目录

主流Python爬虫框架：Scrapy

- 最新版本：`v1.0`
- 更新时间：`20200728`

简介

介绍Python领域最流行的爬虫框架Scrapy的概况，基本用法和实际用法举例，以及Scrapy shell基本介绍；以及一些心得总结，比如Scrapy和PySpider的对比、看到的和抓取到的网页源码不同、PyCharm中如何调试Scrapy、robots.txt、丢失部分链接等，以及Scrapy相关的其他一些内容和参考文档和资料。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/python_spider_scrapy: 主流Python爬虫框架：Scrapy

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 主流Python爬虫框架：Scrapy book.crifan.com
- 主流Python爬虫框架：Scrapy crifan.github.io

离线下载阅读

- 主流Python爬虫框架：Scrapy PDF
- 主流Python爬虫框架：Scrapy ePub
- 主流Python爬虫框架：Scrapy Mobi

版权说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

简介

`Scrapy`是目前Python的最流行的爬虫框架。

TODO:

- 【整理】pyspider vs scrapy

基本原理

引用别人总结的：

- 入口为`start_requests`方法或者`start_urls`数组
- `parse`方法一般用`yield`来生成一个`迭代器`，返回一个或多个`Request / Item`，或者二者都有
- 每个`Request`都会默认使用上次请求的`Cookies`信息
- 对于需登陆的数据，可以在`start_requests`中进行登陆
- 页面内常用`xpath`或者`css selecter`进行解析，也可以使用`BeautifulSoup`等其它工具
- `Spider`的数据请求部分叫`Downloader`，`Scrapy`是基于事件的，因此从宏观上讲，数据请求和响应是线程分离的
- `Spider`的结果用`pipeline`来进行处理，可以自定义各种`pipeline`，`pipeline`中可以对抓取到的`item`进行去重、数据存储等操作
- `Spider`与`Downloader`中间有一种叫做`middleware`的东西，进行管道式数据加工，有点类似于过滤器、代理之类的
- 随机`User_agent`就是通过`downloader middleware`来实现的

教程和资料

- 官网
  - Scrapy | A Fast and Powerful Scraping and Web Crawling Framework
- Github
  - scrapy/scrapy: Scrapy, a fast high-level web crawling & scraping framework for Python.
- 教程
  - 中文
    - crapy入门教程 — Scrapy 0.24.6 文档
  - 英文
    - Scrapy Tutorial — Scrapy 2.2.1 documentation

基本用法

下面介绍Scrapy的基本用法。

官网快速教程

安装：

```
pip install scrapy
```

新建爬虫文件和写基本的代码：

```
cat > myspider.py <<EOF
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://blog.scrapinghub.com']

    def parse(self, response):
        for title in response.css('.post-header>h2'):
            yield {'title': title.css('a ::text').get()}

        for next_page in response.css('a.next-posts-link'):
            yield response.follow(next_page, self.parse)
EOF
```

运行爬虫：

```
scrapy runspider myspider.py
```

更多细节可参考官网教程：

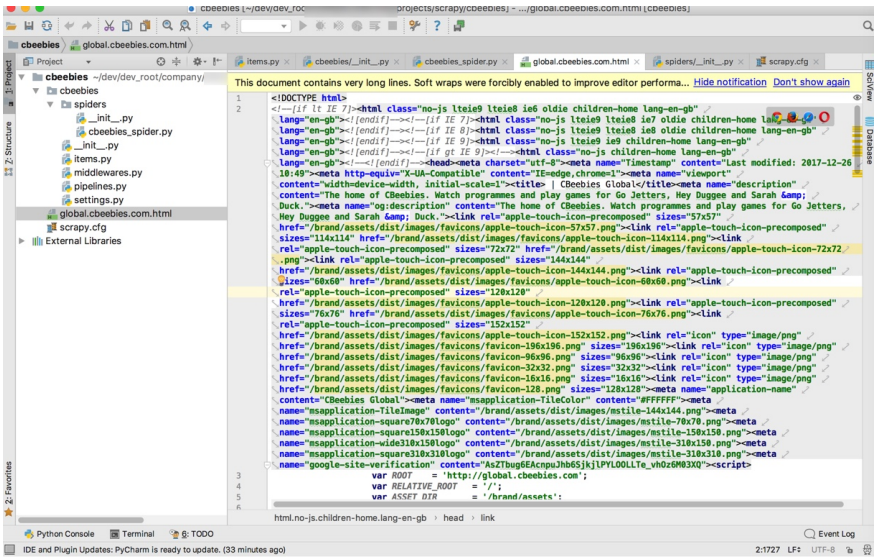
Scrapy入门教程 — Scrapy 0.24.6 文档

举例

举例：新建cbeebies爬虫

```
scrapy startproject cbeebies

cd cbeebies

scrapy crawl Cbeebies
```

即可生成对应的html文件：



举例：抓取Youtube中humf的字幕

需求

想要抓取：

Humf – Official Channel – YouTube – YouTube

其下的各个视频系列中的字幕文件。

其中获取YouTube字幕本身是依赖于：

http://www.yousubtitles.com

去实现的。

实现

```
scrapy startproject youtubeSubtitle

cd youtubeSubtitle

scrapy genspider YoutubeSubtitle youtube.com
```

生成文件内容：

```
# -*- coding: utf-8 -*-
import scrapy
class YoutubesubtitleSpider(scrapy.Spider):
    name = "YoutubeSubtitle"
    allowed_domains = ["youtube.com"]
    start_urls = ["http://youtube.com/"]
    def parse(self, response):
        pass
```

继续优化和添加内容：

```
# -*- coding: utf-8 -*-
import scrapy
class YoutubesubtitleSpider(scrapy.Spider):
    name = "YoutubeSubtitle"
    allowed_domains = ["youtube.com", "yousubtitles.com"]
    start_urls = [
        "https://www.youtube.com/user/theofficialhumf/play
    ]

    def parse(self, response):
        respUrl = response.url
        print "respUrl=%s"%(respUrl)
        filename = respUrl.split("/")[-2] + ".html"
        with open(filename, "wb") as f:
            f.write(response.body)
```

去运行，结果无法访问YouTube，所以要去加上代理：

`/xxx/scrapy/youtubeSubtitle/youtubeSubtitle/settings.py`

中设置：

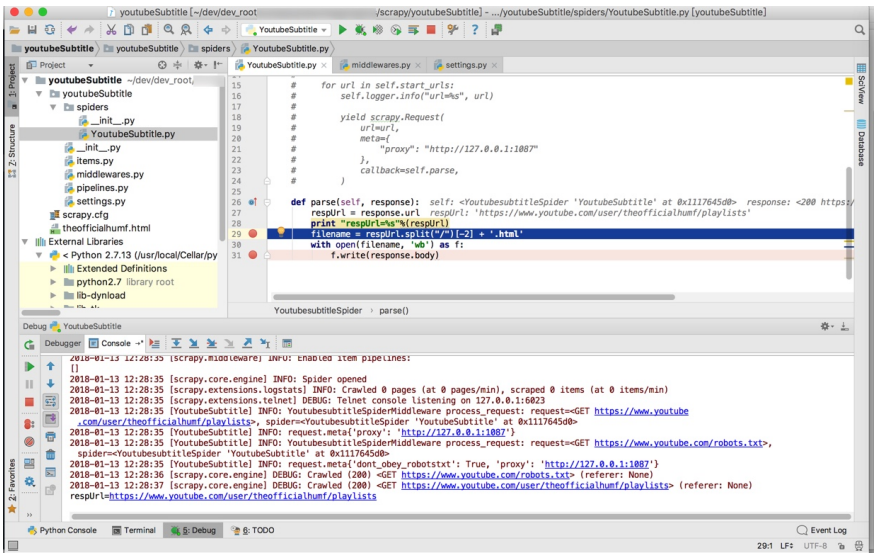
```
DOWNLOADER_MIDDLEWARES = {
    # 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMid
    # 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMid
    "youtubeSubtitle.middlewares.ProxyMiddleware" : 1
}
```

`/xxx/scrapy/youtubeSubtitle/youtubeSubtitle/middlewares.py`

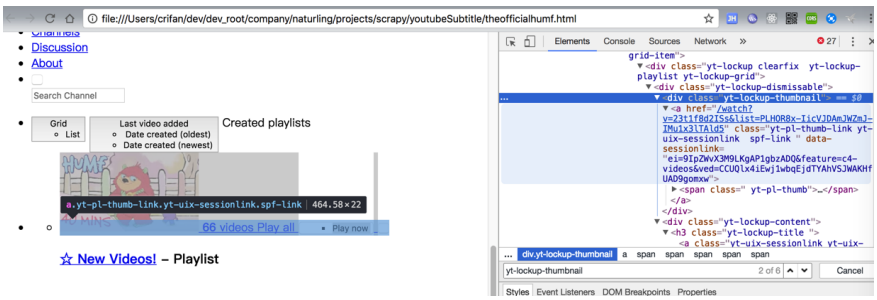
```
# Start your middleware class
class ProxyMiddleware(object):
    def process_request(self, request, spider):
        spider.logger.info("YoutubesubtitleSpiderMiddleware
        request.meta['proxy'] = "http://127.0.0.1:1087"
        spider.logger.info("request.meta%s", request.meta)
```

其中 `http://127.0.0.1:1087` 是本地的ss的代理。

即可通过代理访问到YouTube内容了：



此处希望提取的是：



中的：

`'//div[@class="yt-lockup-thumbnail"]/a[starts-with(@href, "/watch")]'`

经过尝试，是通过：

```
scrapy shell

fetch("https://www.youtube.com/user/theofficialhumf/playlis

view(response)

response.body

response.xpath('//div[@class="yt-lockup-thumbnail"]/a[start
```

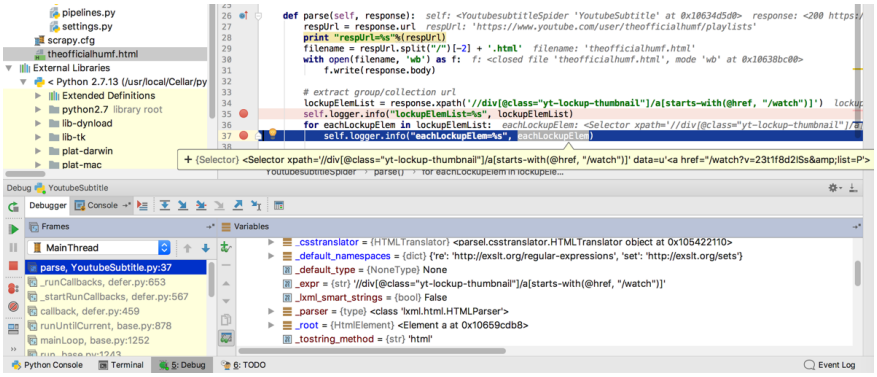
输出了我们希望的html，拷贝出来效果是：

```
<div class="yt-lockup-thumbnail">
          <a href="/watch?v=23t1f8d2ISs&amp;list=PLHOR8x-IicVJDAmJWZmJ-IMu1x3lTAldS" class="yt-pl-thumb-link yt-uix-sessionlink  spf-link "
          data-sessionlink="ei=9IpZWvX3M9LKgAP1gbzADQ&amp;feature=c4-videos&amp;ved=CCUQlx4iEwj1wbqEjdTYAhVSJWAKHfUAD9gomxw">
      <span class=" yt-pl-thumb">

    <span class="video-thumb  yt-thumb yt-thumb-196">
    <span class="yt-thumb-default">
      <span class="yt-thumb-clip">

  <img src="https://i.ytimg.com/vi/23t1f8d2ISs/hqdefault.jpg?sqp=-oaymwEWCMQBEG5IWvKriqkDCQgBFQAAiEIYAQ==&amp;rs=AOn4CLAlYOJF6LvCxIQsdzkOtRxyA3l1pw"
  width="196" data-ytimg="1" alt="" onload=";window.__ytRIL &amp;&amp; __ytRIL(this)" aria-hidden="true">

        <span class="vertical-align"></span>
      </span>
    </span>
  </span>


    <span class="sidebar">
      <span class="yt-pl-sidebar-content yt-valign">
        <span class="yt-valign-container">
          <span class="formatted-video-count-label">
      <b>66</b> videos
  </span>

          <span class="yt-pl-icon yt-pl-icon-reg yt-sprite"></span>
        </span>
      </span>
    </span>

    <span class="yt-pl-thumb-overlay">
      <span class="yt-pl-thumb-overlay-content">
        <span class="play-icon yt-sprite"></span>
        <span class="yt-pl-thumb-overlay-text">
Play all
        </span>
      </span>
    </span>

      <span class="yt-pl-watch-queue-overlay">
    <span class="thumb-menu dark-overflow-action-menu video-actions">
  <button type="button" aria-haspopup="true" class="yt-uix-button-reverse flip addto-watch-queue-menu spf-nolink hide-until-delayloaded
  yt-uix-button yt-uix-button-dark-overflow-action-menu yt-uix-button-size-default yt-uix-button-has-icon no-icon-markup yt-uix-button-empty"
  onclick=";return false;" aria-expanded="false"><span class="yt-uix-button-arrow yt-sprite"></span><ul class="watch-queue-thumb-menu
```

继续调试，写代码：

```
def parse(self, response):
    respUrl = response.url
    print "respUrl=%s"%(respUrl)
    filename = respUrl.split("/")[-2] + '.html'
    with open(filename, 'wb') as f:
        f.write(response.body)

    # extract group/collection url
    lockupElemList = response.xpath('//div[@class="yt-locku
    self.logger.info("lockupElemList=%s", lockupElemList)
    for eachLockupElem in lockupElemList:
        self.logger.info("eachLockupElem=%s", eachLockupEle
```

是可以获得对应的 `div` 的 `element` 的：



接着去用获取其中a的href的值，方式是：

在Scrapy shell中调试的效果：

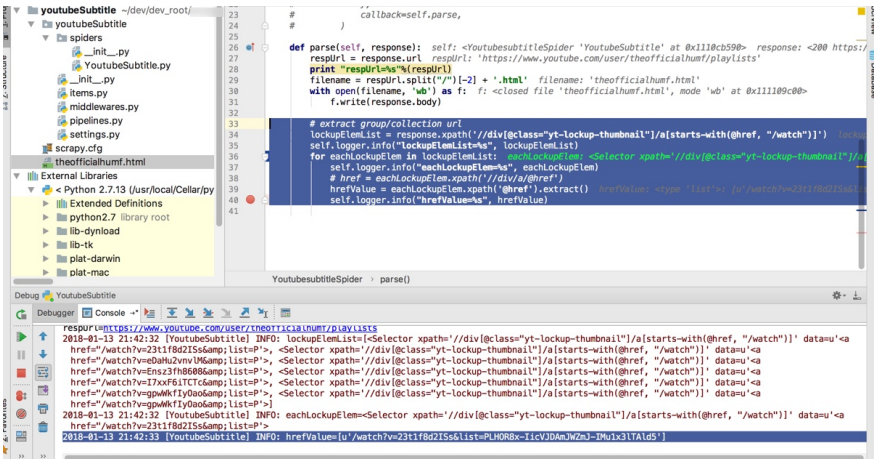
```
>>> response.xpath('//div[@class="yt-lockup-thumbnail"]/a[s

<Selector xpath='//div[@class="yt-lockup-thumbnail"]/a[sta

>>> response.xpath('//div[@class="yt-lockup-thumbnail"]/a[s

u'/watch?v=23t1f8d2ISs&list=PLHOR8x-IicVJDAmJWZmJ-IMu1x3lTA
```

代码：

```
# extract group/collection url
lockupElemList = response.xpath('//div[@class="yt-lockup-th
self.logger.info("lockupElemList=%s", lockupElemList)

for eachLockupElem in lockupElemList:
    self.logger.info("eachLockupElem=%s", eachLockupElem)
    # href = eachLockupElem.xpath('//div/a/@href')
    hrefValue = eachLockupElem.xpath('@href').extract()
    self.logger.info("hrefValue=%s", hrefValue)
```

得到输出：

```
2018-01-13 21:42:33 [YoutubeSubtitle] INFO: hrefValue=[u'/v
```



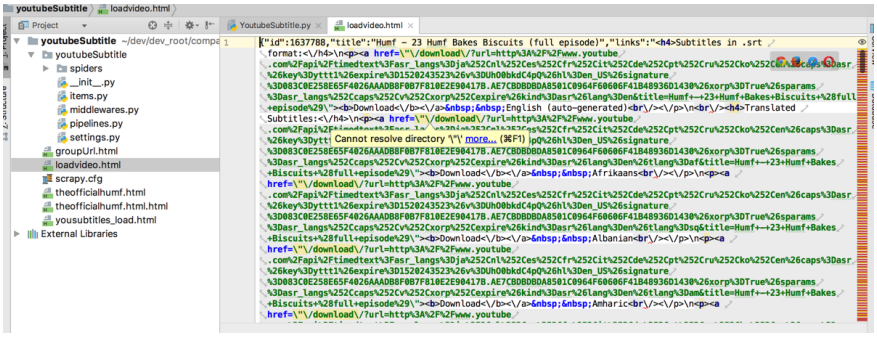
继续调试，为了获取网页的全部内容，经过研究，逻辑是：

```
Request URL: http://www.yousubtitles.com/loadvideo/23t1f8d2
Request Method: POST
```

所以写成代码：

```
import re

            foundVideoId = re.search(r'v=(?P&lt;videoId&gt;
            if foundVideoId:
                videoId = foundVideoId.group("videoId")
                self.logger.info("videoId=%s", videoId) # u

                # http://www.yousubtitles.com/loadvideo/23t
                loadVideoUrl = "http://www.yousubtitles.com
                self.logger.info("loadVideoUrl=%s", loadVid
                yield scrapy.Request(url=loadVideoUrl, call

def parseLoadVideoResp(self, response):
    """
    parse response of yousubtitles load video for each yout
    :param response:
    :return:
    """
    # for debug
    self.saveHtml("loadvideo", response.body)
    respUrl = response.url
    self.logger.info("respUrl=%s", respUrl)
```

效果：



接着去解析html源码，经过考虑决定用之前就用过的BeautifulSoup

先去安装：

```
pip install beautifulsoup4
```

然后写解析代码：

```
decodedLinksDict = json.loads(response.body)
self.logger.info("decodedLinksDict=%s", decodedLinksDict)
linksHtml = decodedLinksDict["links"]
# self.logger.info("linksHtml=%s", linksHtml)
linksSoup = BeautifulSoup(linksHtml)
englishNode = linksSoup.find(lambda tag : tag.name == "p" a
if englishNode:
    # self.logger.info("englishNode.contents=%s", englishNo
    self.logger.info("englishNode.text=%s", englishNode.tex
    # self.logger.info("englishNode=%s", englishNode)
    downloadHref = englishNode.a["href"]
    self.logger.info("downloadHref=%s", downloadHref) # /do
    downloadUrl = "http://www.yousubtitles.com" + downloadH
    self.logger.info("downloadUrl=%s", downloadUrl)
```

至此，完整代码是：

举例

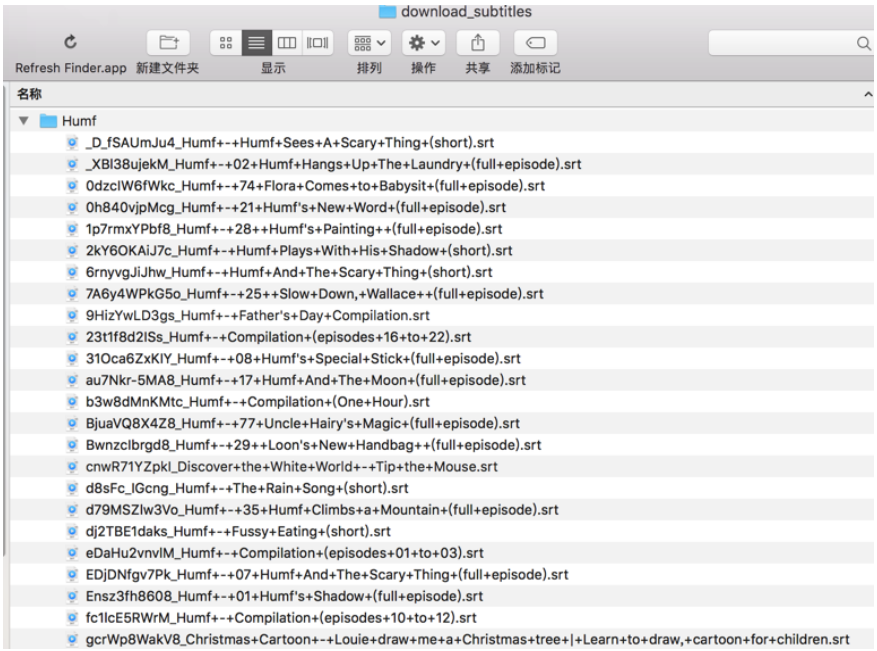
```
# -*- coding: utf-8 -*-
import scrapy
# from scrapy import Request, Spider
from urllib import urlencode, unquote
import re
import json
from bs4 import BeautifulSoup
import os

class YoutubesubtitleSpider(scrapy.Spider):
    name = 'YoutubeSubtitle'
    allowed_domains = ['youtube.com', "yousubtitles.com"]
    start_urls = [
        "https://www.youtube.com/user/theofficialhumf/play
    ]

    # def start_requests(self):
    #     """This is our first request to grab all the urls
    #     """
    #     for url in self.start_urls:
    #         self.logger.info("url=%s", url)
    #
    #         yield scrapy.Request(
    #             url=url,
    #             meta={
    #                 "proxy": "http://127.0.0.1:1087"
    #             },
    #             callback=self.parse,
    #         )

    def saveToFile(self, filename, content, suffix=".html"
        filename = filename + suffix
        with open(filename, 'wb') as f:
            f.write(content)

    def parse(self, response):
        respUrl = response.url
        print "respUrl=%s"%(respUrl)
        filename = respUrl.split("/")[-2] + '.html'
        self.saveToFile(filename=filename, content=response
        # with open(filename, 'wb') as f:
        #     f.write(response.body)

        # extract group/collection url
        lockupElemList = response.xpath('//div[@class="yt-
        self.logger.info("lockupElemList=%s", lockupElemLis
        for eachLockupElem in lockupElemList:
            self.logger.info("eachLockupElem=%s", eachLocku
            # href = eachLockupElem.xpath('//div/a/@href')
            hrefValue = eachLockupElem.xpath('@href').extra
```

```
            self.logger.info("hrefValue=%s", hrefValue)
            groupUrl = u"https://www.youtube.com" + hrefVa
            self.logger.info("groupUrl=%s", groupUrl)

            yield scrapy.Request(url=groupUrl, callback=se

    def parseEachYoutubeUrl(self, response):
        """
        parse each youtube url from group url's response
        :param response:
        :return:
        """
        # for debug
        respUrl = response.url
        self.logger.info("respUrl=%s", respUrl)
        self.saveToFile("groupUrl", response.body)

        dataIdElemList = response.xpath('//li[@data-video-
        self.logger.info("dataIdElemList=%s", dataIdElemLis
        if dataIdElemList:
            dataIdElemListCount = len(dataIdElemList)
            self.logger.info("dataIdElemListCount=%s", data

            for eachDataIdElem in dataIdElemList:
                self.logger.info("eachDataIdElem=%s", eachD
                hrefValue = eachDataIdElem.xpath('@href').e
                self.logger.info("hrefValue=%s", hrefValue)
                singleVideoUrl = u"https://www.youtube.com"
                self.logger.info("singleVideoUrl=%s", sing
                # https://www.youtube.com/watch?v=23t1f8d2

                # # http://www.yousubtitles.com/load/?url=h
                # yousubtitlesUrlBase = "http://www.yousub
                # youtubeUrlParaDict = {"url" : singleVideo
                # encodedUrlPara = urlencode(youtubeUrlPara
                # fullYousubtitlesUrl = yousubtitlesUrlBase
                #
                # yield scrapy.Request(url=fullYousubtitles

                foundVideoId = re.search(r'v=(?P&lt;videoId
                if foundVideoId:
                    videoId = foundVideoId.group("videoId")
                    self.logger.info("videoId=%s", videoId)

                    # http://www.yousubtitles.com/loadvideo
                    loadVideoUrl = "http://www.yousubtitles
                    self.logger.info("loadVideoUrl=%s", loa
                    yield scrapy.Request(url=loadVideoUrl,

    def parseLoadVideoResp(self, response):
        """
```

```
        parse response of yousubtitles load video for each
        :param response:
        :return:
        """
        # for debug
        self.saveToFile("loadvideo", response.body)
        respUrl = response.url # 'http://www.yousubtitles.c
        self.logger.info("respUrl=%s", respUrl)

        decodedLinksDict = json.loads(response.body)
        self.logger.info("decodedLinksDict=%s", decodedLinl
        if "links" not in decodedLinksDict:
            self.logger.warning("links not in decodedLinksD
            # {u'error': 1}
        else:
            linksHtml = decodedLinksDict["links"]
            # self.logger.info("linksHtml=%s", linksHtml)
            linksSoup = BeautifulSoup(linksHtml)
            self.logger.info("linksSoup=%s", linksSoup)
            englishNode = linksSoup.find(lambda tag : tag.r
            if englishNode:
                # self.logger.info("englishNode.contents=%s
                self.logger.info("englishNode.text=%s", eng
                # self.logger.info("englishNode=%s", englis
                downloadHref = englishNode.a["href"]
                self.logger.info("downloadHref=%s", downloa
                downloadUrl = "http://www.yousubtitles.com"
                self.logger.info("downloadUrl=%s", downloac

                yield scrapy.Request(url=downloadUrl, callb
            else:
                self.logger.warning("can not find english s

    def parseDownloadSubtitlesResp(self, response):
        """
        parse download subtitles of youtube video
        :param response:
        :return:
        """
        # for debug
        respUrl = response.url # http://www.yousubtitles.co
        self.logger.info("respUrl=%s", respUrl)
        decodedUrl = unquote(respUrl) # http://www.yousubti
        self.logger.info("decodedUrl=%s", decodedUrl)
        foundIdName = re.search(r'[&amp;|?]v=(?P&lt;videoId
        if foundIdName:
            videoId = foundIdName.group("videoId") # '7A6y4
            videoTitle = foundIdName.group("videoTitle") #
            subtitleFilename = videoId + "_" + videoTitle

            downloadFolder = "download_subtitles/Humf"
```

```
            if not os.path.exists(downloadFolder):
                os.makedirs(downloadFolder)

            self.saveToFile(downloadFolder + "/" + subtitle
        else:
            self.logger.warning("can not find id and name")

    # def parseYousubtitlesUrl(self, response):
    #     """
    #     parse each yousubtitles.com download response
    #     :param response:
    #     :return:
    #     """
    #     # for debug
    #     self.saveToFile("yousubtitles_load", response.bod
    #     respUrl = response.url
    #     self.logger.info("respUrl=%s", respUrl)
```

即可下载到对应字幕文件了：



后续

其他一些额外优化和需求的实现：

向后传递参数的，可以借用meta

```
yield scrapy.Request(url=loadVideoUrl, callback=self.parseL
```

之后的callback中即可通过response.meta去取值：

```
def parseLoadVideoResp(self, response):
    humfGroupTitle = response.meta["humfGroupTitle"]
```

Scrapy shell

Scrapy提供了一个命令行 `shell` ，用于无需新建爬虫项目，即可直接调试爬取页面。

下面举例介绍简单的用法。

爬取一个url=页面：

```
scrapy shell “http://global.cbeebies.com/shows/“
```

查看返回页面的body：

```
response.body
```

查看头信息：

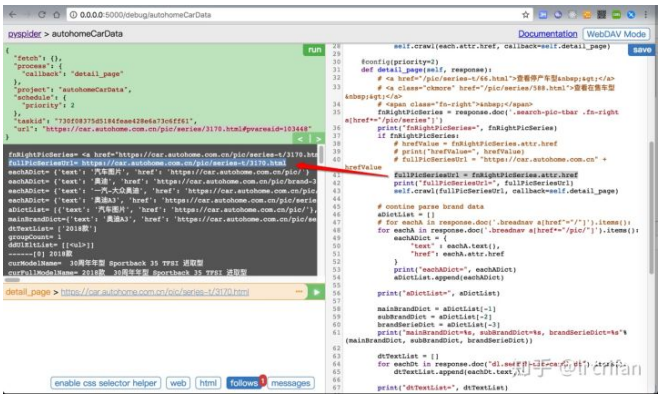
```
response.headers
```

用xpath查找所需元素：

```
response.xpath("//title")

response.xpath("//title").text()

response.xpath("//title").extract()

response.xpath("//title/text()").extract()
```

Scrapy vs PySpider

关于Scrapy和PySpider的对比，把之前别处的回复，整理过来供参考

- 对比
  - PySpider：简单易上手，带图形界面（基于浏览器页面）
    - 一图胜千言：在WebUI中调试爬虫代码
      - 
  - Scrapy：可以高级定制化实现更加复杂的控制
    - 但是没有一个直观的方便的调试界面
      - 一图胜千言：Scrapy一般是在命令行界面中调试页面返回数据
        - 
- 建议
  - 对于**普通的爬虫**，去爬你要的数据，PySpider已经足够你用了
    - 举例：
      - 【已解决】写Python爬虫爬取汽车之家品牌车系车型数据 – 在路上
  - 除非有其他**高级的特殊需求**，再考虑用Scrapy
    - 举例：
      - 用Scrapy去爬取Youtube中Humf的字幕

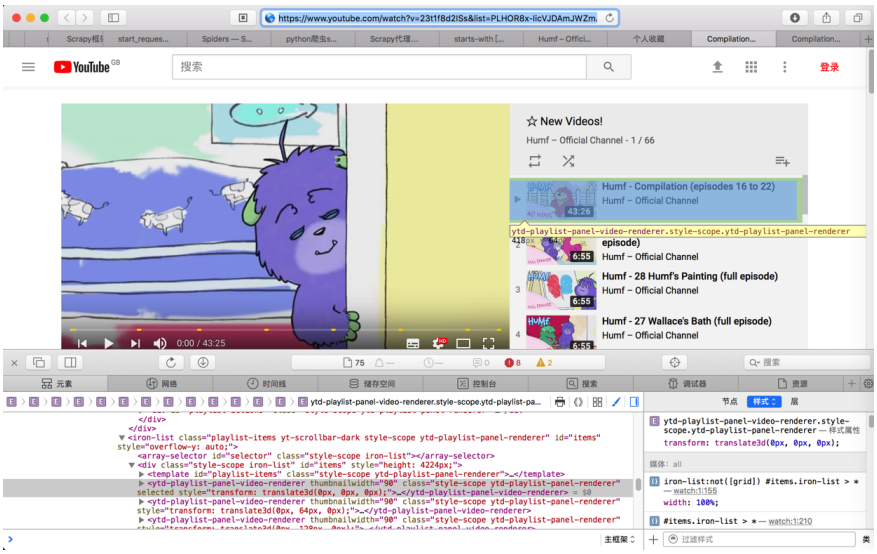
看到和抓到的源码不同

同一个url，浏览器直接打开和用scrapy去打开，html内容不一样，所以查找元素的 `xpath` 写法也不同

举例：

https://www.youtube.com/watch?v=23t1f8d2ISs&list=PLHOR8x-licVJDAmJWZmJ-IMu1x3ITAld5

用 `Safari` 浏览器打开的效果是：



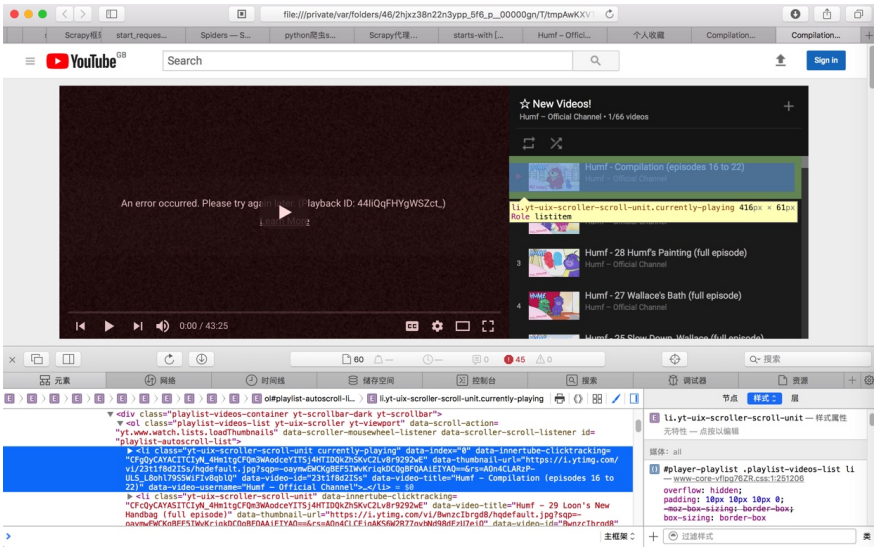
对应要找到元素的 `xpath` 写法是：

```
//*[@id="items"]/ytd-playlist-panel-video-renderer[1]
```

但是用 `Scrapy` 的 `shell` 中用 `view(response)` 保存后打开的本地html：

`file:///private/var/folders/46/2hjxz38n22n3ypp_5f6_p__00000gn/T/tmpAwKXVT.html`

的预览效果是：



对应 `xpath` 是：

```
//*[@id="playlist-autoscroll-list"]/li[1]
```

即，网页中看到内容，和Scarpy抓到的内容，不一样，对应要找的元素的xpath也就不一样了.

最后用：

```
>>> response.xpath('//ol[@id="playlist-autoscroll-list"]')
[<Selector xpath='//ol[@id="playlist-autoscroll-list"]' dat
>>> response.xpath('//ol[@id="playlist-autoscroll-list"]/li
```

才能找到要的元素：

心得：

用 `scrapy` 的 `shell` 的 `view(response)` 去打开浏览器，然后再去看html的xpath -》这样往往可以省去很多的调试时间

否则就会出现之前的问题：

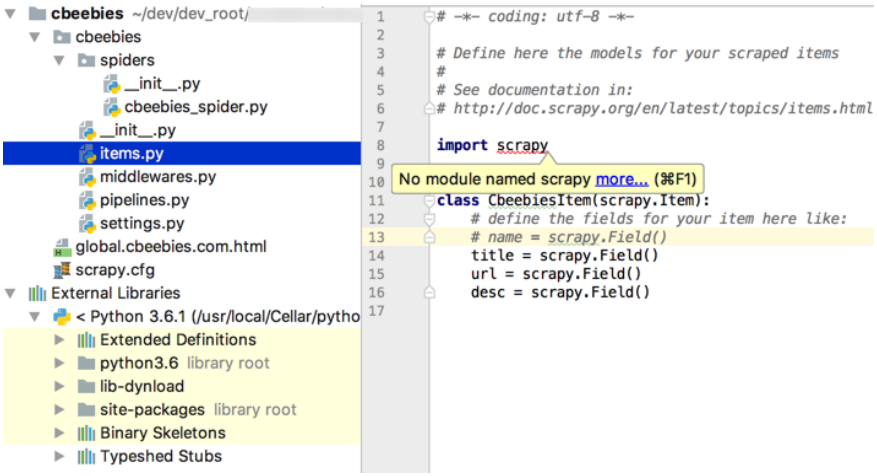
在用浏览器打开的页面中，调试了半天，找到想要的url的内容，结果实际上scrapy得到的html已经不一样，无法用之前写法得到想要的内容。

PyCharm调试Scrapy

此处整理用PyCharm调试Scrapy的一些心得。

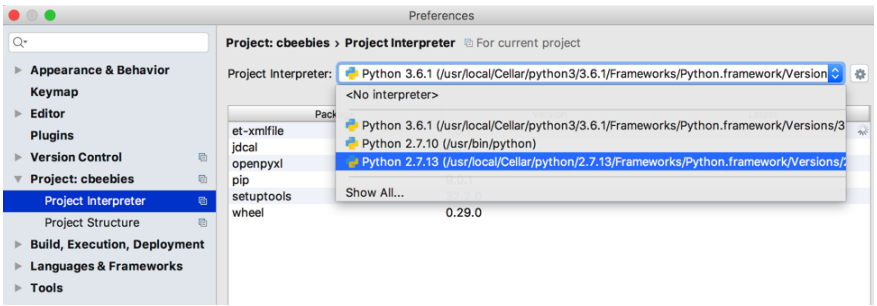
No module named scrapy

- **问题**：已安装Scrapy，但是PyCharm中找不到而报错 `No module named scrapy`

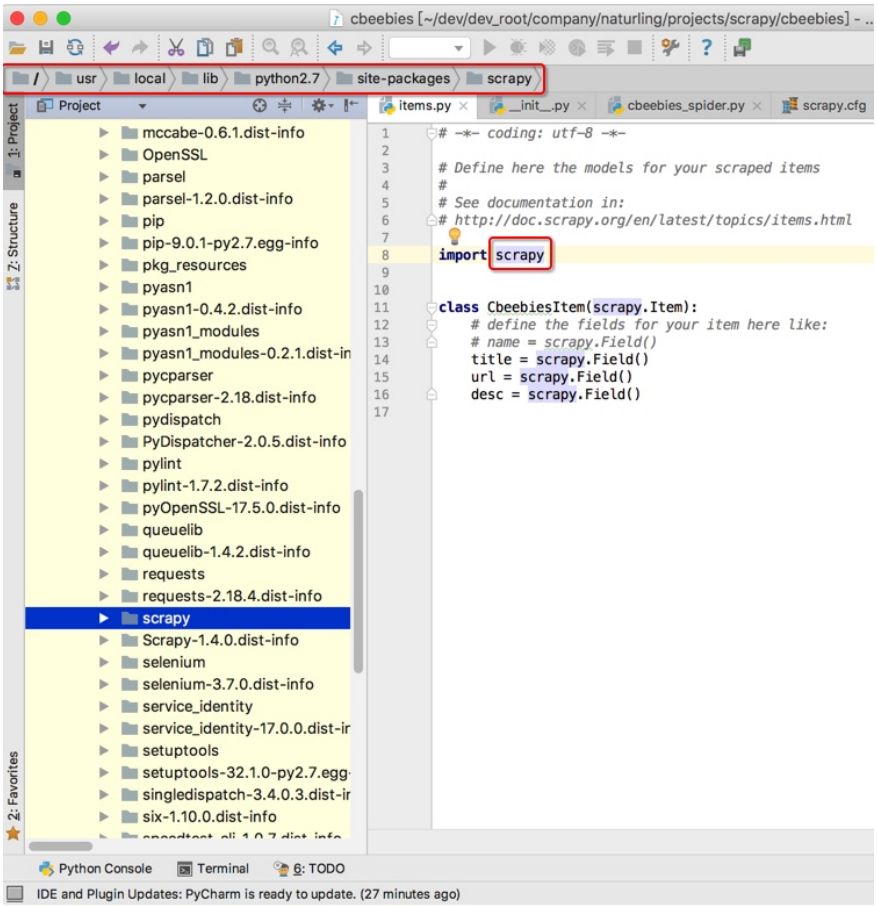


- **原因**：Python版本=Python解释器 设置不正确
- **解决办法**：设置正确的当前所使用的Python版本

`设置` -> `Project: xxx` -> `Project Interpreter` -> 换成你希望使用的Python版本

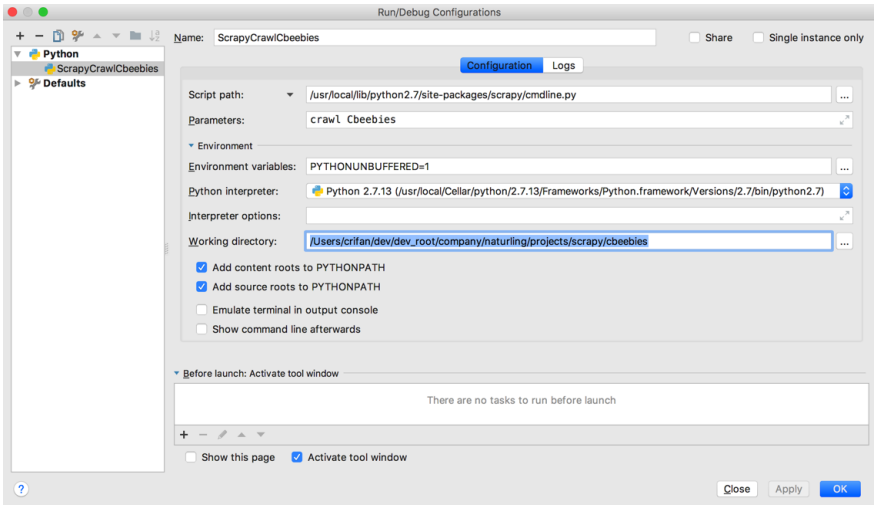


即可正常导入scrapy，以及看到对应源码：

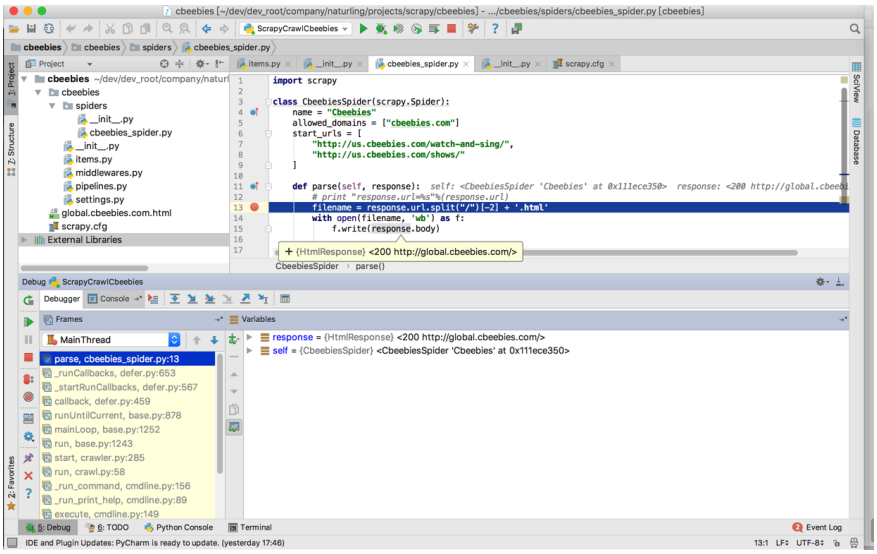


如何用PyCharm调试Scrapy

- **原理**：
  - **背景**：scrapy中有个 `cmdline.py` 是用于命令行运行Scrapy的。
    - 而PyCharm调试Scrapy，其实就是基于命令行方式去启动Scrapy
      - 所以就是去启动 `cmdline.py` 并去加上合适参数即可
- **步骤**：新建 `Python` 的 `Debug Configurations` ，加上参数：
- `Script path` ： `/{PYTHON_ROOT}/site-packages/scrapy/cmdline.py`
  - 举例： `/usr/local/lib/python2.7/site-packages/scrapy/cmdline.py`
- `Parameter` ：你要调试的Scrapy项目
  - 举例： `crawl Cbeebies`
- `Working directory` ：你的Scrapy爬虫所在根目录
  - 举例： `/User/crifan/dev/dev_root/xxx/projects/scrapy/cbeebies`



然后启动调试，即可实时调试了：



DEBUG: Forbidden by robots.txt

有些网站跟目录中有 `robots.txt` ，用于申明不让爬虫爬取。所以Scrapy默认就是遵守协议，不爬取这些网站。

为了能继续爬取，则需要去设置：

```
ROBOTSTXT_OBEY = False
```

即可。

丢失部分链接

之前遇到过：本来有18个url，但是最终只抓取到8个，缺了10个。

原因：可能是被filter过滤掉了，也可能是爬取频率太快导致的

解决办法：

`youtubeSubtitle/settings.py`

中配置为：

```
# Configure a delay for requests for the same website (defa
# See http://scrapy.readthedocs.org/en/latest/topics/settir
# See also autothrottle settings and docs
# DOWNLOAD_DELAY = 3
DOWNLOAD_DELAY = 0.5
# The download delay setting will honor only one of:
CONCURRENT_REQUESTS_PER_DOMAIN = 16
CONCURRENT_REQUESTS_PER_IP = 16

# Configure item pipelines

# See http://scrapy.readthedocs.org/en/latest/topics/item-p
ITEM_PIPELINES = {
   'youtubeSubtitle.pipelines.YoutubesubtitlePipeline': 300
}
```

以及每个Request都加上 `dont_filter=True` ：

```
yield scrapy.Request(url=groupUrl,
                     callback=self.parseEachYoutubeUrl,
                     dont_filter=True)


yield scrapy.Request(url=loadVideoUrl,
                     callback=self.parseLoadVideoResp,
                     method="POST",
                     dont_filter=True,
                     meta={"humfGroupTitle" : humfGroupTit

yield scrapy.Request(url=downloadUrl,
                     callback=self.parseDownloadSubtitlesRe
                     meta=response.meta,
                     dont_filter=True)
```

即可解决问题，速度慢一点爬取，最终爬取到了所有的url。

相关内容

下面介绍一些和 `Scrapy` 相关的一些内容：

Scrapyd

- Scrapyd
  - 简介
    - scrapyd是运行scrapy爬虫的服务程序，它支持以http命令方式发布、删除、启动、停止爬虫程序。而且scrapyd可以同时管理多个爬虫，每个爬虫还可以有多个版本
  - 文档
    - Github
      - scrapy/scrapyd: A service daemon to run Scrapy spiders
        - https://github.com/scrapy/scrapyd
          - crapyd is a service for running Scrapy spiders.
            - It allows you to deploy your Scrapy projects and control their spiders using an HTTP JSON API.
    - 官网教程
      - Scrapyd — Scrapyd 1.2.0 documentation
        - http://scrapyd.readthedocs.io/en/stable/
      - Overview — Scrapyd 1.2.0 documentation
        - http://scrapyd.readthedocs.io/en/stable/overview.html

附录

下面列出相关参考资料。

文档和资料

Scrapy官网教程

- 英文
  - Scrapy 1.5 documentation — Scrapy 1.5.0 documentation
    - https://doc.scrapy.org/en/latest/index.html
- 中文
  - 新
    - Scrapy 1.0 文档(未完成,只更新了intro部分,请谨慎参考) — Scrapy 1.0.5 文档
      - http://scrapy-chs.readthedocs.io/zh_CN/1.0/
  - 旧
    - Scrapy 0.25 文档 — Scrapy 0.24.1 文档
      - https://scrapy-chs.readthedocs.io/zh_CN/latest/index.html
    - Scrapy入门教程 — Scrapy 0.24.6 文档
      - http://scrapy-chs.readthedocs.io/zh_CN/0.24/intro/tutorial.html

help=帮助信息=语法

```
 ~ □ scrapy --help
Scrapy 2.2.1 - no active project

Usage:
  scrapy <command> [options] [args]

Available commands:
  bench         Run quick benchmark test
  commands
  fetch         Fetch a URL using the Scrapy downloader
  genspider     Generate new spider using pre-defined temp
  runspider     Run a self-contained spider (without creat
  settings      Get settings values
  shell         Interactive scraping console
  startproject  Create new project
  version       Print Scrapy version
  view          Open URL in browser, as seen by Scrapy

  [ more ]      More commands available when run from proj

Use "scrapy <command> -h" to see more info about a command
```

startproject创建项目的help

```
~ □ scrapy startproject -h
Usage
=====
  scrapy startproject <project_name> [project_dir]

Create new project

Options
=======
--help, -h              show this help message and exit

Global Options
--------------
--logfile=FILE          log file. if omitted stderr will be
--loglevel=LEVEL, -L LEVEL
                        log level (default: DEBUG)
--nolog                 disable logging completely
--profile=FILE          write python cProfile stats to FILE
--pidfile=FILE          write process ID to FILE
--set=NAME=VALUE, -s NAME=VALUE
                        set/override setting (may be repeat
--pdb                   enable pdb on failure
```

参考资料

- 使用scrapyd 管理爬虫 · 网络爬虫教程
- 基于scrapyd爬虫发布总结 - 黯然销魂掌2015 - 博客园
- 【记录】用Python的Scrapy去爬取cbeebies.com
- 【已解决】Mac中PyCharm中找不到已安装的Scrapy库: No module named scrapy
- 【记录】用Python的Scrapy去爬取Youtube中Humf的字幕
- 【记录】尝试Scrapy shell去提取cbeebies.com页面中的子url
- 【已解决】scrapy中警告：DEBUG: Forbidden by robots.txt
- 【已解决】Scrapy中丢失部分url链接没有抓取
- 【已解决】Scrapy如何向后续Request的callback中传递参数
- 【已解决】Scrapy如何加载全部网页内容
- 【已解决】如何从Scrapy的Selector中获取html元素a的href属性的值
- 【已解决】Scrapy如何添加本地socks代理以便能打开Youtube网页
- 【已解决】Scrapy的Python中如何解析部分的html字符串并格式化为html网页源码
- 【已解决】Mac中PyCharm中去加断点实时调试scrapy的项目
- 【整理】pyspider vs scrapy
- Scrapy Spider 原理 · Scrapy with Python3
- 【已解决】Scrapy如何添加本地socks代理以便能打开Youtube网页 – 在路上