.01)
except Exception as e:
    print(e)
```

添加 headers 和 查询参数

浏览器就是互联网世界上公认被允许的身份，服务器就是通过查看 Headers 中的 User

Agent 来判断是谁在访问。用不同的浏览器在发送请求的时候，会有不同的 User-Agent 头。中文名为用户代理，简称 UA

如果我们希望我们的爬虫程序更像一个真实用户，那我们第一步，就是需要伪装成公认的浏览器。

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",
  "https": "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'}

案例：cookie 模拟登录人人网

import requests
# 1. 构建一个已经登录过的用户的 headers 信息
headers = {
"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36",
重点：这个 Cookie 是保存了密码无需重复登录的用户的 Cookie，这个 Cookie 里记录了用户名，密码(通常经过 RAS 加密)
"Cookie": "anonymid=jru48ifmt3j3si; _r01_=1; ln_uact=17752558702; ln_hurl=http://head.xiaonei.com/photos/0/0/men_main.gif; _de=32B20555AD3784A6BF2D3D01B72FE013; jebe_key=757177d7-4364-40db-be6b-cdd5b17c1d81%7C077a3e2b1c00096d5c13732ceee74ce5%7C1549513361097%7C1%7C1551858470234; depovince=GW; jebecookies=a63ced8a-74ea-4f0f-8a3d-8081b5b5867e|||||; JSESSIONID=abcBw69BBdAWu92C1eQMw; ick_login=35cfae73-b0cc-4c81-8fcc-a9f8ccb8e3aa; Hm_lvt_bfc6c23974fbad0bbfed25f88a973fb0=1553336590; Hm_lpvt_bfc6c23974fbad0bbfed25f88a973fb0=1553336620; p=78a2864fb08cd420e9d7735422c314ca2; first_login_flag=1; t=ac4860a9caa4ff859b2470a628f053312; societyguester=ac4860a9caa4ff859b2470a628f053312; id=966924492; xnsid=55ec2702; ver=7.0; loginfrom=null;

```
jebe_key=757177d7-4364-40db-be6b-cdd5b17c1d81%7C077a3e2b1c00096d5c13732ceee74ce5
%7C1553336624497%7C1%7C1553336624822; wp_fold=0",
}
response = requests.get('http://www.renren.com/966924492',headers=headers)
print(response.text)
```

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 。

案例：模拟表单实现人人网登录

```
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)
```

图片下载

```
import requests

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"}
```

```
response = requests.get("http://c1.haibao.cn/img/600_0_100_1/1549794487.7856/fa60e1e7264e6082569d729e4ee302dd.jpg", headers = headers)

with open('./images/img.jpg','wb') as file:
    file.write(response.content)
```

视频下载

```
import requests

def download_file(url, path):
    with requests.get(url, stream=True) as r:
        chunk_size = 1024
        content_size = int(r.headers['content-length'])
        print('下载开始。。。')
        with open(path, "wb") as f:
            for chunk in r.iter_content(chunk_size=chunk_size):
                f.write(chunk)
    print('下载结束。。。')


if __name__ == '__main__':
    url = 'https://gss3.baidu.com/6LZ0ej3k1Qd3ote6lo7D0j9wehsv/tieba-smallvideo-transcode/51722773_98a3bb47a51fef46145630154f007ddc_0.mp4'
    path = './videos/v.mp4'
    download_file(url, path)
```

处理 HTTPS 请求 SSL 证书验证

Requests 也可以为 HTTPS 请求验证 SSL 证书，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 ， SSL 验证默认是开启的。

可以使用 verify 参数设置是否验证 SSL 证书

True：校验某个主机的 SSL 证书

False：关闭某个主机的 SSL 证书

网页的证书没有被官方 CA 机构信任，所以这里会出现证书验证的错误：

requests.exceptions.SSLError: HTTPSConnectionPool(host='kennethreitz.com', port=443): Max retries exceeded with url: / (Caused by SSLError(SSLError("bad handshake: Error([('SSL routines', 'ssl3_read_bytes', 'tlsv1 alert internal error')],)",),))

关闭证书的验证：

```
import requests
import urllib3
urllib3.disable_warnings()
response = requests.get("https://www.12306.cn/mormhweb/",verify=False)
```