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

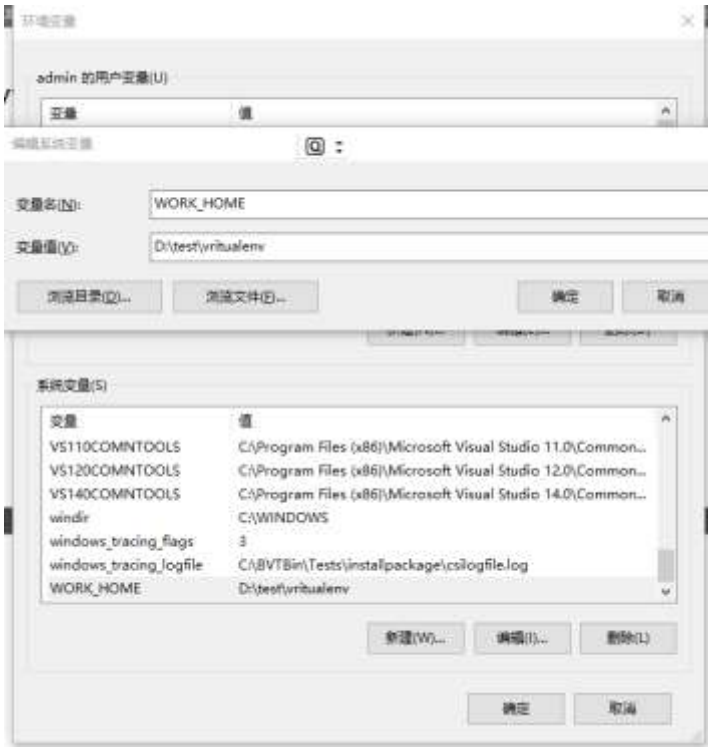
virtualenvwrapper 安装

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

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。

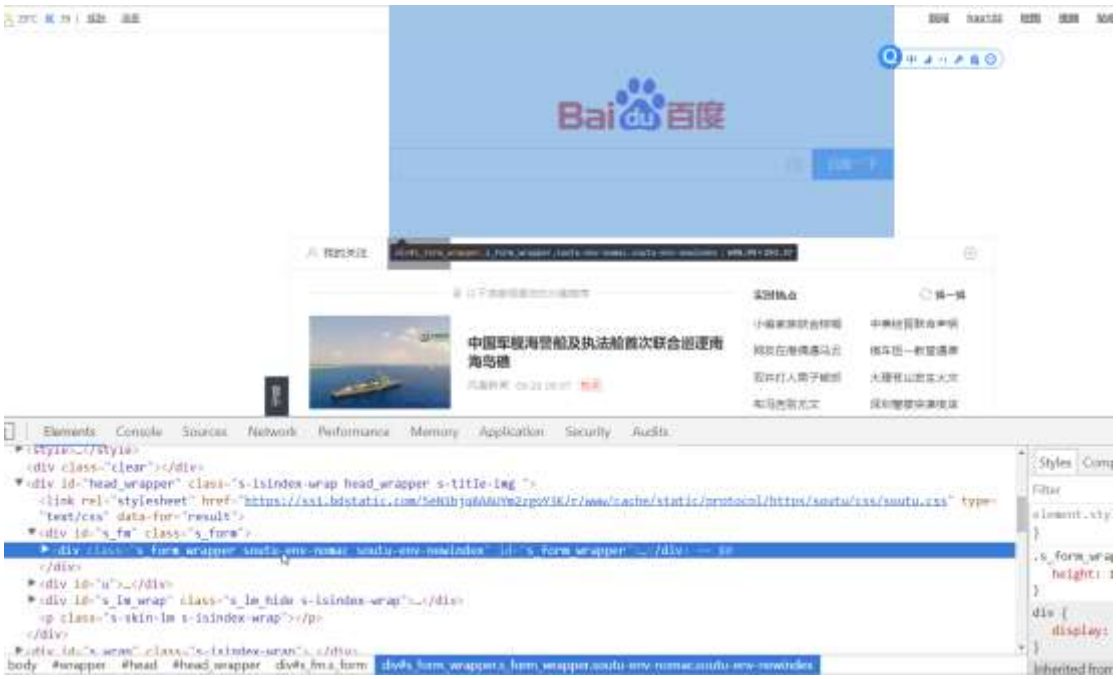
HTTPS（Hypertext Transfer Protocol over Secure Socket Layer）HTTP 的安全版，在 HTTP 下加入 SSL 层。

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 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)
# 调用 chardet.detect()来识别文本编码
print(response. apparent_encoding)
# 查看响应码
print(response.status_code)
```

编码问题

编码获取原理

requests 会从服务器返回的响应头的 Content-Type 去获取字符集编码，如果 content-type 有 charset 字段那么 requests 才能正确识别编码，否则就使用默认的 ISO-8859-1.

iso-8859-1 是 Latin-1 或“西欧语言”

如何获取正确的编码?

那些不规范的页面往往 content-type 没有 charset 字段

响应对象中有 apparent_encoding 通过调用 chardet.detect()来识别文本编码。但是需要注意的是，这有些消耗计算资源。

requests 的 text() 跟 content() 有什么区别?

text 属性返回的是 decode()解码的 Unicode 型的数据，如果 headers 没有 charset 字符集的化 text 属性会调用 chardet 来计算字符集

而 content 属性返回的是 bytes 型的原始数据，更节省计算资源。

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

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)
```

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'...
```

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)
```