# Scrapy简要教程

## 初始化项目

`scrapy startproject project_name`

## Shell 调试

`scrapy shell url`

e.g. `scrapy shell www.baidu.com`

## 执行爬虫

`scrapy crawl url_name`

## 项目结构

| | |
|---|---|
| 项目名字 | |
| 项目名字 | |
| /spiders | |
| cutomerspider.py | 自定义爬虫文件 |
| __init.py | |
| items.py | 定义数据结构 爬取哪些数据 |
| middlewares.py | 中间件 代理 |
| pipelines.py | 管道 处理下载的数据 |
| settings.py | 配置文件 ua定义 robots协议 开启管道 等 |

## response 属性

- Response.text  #字符串
- Response.body  #二进制数据
- Response.xpath()  #xpath解析
- Response.extract() #提取selector对象的data属性值
- Response.extract_first()  #提取selector列表的第一个属性值

## cutomerspider.py

在 `spiders` 下创建自定义爬虫文件
`scrapy genspider url_name url`
e.g. `scrapy genspider baidu www.baidu.com`

生成文件如下：

```python
import scrapy


class Baidu1Spider(scrapy.Spider):
    name = 'baidu'
    allowed_domains = ['www.baidu.com']
    start_urls = ['http://www.baidu.com/']

    def parse(self, response):
        pass
```

## GET请求

多页面下载

```python
def parse(self, response):
  name = response.xpath('//')
  url = response.xpath('//')
  data = ScrapyNoobItem(name=name, url=url)
  yield data
  for i in range(1, 101):
    url = self.base_url + str(i)
    # scrapy.Request 相当于 requests.get() 请求
    # callback 的值 只需要函数名， 不需要加括号
    yield scrapy.Request(url=url, callback=self.parse)
```

多页面解析逻辑不一致并需要传参时，需要自定义一个新的解析函数，并通过 `meta` 参数传值

```python
def parse(self, response):
    parser = 
response.xpath('//div[@class="co_area2"]/div[@class="co_content8"]//td[1]/a[2]')
    for p in parser:
        name = p.xpath('./text()').extract_first()
        detail_url = 'https://www.dytt8.net' + p.xpath('./@href').extract_first()
        yield scrapy.Request(url=detail_url, callback=self.parse_img, meta={'name': 
name})

def parse_img(self, response):
    img_url = response.xpath('//div[@id="Zoom"]//img/@src').extract_first()
    name = response.meta['name']
    movie = DyttItem(name=name, img_url=img_url)
    yield movie
```

***如果解析不到数据，首先查看 `xpath` 解析是否正确，再确定解析的网址是否在 `allowed_domains` 内***

## POST请求

重写 `start_requests()` 方法

`start_requests()` 的返回值为 `scrapy.FormRequest()`

- url:　　　　请求地址
- headers:　　头信息
- callback:　　回调函数（无括号）
- formdata:　　表单请求数据

```
class BaiduSpider(scrapy.Spider):
    name = 'baidu'
    allowed_domains = ['fanyi.baidu.com']
    # start_urls = ['http://fanyi.baidu.com/sug']

    def start_requests(self):
        url = 'http://fanyi.baidu.com/sug'

        data = {
            'kw': 'test'
        }

        yield scrapy.FormRequest(url=url, formdata=data, callback=self.parse)

    def parse(self, response):
        content = json.loads(response.text)
        print(content)
```

## items.py

自定义要爬取的数据

```
class ScrapyNoobItem(scrapy.Item):
    # define the fields for your item here like:
    name = scrapy.Field()
    url = scrapy.Field()
```

要在 `/spiders` 下的自定义爬虫 `cutomerspider.py` 文件中，引入该 `items`

```
# /spiders/customerspider.py
import scrapy

from scrapy_noob.items import ScrapyNoobItem
```

```python
class BaiduSpider(scrapy.Spider):
    name = 'baidu'
    allowed_domains = ['www.baidu.com']
    start_urls = ['http://www.baidu.com/']

    def parse(self, response):
        name = response.xpath('//')
        url = response.xpath('//')
        data = ScrapyNoobItem(name=name, url=url)
        yield data   # data 将会传入到 pipelines.py 中
```

## middlewares.py

作用：

- 对 `headers` 和 `cookies` 进行更换和处理
- 代理IP
- 对请求进行定制化操作

在`settings.py`中添加开启

```python
# Enable or disable downloader middlewares
# See https://docs.scrapy.org/en/latest/topics/downloader-middleware.html
DOWNLOADER_MIDDLEWARES = {
   'fanyi.middlewares.FanyiDownloaderMiddleware': 543,
}
```

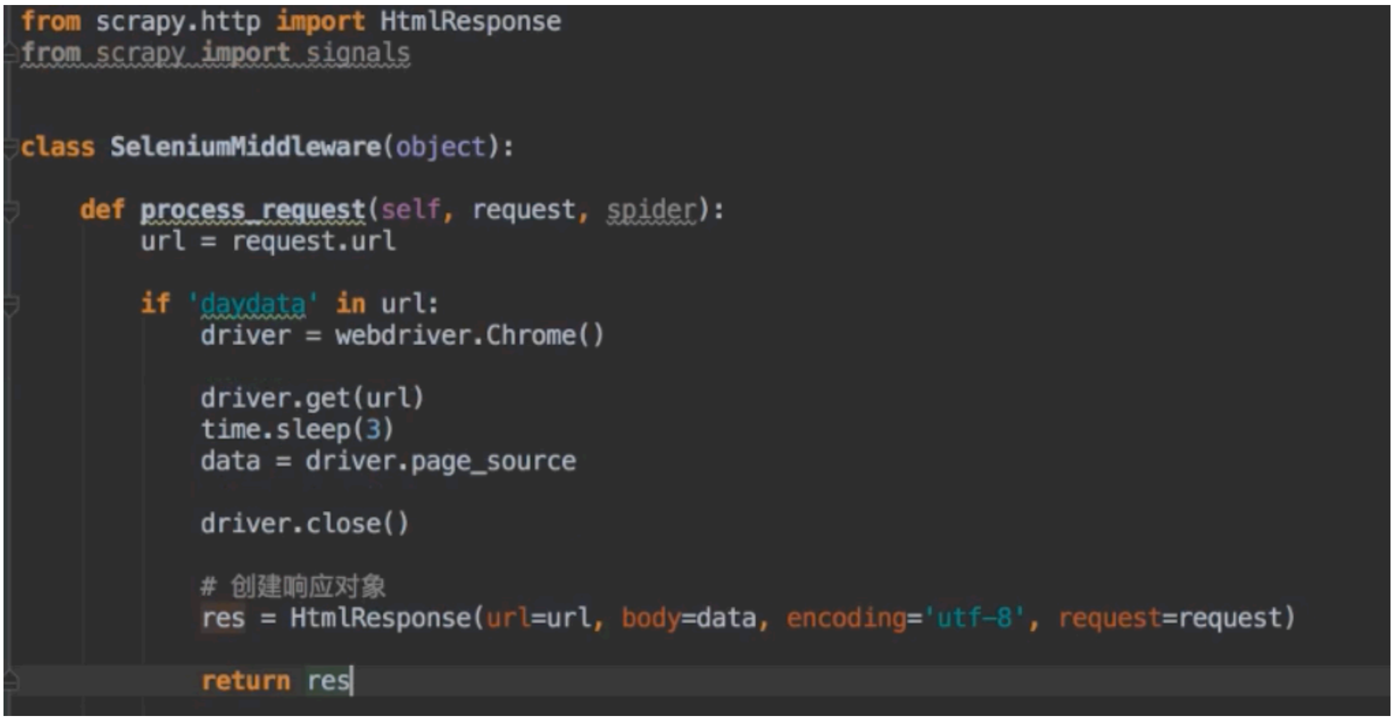
分为下载中间件和爬虫中间件，**通常只编写下载中间件**

对 `headers` 进行更换和处理， 代理IP

```python
def process_request(self, request, spider):
    # Called for each request that goes through the downloader middleware.

    # Must either:
    # - return None: continue processing this request
    # - or return a Response object
    # - or return a Request object
    # - or raise IgnoreRequest: process_exception() methods of
    #   installed downloader middleware will be called
    request.headers['User-Agent'] = ua.random
    request.meta['proxy'] = 'https://10.0.0.0:8888'
    return None
```

搭配 `selenium` 渲染页面



```python
from scrapy.http import HtmlResponse
from scrapy import signals


class SeleniumMiddleware(object):

    def process_request(self, request, spider):
        url = request.url

        if 'daydata' in url:
            driver = webdriver.Chrome()

            driver.get(url)
            time.sleep(3)
            data = driver.page_source

            driver.close()

            # 创建响应对象
            res = HtmlResponse(url=url, body=data, encoding='utf-8', request=request)

            return res
```

## pipelines.py

初始执行函数

```python
def open_spider(self, spider):
  pass
```

结束执行函数

```python
def close_spider(self, spider):
  pass
```

开启多条管道
在`pipelines.py`中增加一个管道类，并在`settings.py`中添加开启

`pipelines.py`

```python
class CustomPipeline:
    def process_item(self, item, spider):
        return item
```

`settings.py`

```
ITEM_PIPELINES = {
    'scrapy_noob.pipelines.ScrapyNoobPipeline': 300,
    'scrapy_noob.pipelines.CustomPipeline': 301,
}
```

***下载目录位置相对于 /spiders 文件夹***

## settings.py

开启管道

```
# Configure item pipelines
# See https://docs.scrapy.org/en/latest/topics/item-pipeline.html
# 数值越小，优先级越高
ITEM_PIPELINES = {
    'scrapy_noob.pipelines.ScrapyNoobPipeline': 300,
}
```

遵守`robots.txt`协议

```
# Obey robots.txt rules
ROBOTSTXT_OBEY = True
```

指定日志信息

```
# 指定日志等级
LOG_LEVEL = 'WARNING'
# 指定日志文件
LOG_FILE = 'logging.log'
```

## 链接提取器

创建文件

`scrapy genspider -t crawl url_name url`

生成文件如下

```
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule
```

```python
class ReadSpider(CrawlSpider):
    name = 'read'
    allowed_domains = ['www.dushu.com']
    start_urls = ['http://www.dushu.com/']

    rules = (
        Rule(LinkExtractor(allow=r'Items/'), callback='parse_item', follow=True),
    )

    def parse_item(self, response):
        item = {}
        #item['domain_id'] = response.xpath('//input[@id="sid"]/@value').get()
        #item['name'] = response.xpath('//div[@id="name"]').get()
        #item['description'] = response.xpath('//div[@id="description"]').get()
        return item
```

`LinkExtractor`的参数

- allow=()                    #符合正则表达式的提取
- deny=()                     #符合正则表达式的拒绝
- allow_domains=()        #符合域名的提取
- deny_domains=()         #符合域名的拒绝
- restrict_xpaths=()       #符合xpath规则的提取
- restrict_css=()            #符合css规则的提取
- callback=''                  #返回的解析规则，**只能是字符串形式**
- follow=True/False       #是否跟进（继续按照链接规则继续提取）