
目录

前言	1.1
简介	1.2
基本用法	1.3
举例	1.3.1
Scrapy shell	1.3.2
心得	1.4
Scrapy vs PySpider	1.4.1
看到和抓到的不同	1.4.2
PyCharm调试Scrapy	1.4.3
robots.txt	1.4.4
丢失部分链接	1.4.5
相关内容	1.5
附录	1.6
文档和资料	1.6.1
参考资料	1.6.2

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

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您的版权，请通过邮箱联系我 [admin 艾特 crifan.com](#)，我会尽快删除。谢谢合作。

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。

更多其他电子书

本人 crifan 还写了其他 100+ 本电子书教程，感兴趣可移步至：

[crifan/crifan_ebook_readme](#): Crifan的电子书的使用说明

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2021-01-17 12:02:46

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 selector` 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](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)](#)协议发布 all right reserved, powered by Gitbook最后更新: 2020-08-09 10:19:44

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](#)

crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved，powered by Gitbook最后更新：2020-08-09 10:19:44

举例

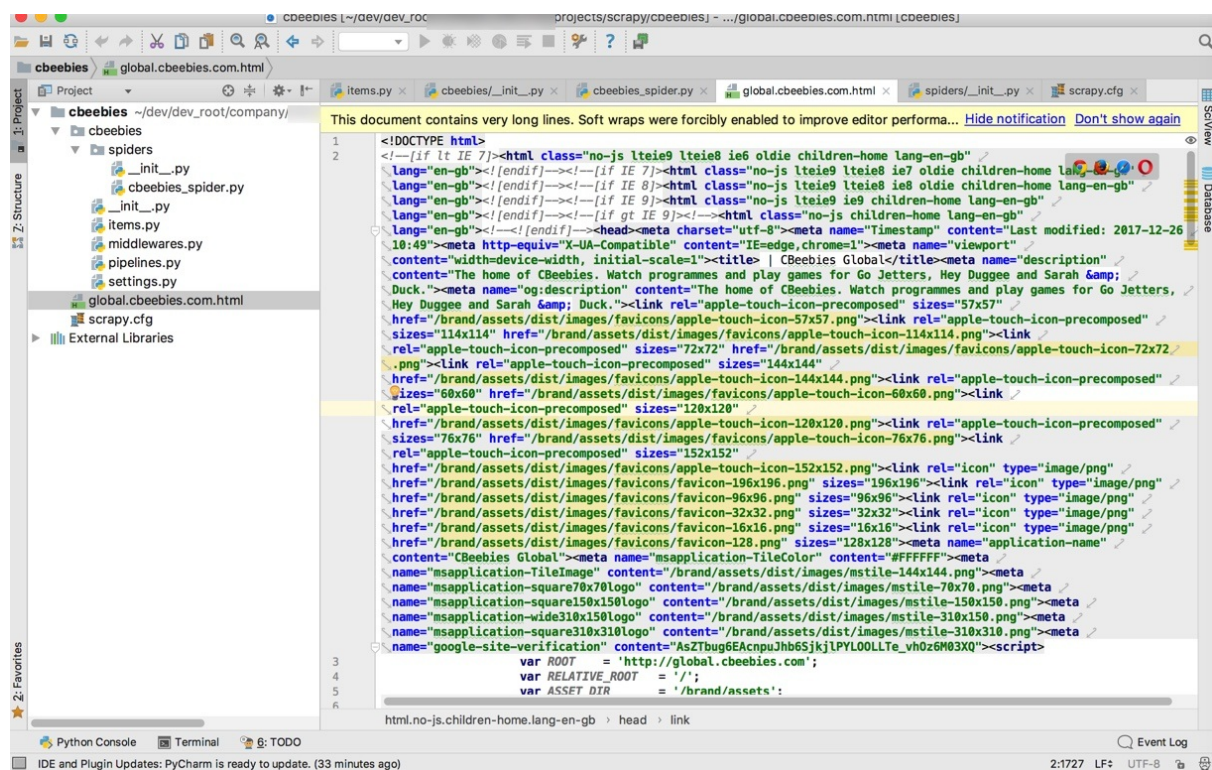
举例：新建cbeebies爬虫

```
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/playlists"
    ]

    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.HttpProxyMiddleware': 1,
    # 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware': None,
    "youtubeSubtitle.middlewares.ProxyMiddleware" : 1
```

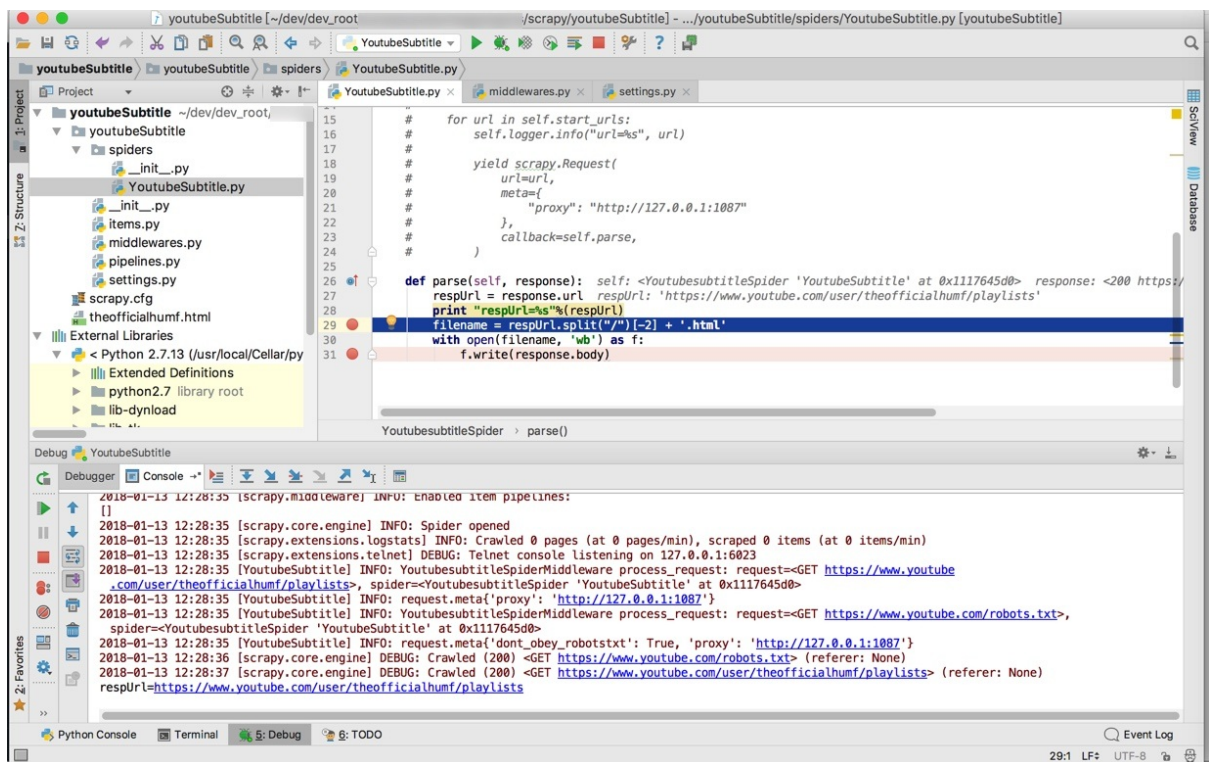
}

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

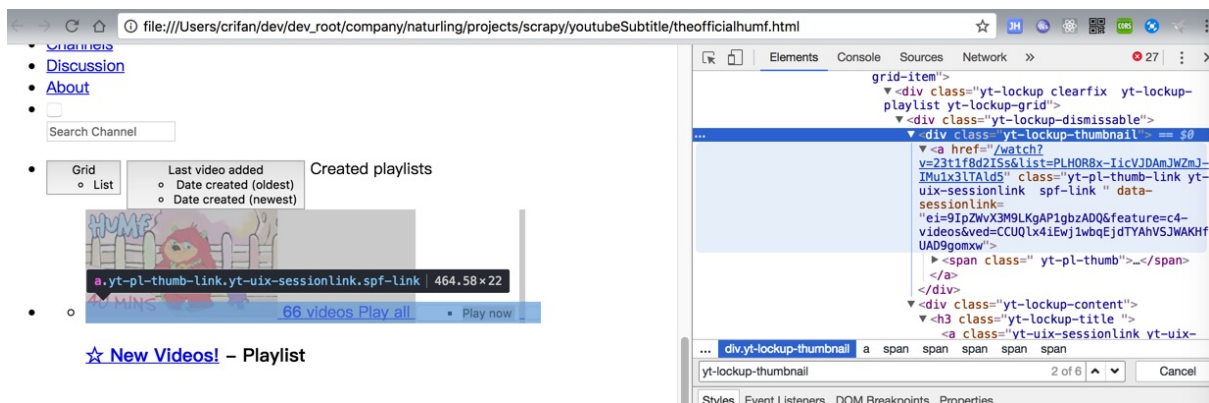
```
# Start your middleware class
class ProxyMiddleware(object):
    def process_request(self, request, spider):
        spider.logger.info("YoutubesubtitleSpiderMiddleware process_request: request=%s, spider=%s", request, spider)
        request.meta['proxy'] = "http://127.0.0.1:1087"
        spider.logger.info("request.meta%s",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

view(response)

response.body

response.xpath('//div[@class="yt-lockup-thumbnail"]/a[starts-with(@href, "/watch")]')[0].extract()
```

输出了我们希望的html, 拷贝出来效果是:

```
1 <div class="yt-lockup-thumbnail">
2   <a href="/watch?v=23t1f8d2ISs&list=PLH0R8x-IicVJDAmJWZmJ-IMu1x3lTAld5" class="yt-pl-thumb-link yt-ux-sessionlink spf-link "
   data-sessionlink="ei=9IpZwVX3M9LKgAP1gbzADQ&feature=c4-videos&ved=CCUQLx4iEwj1wbqEjdTYAhV5JWAKHfUAD9gomxw">
3     <span class="yt-pl-thumb">
4
5     <span class="video-thumb yt-thumb yt-thumb-196">
6       <span class="yt-thumb-default">
7         <span class="yt-thumb-clip">
8
9         
10
11       <span class="vertical-align"></span>
12     </span>
13   </span>
14 </div>
15
16   <span class="sidebar">
17     <span class="yt-pl-sidebar-content yt-valign">
18       <span class="yt-valign-container">
19         <span class="formatted-video-count-label">
20           <b>66</b> videos
21         </span>
22       </span>
23
24       <span class="yt-pl-icon yt-pl-icon-reg yt-sprite"></span>
25     </span>
26   </span>
27 </div>
28
29   <span class="yt-pl-thumb-overlay">
30     <span class="yt-pl-thumb-overlay-content">
31       <span class="play-icon yt-sprite"></span>
32       <span class="yt-pl-thumb-overlay-text">
33         Play all
34       </span>
35     </span>
36   </span>
37
38   <span class="yt-pl-watch-queue-overlay">
39     <span class="thumb-menu dark-overflow-action-menu video-actions">
40     <button type="button" aria-haspopup="true" class="yt-ux-button-reverse flip addto-watch-queue-menu spf-nolink hide-until-delayloaded
    yt-ux-button yt-ux-button-dark-overflow-action-menu yt-ux-button-size-default yt-ux-button-has-icon no-icon-markup yt-ux-button-empty"
    onclick=";return false;" aria-expanded="false"><span class="yt-ux-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)

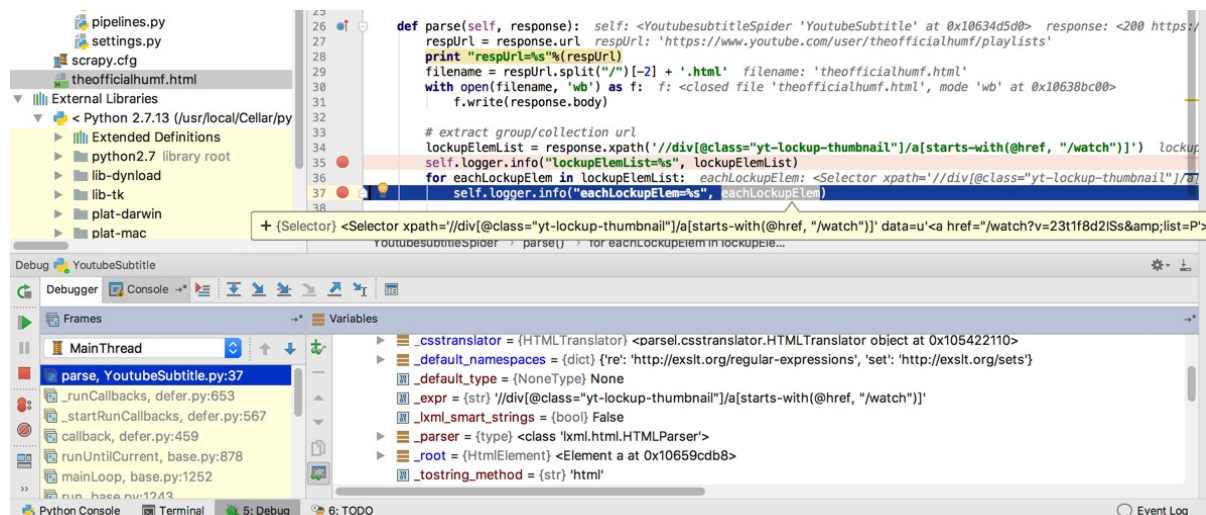
    # extract group/collection url
    lockupElemList = response.xpath('//div[@class="yt-lockup-thumbnail"]/a[starts-with(@href, "/watch")]')
```

```

self.logger.info("lockupElemList=%s", lockupElemList)
for eachLockupElem in lockupElemList:
    self.logger.info("eachLockupElem=%s", eachLockupElem)

```

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:



代码:

```

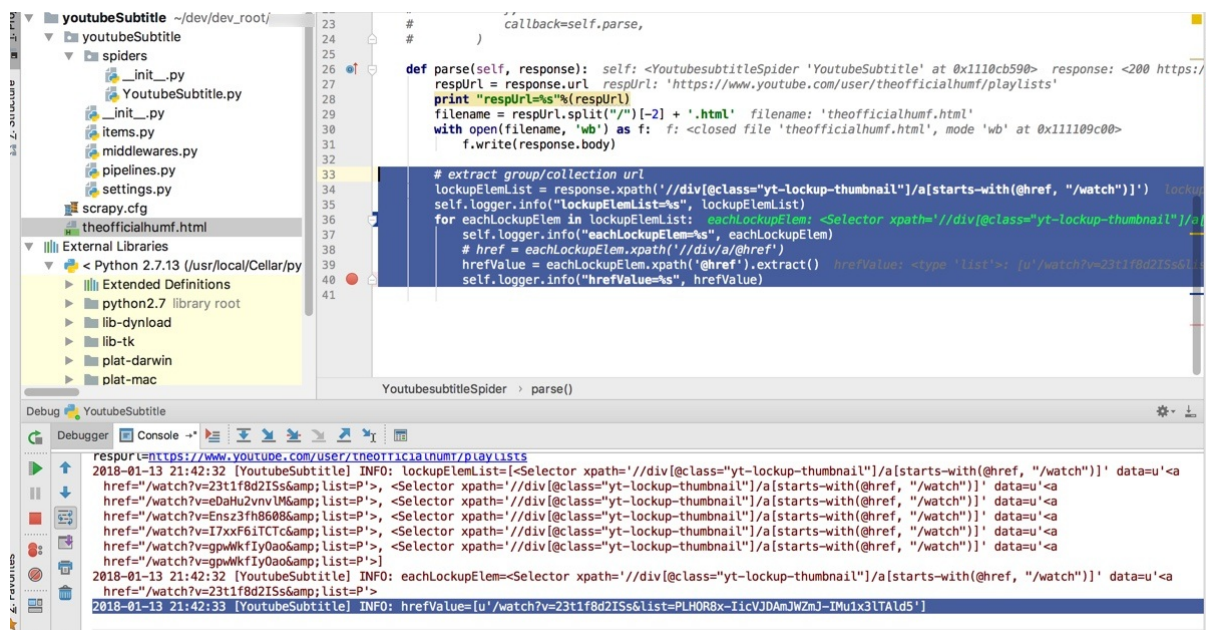
# extract group/collection url
lockupElemList = response.xpath('//div[@class="yt-lockup-thumbnail"]/a[starts-with(@href, "/watch")])
self.logger.info("lockupElemList=%s", lockupElemList)

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'/watch?v=23t1f8d2ISs&list=PLHOR8x-IicVJDAmJWzmJ-IMu1x3lTAld5']



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/23t1f8d2ISs>
Request Method: POST

所以写成代码：

```
import re

foundVideoId = re.search(r'v=(?P<videoId>[w\-\-]+)', singleVideoUrl)
if foundVideoId:
    videoId = foundVideoId.group("videoId")
    self.logger.info("videoId=%s", videoId) # u'23t1f8d2ISs'

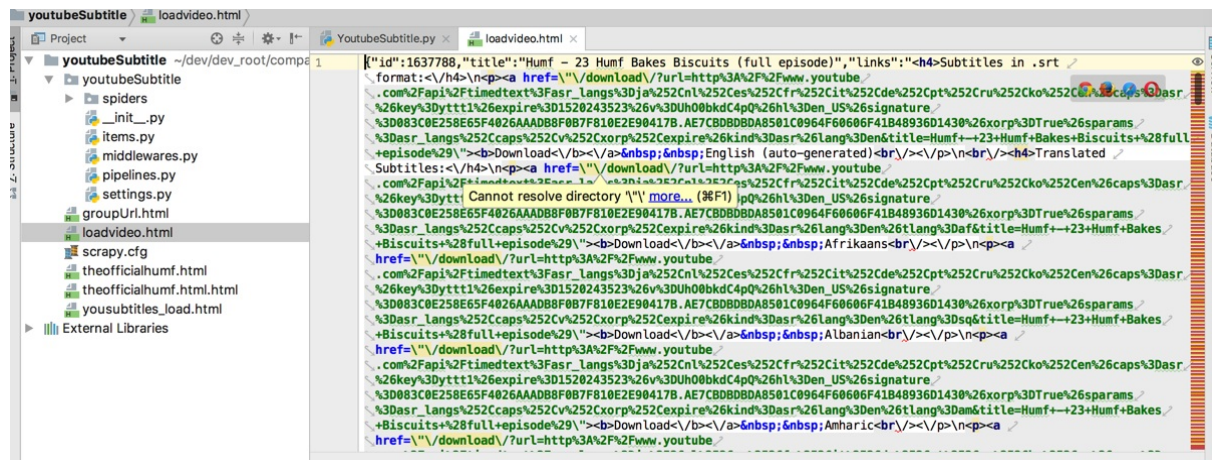
    # http://www.yousubtitles.com/loadvideo/23t1f8d2ISs
    loadVideoUrl = "http://www.yousubtitles.com/loadvideo/" + videoId
    self.logger.info("loadVideoUrl=%s", loadVideoUrl) # u'http://www.yousubtit
les.com/loadvideo/23t1f8d2ISs'
    yield scrapy.Request(url=loadVideoUrl, callback=self.parseLoadVideoResp, m
ethod="POST")

def parseLoadVideoResp(self, response):
    """
    parse response of yousubtitles load video for each youtube video id
    :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" and "English" in tag.get_text())
if englishNode:
    # self.logger.info("englishNode.contents=%s", englishNode.contents)
    self.logger.info("englishNode.text=%s", englishNode.text)
    # self.logger.info("englishNode=%s", englishNode)
    downloadHref = englishNode.a["href"]
    self.logger.info("downloadHref=%s", downloadHref) # /download/?url=...&title=Humf+-+Compilation+%28episodes+16+to+22%29
    downloadUrl = "http://www.yousubtitles.com" + downloadHref # http://www.yousubtitles.com/download/?url=http%3A%2F%2F...&title=Humf+-+Compilation+%28episodes+16+to+22%29
    self.logger.info("downloadUrl=%s", downloadUrl)
```

至此，完整代码是：

```
# -*- 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/playlists"
    ]

    # def start_requests(self):
    #     """This is our first request to grab all the urls of the profiles.
    #     """
    #     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, response.body)
        # with open(filename, 'wb') as f:
        #     f.write(response.body)

        # extract group/collection url
        lockupElemList = response.xpath('//div[@class="yt-lockup-thumbnail"]/a[starts-with(@href, "/watch")]')
        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)
            groupUrl = u"https://www.youtube.com" + hrefValue[0]
            self.logger.info("groupUrl=%s", groupUrl)

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

    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-id]/a[@href]')
self.logger.info("dataIdElemList=%s", dataIdElemList)
if dataIdElemList:
    dataIdElemListCount = len(dataIdElemList)
    self.logger.info("dataIdElemListCount=%s", dataIdElemListCount)

    for eachDataIdElem in dataIdElemList:
        self.logger.info("eachDataIdElem=%s", eachDataIdElem)
        hrefValue = eachDataIdElem.xpath('@href').extract()
        self.logger.info("hrefValue=%s", hrefValue)
        singleVideoUrl = u"https://www.youtube.com" + hrefValue[0]
        self.logger.info("singleVideoUrl=%s", singleVideoUrl)
        # https://www.youtube.com/watch?v=23t1f8d2ISs&list=PLHOR8x-I1cVJDAmJWZ
        # mJ-IMu1x3lTAld5&index=1

        # # http://www.yousubtitles.com/load/?url=https%3A%2F%2Fwww.youtube.com%2F
        # watch%3Fv%3DgtaC1JVjvO4%26index%3D58%26list%3DPLHOR8x-I1cVJDAmJWZmJ-IMu1x3lTAld5&ch=1
        # yousubtitlesUrlBase = "http://www.yousubtitles.com/load/"
        # youtubeUrlParaDict = {"url" : singleVideoUrl}
        # encodedUrlPara = urlencode(youtubeUrlParaDict)
        # fullYousubtitlesUrl = yousubtitlesUrlBase + encodedUrlPara
        #
        # yield scrapy.Request(url=fullYousubtitlesUrl, callback=self.parseYousub
titlesUrl)

        foundVideoId = re.search(r'v=(?P<videoId>[\w\-\+])', singleVideoUrl)
        if foundVideoId:
            videoId = foundVideoId.group("videoId")
            self.logger.info("videoId=%s", videoId) # u'23t1f8d2ISs'

            # http://www.yousubtitles.com/loadvideo/23t1f8d2ISs
            loadVideoUrl = "http://www.yousubtitles.com/loadvideo/" + videoId
            self.logger.info("loadVideoUrl=%s", loadVideoUrl) # u'http://www.yousu
btitles.com/loadvideo/23t1f8d2ISs'
            yield scrapy.Request(url=loadVideoUrl, callback=self.parseLoadVideoResp
, method="POST")

def parseLoadVideoResp(self, response):
    """
    parse response of yousubtitles load video for each youtube video id
    :param response:
    :return:
    """

```

```

# for debug
self.saveToFile("loadvideo", response.body)
respUrl = response.url # 'http://www.yousubtitles.com/loadvideo/Uh00bkdc4pQ'
self.logger.info("respUrl=%s", respUrl)

decodedLinksDict = json.loads(response.body)
self.logger.info("decodedLinksDict=%s", decodedLinksDict)
if "links" not in decodedLinksDict:
    self.logger.warning("links not in decodedLinksDict=%s", decodedLinksDict)
    # {'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.name == "p" and "English" in tag.get_text())
    if englishNode:
        # self.logger.info("englishNode.contents=%s", englishNode.contents)
        self.logger.info("englishNode.text=%s", englishNode.text)
        # self.logger.info("englishNode=%s", englishNode)
        downloadHref = englishNode.a["href"]
        self.logger.info("downloadHref=%s", downloadHref) # /download/?url=http%3A%2F%2Fwww.youtube.com%2Fapi%2Ftimedtext%3Fsignature%3D0300D3B6307468779D9FE34128AAC8FA4EE978E4.7D5E553538919EB4479D2AD710C693BABDED961D%26sparams%3Dasr_langs%252Ccaps%252Ccv%252Cxp%252Cexpire%26asr_langs%3Des%252Cru%252Cit%252Cnl%252Cpt%252Cde%252Cko%252Cja%252Cen%252Cfr%26caps%3Dasr%26hl%3Den_US%26xorp%3DTrue%26key%3Dytt1%26expire%3D1520333270%26v%3D23t1f8d2ISs%26kind%3Dasr%26lang%3Den&title=Humf+-+Compilation+%28episodes+16+to+22%29
        downloadUrl = "http://www.yousubtitles.com" + downloadHref # http://www.yousubtitles.com/download/?url=http%3A%2F%2Fwww.youtube.com%2Fapi%2Ftimedtext%3Fsignature%3D0300D3B6307468779D9FE34128AAC8FA4EE978E4.7D5E553538919EB4479D2AD710C693BABDED961D%26sparams%3Dasr_langs%252Ccaps%252Ccv%252Cxp%252Cexpire%26asr_langs%3Des%252Cru%252Cit%252Cnl%252Cpt%252Cde%252Cko%252Cja%252Cen%252Cfr%26caps%3Dasr%26hl%3Den_US%26xorp%3DTrue%26key%3Dytt1%26expire%3D1520333270%26v%3D23t1f8d2ISs%26kind%3Dasr%26lang%3Den&title=Humf+-+Compilation+%28episodes+16+to+22%29
        self.logger.info("downloadUrl=%s", downloadUrl)

        yield scrapy.Request(url=downloadUrl, callback=self.parseDownloadSubtitlesResp)
    else:
        self.logger.warning("can not find english subtitle node")

def parseDownloadSubtitlesResp(self, response):
    """
    parse download subtitles of youtube video
    :param response:
    :return:
    """
    # for debug
    respUrl = response.url # http://www.yousubtitles.com/download/?url=http%3A%2F%2Fwww.youtube.com%2Fapi%2Ftimedtext%3Fsignature%3D4A2A6F9EA37E8A508D6D3C93B575ADFFAA910E66.40BD74A9A28E3B158532EE17837676F7B11B1180%26xorp%3DTrue%26sparams%3Dasr_langs%252Ccaps%252Ccv%252Cxp%252Cexpire%26asr_langs%3Dde%252Cru%252Cko%252Cja%252Cen%252Cpt%252Cit%252Ces%252Cn

```

```

1%252Cfr%26h1%3Den_US%26key%3Dyttt1%26caps%3Dasr%26expire%3D1520333397%26v%3D23t1f8d2ISs%2
6kind%3Dasr%26lang%3Den&title=Humf+-+Compilation+%28episodes+16+to+22%29
    self.logger.info("respUrl=%s", respUrl)
    decodedUrl = unquote(respUrl) # http://www.yousubtitles.com/download/?url=http://w
ww.youtube.com/api/timedtext?caps=asr&v=7A6y4WPkG5o&expire=1520334242&asr_lang
s=en%2Cfr%2Cja%2Cko%2Cde%2Cpt%2Cit%2Cn1%2Cru%2Ces&hl=en_US&xorp=True&key=yttt1
&signature=26E544FE885C617E1B6AC946C1EE21C1E0C95795.83020B2CBAF8806252387C5AD7BA0879B2
CC1888&sparams=asr_langs%2Ccaps%2Cv%2Cxorp%2Cexpire&kind=asr&lang=en&title
=Humf+-+25++Slow+Down,+Wallace++(full+episode)
    self.logger.info("decodedUrl=%s", decodedUrl)
    foundIdName = re.search(r'([&|?])v=(?P<videoId>[\\w\\-]+).*&title=(?P<
;videoTitle>[&|?])+)', decodedUrl)
    if foundIdName:
        videoId = foundIdName.group("videoId") # '7A6y4WPkG5o'
        videoTitle = foundIdName.group("videoTitle") # Humf+-+25++Slow+Down,+Wallace++
(full+episode)
        subtitleFilename = videoId + "_" + videoTitle

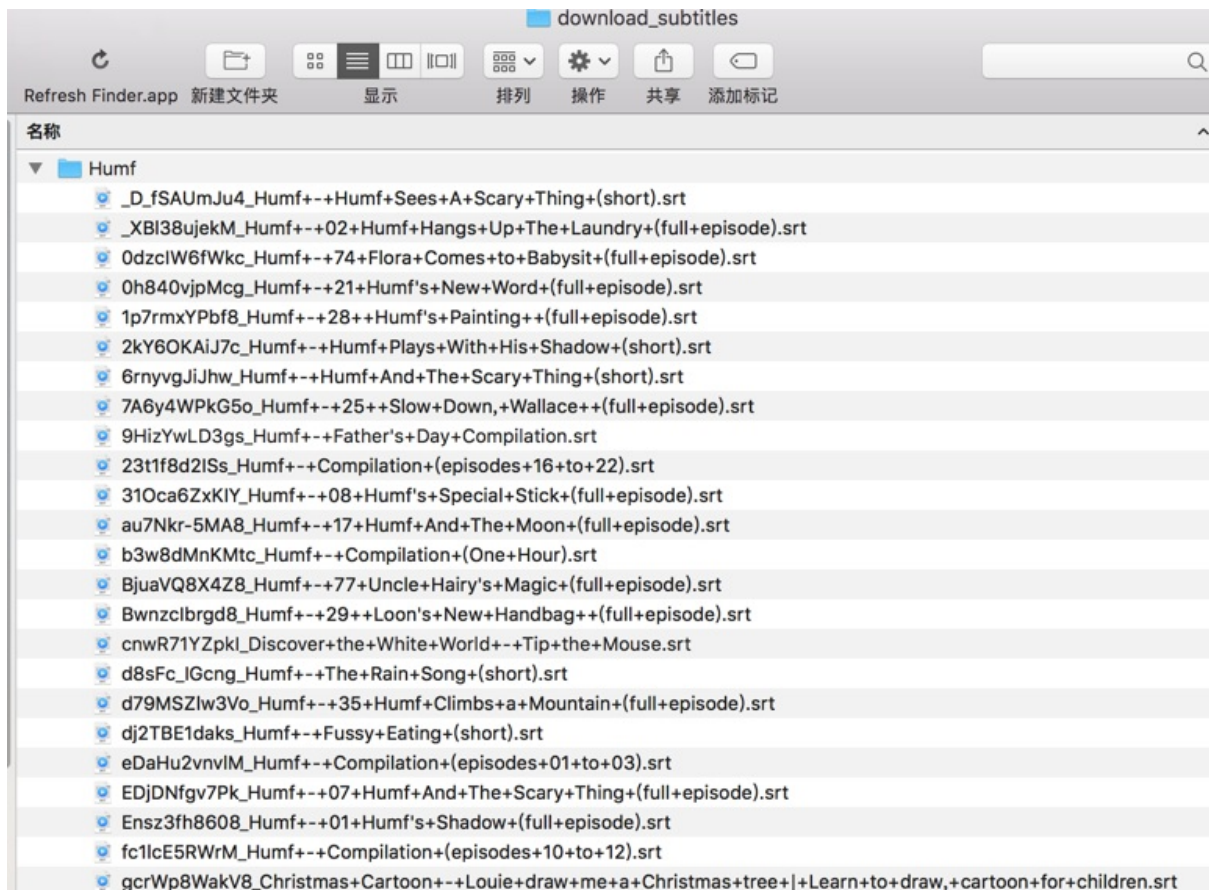
        downloadFolder = "download_subtitles/Humf"
        if not os.path.exists(downloadFolder):
            os.makedirs(downloadFolder)

        self.saveToFile(downloadFolder + "/" + subtitleFilename, response.body, suffix
".srt")
    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.body)
#     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.parseLoadVideoResp, method "POST", meta {"humfGroupTitle" : humfGroupTitle})
```

之后的callback中即可通过response.meta去取值：

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

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新： 2020-08-09 10:19:44

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：

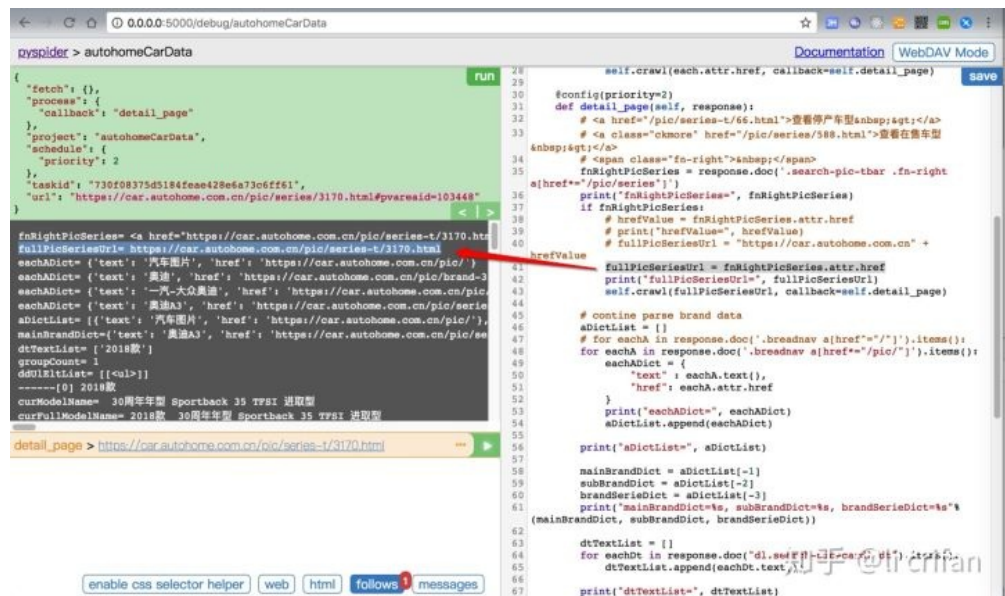
```
response.xpath("//title")  
response.xpath("//title").text()  
response.xpath("//title").extract()  
response.xpath("//title/text()").extract()
```

crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved，powered by Gitbook最后更新： 2020-08-09 10:19:44

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](#)

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2020-08-09 10:19:44

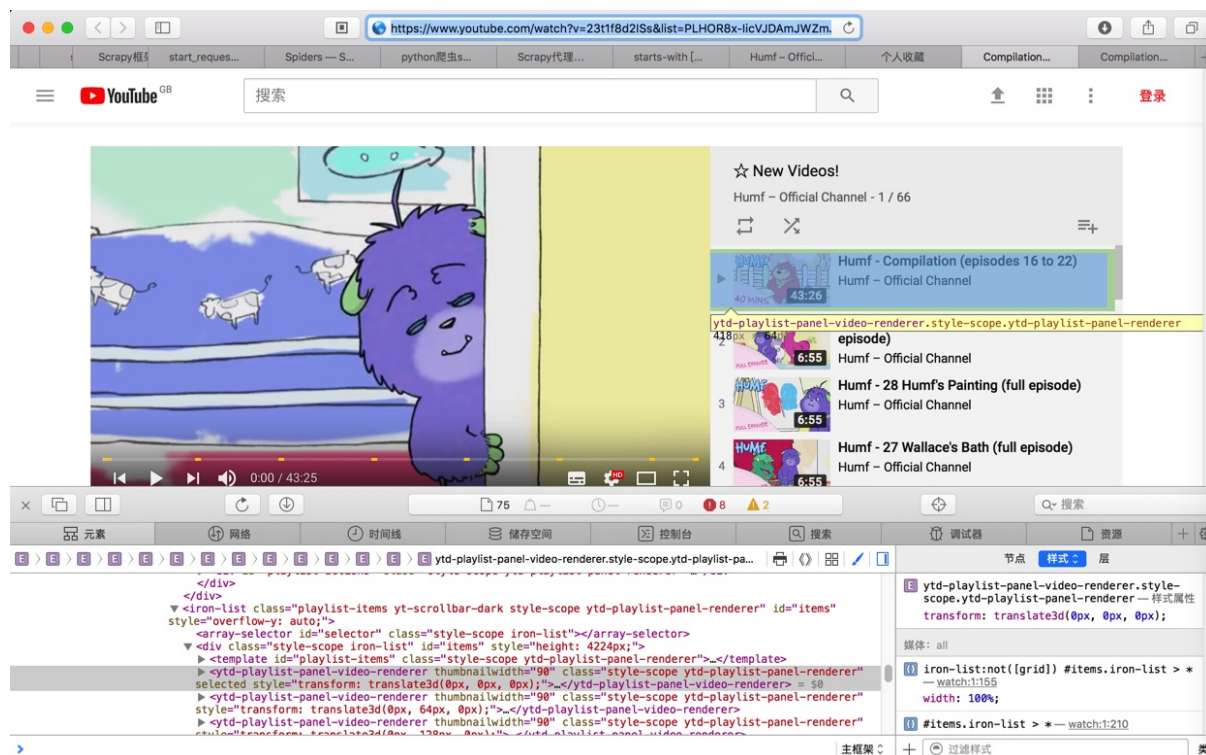
看到和抓到的源码不同

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-IMu1x3ITAlId5>

用 Safari 浏览器打开的效果是:



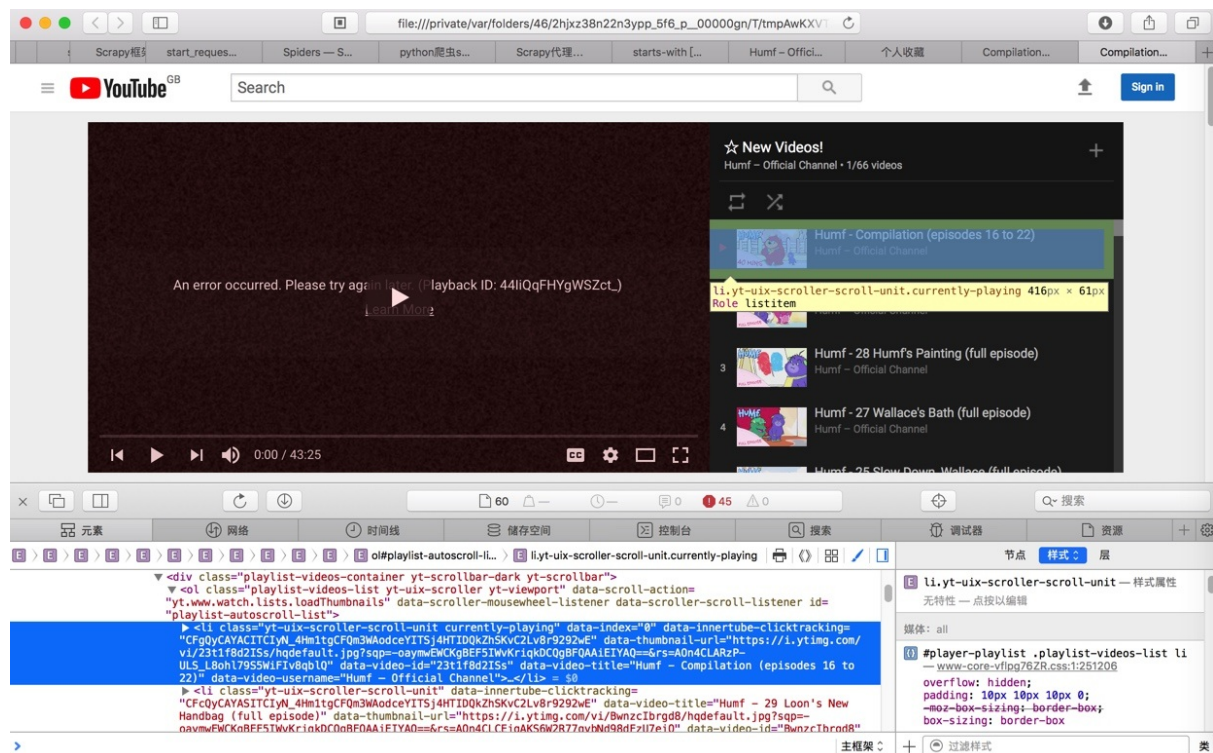
对应要找到元素的 xpath 写法是:

```
//*[@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Scrapy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"' data-u'<ol id="playlist-autoscroll-list" class='>]
>>> response.xpath('//ol[@id="playlist-autoscroll-list"]/li')
```

才能找到要的元素：

[illegible]

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。

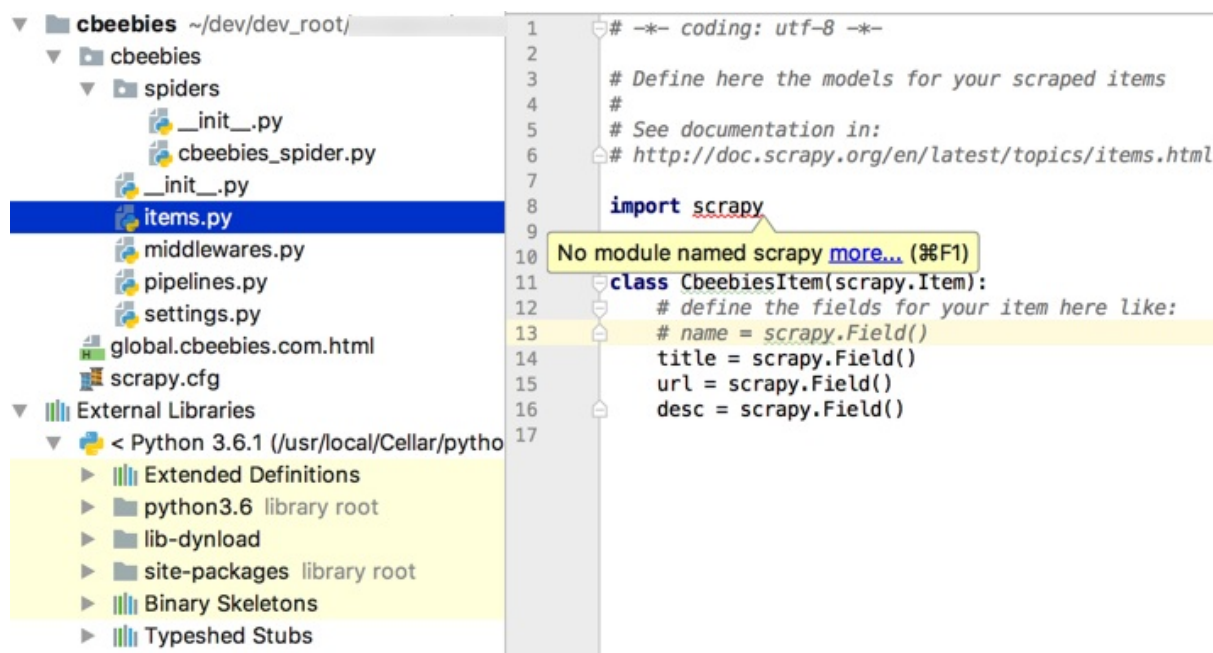
crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved，powered by Gitbook最后更新：2020-08-09 10:19:44

PyCharm调试Scrapy

此处整理用PyCharm调试Scrapy的一些心得。

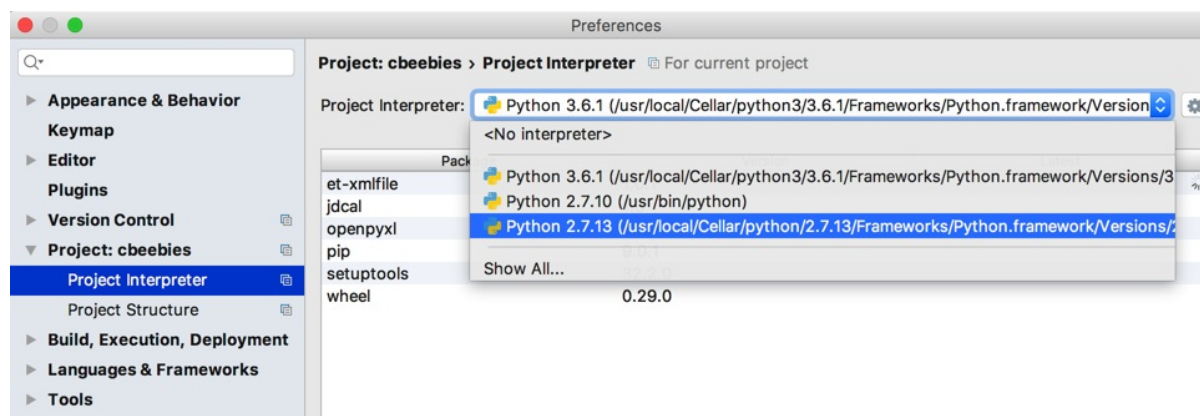
No module named scrapy

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

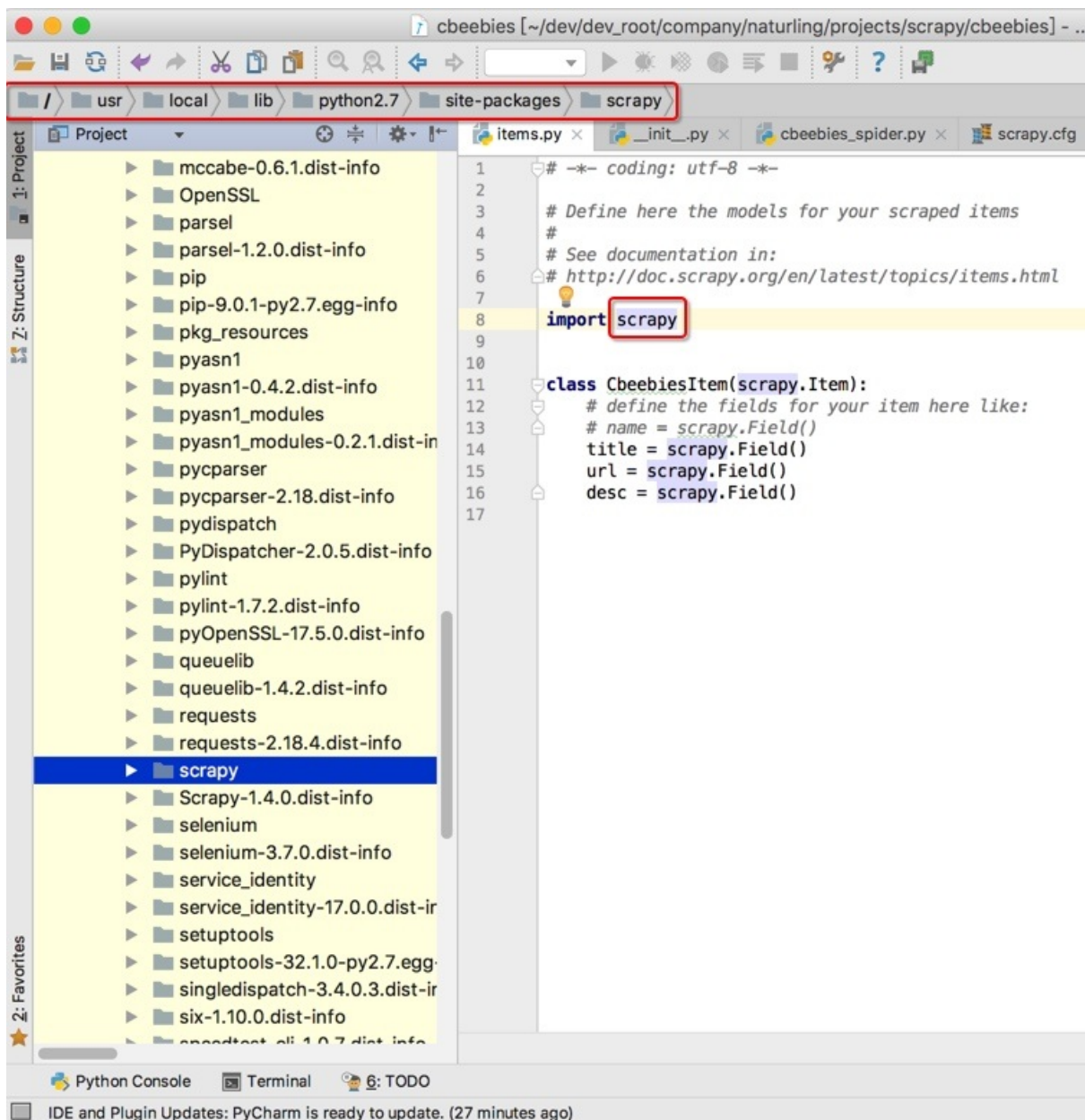


- 原因：Python版本=Python解释器 设置不正确
- 解决办法：设置正确的当前所使用的Python版本

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



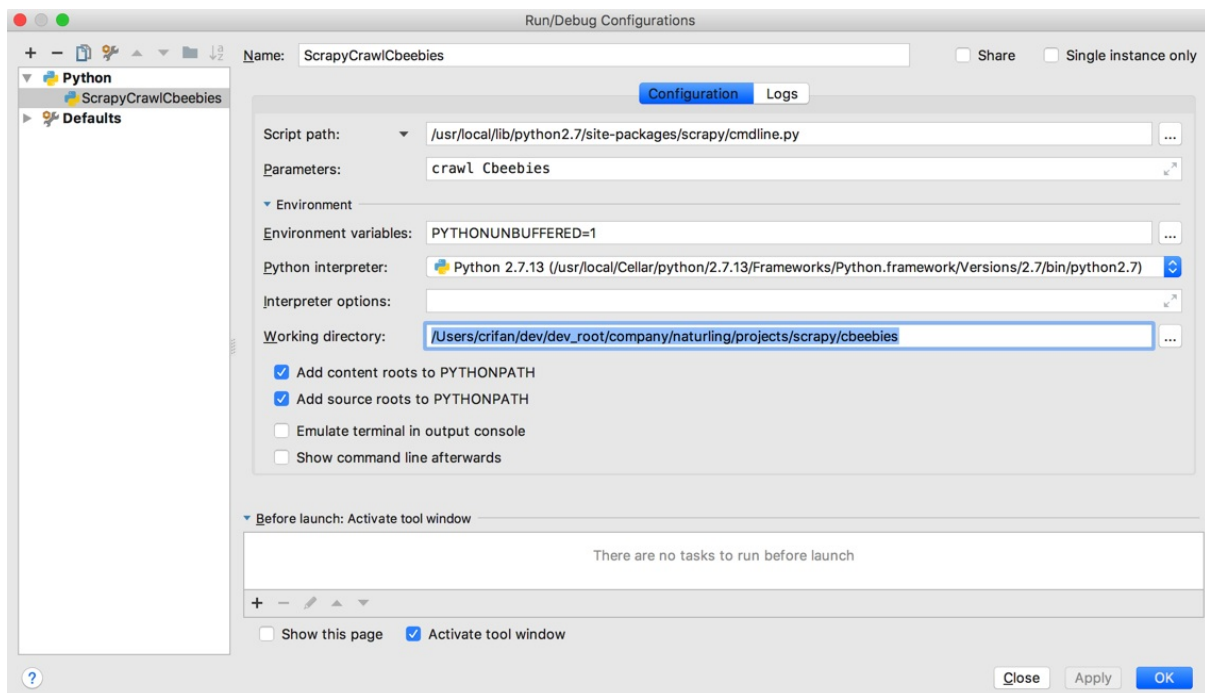
即可正常导入scrapy，以及看到对应源码：



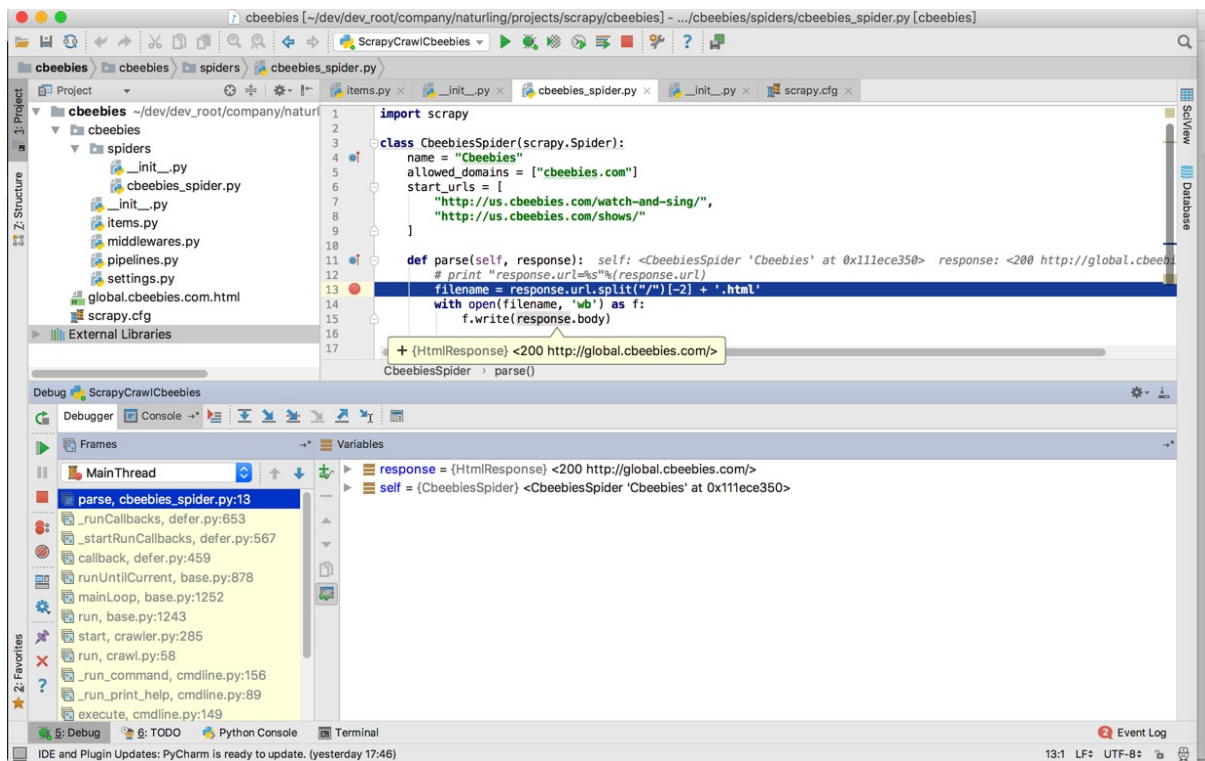
如何用PyCharm调试Scrapy

- 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：



crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved，powered by Gitbook最后更新：2020-08-09 10:19:44

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。

crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新： 2020-08-09 10:19:44

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 (default: 0)
# See http://scrapy.readthedocs.org/en/latest/topics/settings.html#download-delay
# 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-pipeline.html
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": humfGroupTitle})

yield scrapy.Request(url_downloadUrl,
                    callback=self.parseDownloadSubtitlesResp,
                    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。

crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2020-08-09 10:19:44

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>
 - scrapyd 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>

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新： 2020-08-09 10:19:44

附录

下面列出相关参考资料。

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2020-08-09 10:19:44

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 templates
  runspider      Run a self-contained spider (without creating a project)
  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 project directory

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 used

--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 repeated)

--pdb **enable** pdb on failure

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2020-08-09 10:19:44

参考资料

- [使用scrapyd 管理爬虫 · 网络爬虫教程](#)
- [基于scrapy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](#)
-

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2020-08-09 10:19:44