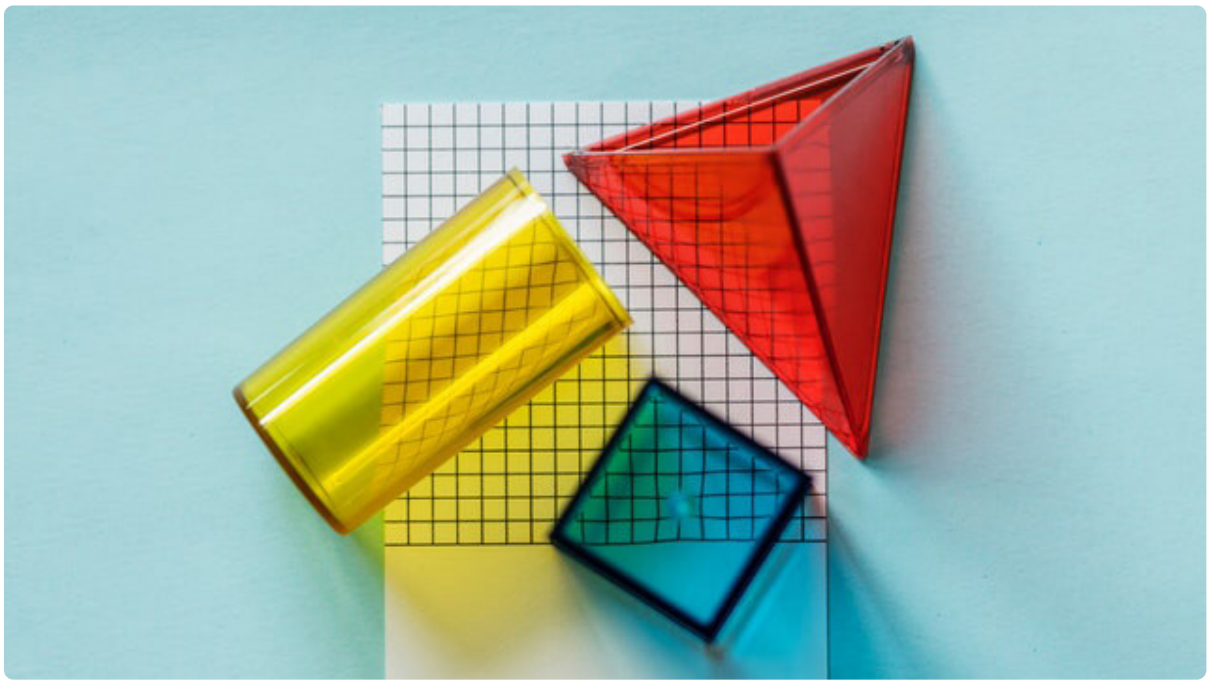
专栏福利Scrapy-plus

更新时间：2019-07-10 14:36:47



> 学习知识要善于思考，思考，再思考。
>
> —— 爱因斯

我将本专栏中用到的所有具有通用性的类收集并做成了一个 python 工具包，本节将介绍如何让`scrapy-plus`加速你的爬虫项目开发。

scrapy-plus是一个scrapy的辅助扩展工具包，它将本专栏所采用的的示例中可以被重用的代码都集成其中，便于读者们在以后的爬虫项目中快速开发。

安装

创建scrapy爬虫项目后并激活python虚环境，执行以下的安装指令:

```
$ pip install scrapy_plus
```

过滤器

根据本专栏第4章网易爬虫的优化——大规模数据处理技术中所介绍的Redis去重过滤器与高效的布隆过滤器的内容进行抽象与优化。

所有的过滤器都放置于`scrapy_plus.dupefilters`下。

Redis 去重过滤器

`RedisDupeFilter`的模块位置：

```
scrapy_plus.dupefilters.RedisDupeFilter
```

基于Redis使用`Set`存储曾访问过的URL。

使用方法

首先要安装Redis或者起动一个Redis的容器，具体做法请参考本专栏的第4章第2节去重处理——高性能爬虫调优技术中的Redis安装介绍。

`RedisDupeFilter`的使用极其简单，只需要在配置文件中做以相应的修改即可。具体做法是在`settings`文件内引入以下的内容:

```
# 覆盖原有的去重过滤器
DUPEFILTER_CLASS = 'scrapy_plus.dupefilters.RedisDupeFilter'
REDIS_PORT = 6379                       # REDIS服务器端口
REDIS_HOST = '127.0.0.1'                # REDIS服务器地址
REDIS_DB = 0                            # 数据库名
```

默认配置

```
REDIS_PORT = 6379                       # REDIS服务器端口
REDIS_HOST = '127.0.0.1'                # REDIS服务器地址
REDIS_DB = 0                            # 数据库名
```

如果你不修改Redis的安装配置可以只在`settings.py`文件中加入以下这行即可：

```
DUPEFILTER_CLASS = 'scrapy_plus.dupefilters.RedisDupeFilter'
```

Redis 布隆去重过滤器

这是基于`RedisDupeFilter`并加入布隆算法后的最高效的去重过滤器。

`RedisBloomDupeFilter`的模块位置：

```
scrapy_plus.dupefilters.RedisBloomDupeFilter
```

使用方法

使用方法与`RedisDupeFilter`相同在`settings`文件内引入以下的内容:

```
# 覆盖原有的去重过滤器
DUPEFILTER_CLASS = 'scrapy_plus.dupefilters.RedisBloomDupeFilter'
REDIS_PORT = 6379                       # REDIS服务器端口
REDIS_HOST = '127.0.0.1'                # REDIS服务器地址
REDIS_DB = 0                            # 数据库名
```

默认配置

```
REDIS_PORT = 6379                       # REDIS服务器端口
REDIS_HOST = '127.0.0.1'                # REDIS服务器地址
REDIS_DB = 0                            # 数据库名
BLOOMFILTER_REDIS_KEY = 'bloomfilter'   # 去重键名
BLOOMFILTER_BLOCK_NUMBER = 1            # 块大小
```

与`RedisDupeFilter`不同的是`RedisBloomDupeFilter`增加了两个配置项:

- `BLOOMFILTER_REDIS_KEY` - 设置Redis中去重键的名称。
- `BLOOMFILTER_BLOCK_NUMBER` - 设置布隆算法块的大小。

这两个选项推荐使用默认值，也可以根据你项目的实际情况进行调节。

中间件

Scrapy+的中间件放置于`scrapy_plus.middlewares`包内。

自登录中间件

这是一个通用的中间件，可以应用于所有能提供登陆URL的网站，`LoginMiddleWare`会判断是否已登录，如果已登录则不会进行重复登录。

使用这个中间件时需要在`settings.py`配置文件中的`COOKIES_ENABLED`打开

例如在网页中找到一个`<form>`元素：

```
<form method="post" action="/login">
 <input name="username" />
 <input type="password" />
</form>
```

那就完全可以应用`LoginMiddleware`完成自动登录。

模块位置

```
scrapy_plus.middlewares.LoginMiddleWare
```

以下是自动登录中间件的配置：

```
COOKIES_ENABLED=True
LOGIN_URL = '网站登录地址'
LOGIN_USR = '用户名'
LOGIN_PWD = '密码'
LOGIN_USR_FIELD = '用户名input元素名称(name)'
LOGIN_PWD_FIELD = '密码input元素名称(name)'
DOWNLOADER_MIDDLEWARES = {
    'scrapyplus.middlewares.LoginMiddleWare': 330
}
```

花瓣网专用中间件

这是一个基于Chrome无头浏览器，可以自动登录花瓣网并能正确渲染花瓣网javascript页面的中间件。

模块位置

```
scrapy_plus.middlewares.HuabanMiddleware
```

使用方法

首先，你需要安装chromedriver，具体做法请参考本专栏第6章第4节用Chrome无头浏览器处理js网页关于安装chromedriver的相关内容。

其次，你需要拥有一个花瓣网的注册账号。

最后，在`settings.py`配置文件内加入以下的配置项：

```
SELENIUM_TIMEOUT = 30 # 设置页面打开的超时秒数
CHROMEDRIVER = "/path/to/chrome" # Chrome浏览器驱动地址
# 以下的macOS上示例:
# CHROMEDRIVER = "/usr/local/Caskroom/chromedriver/75.0.3770.90/chromedriver"
DOWNLOADER_MIDDLEWARES = {
    'scrapyplus.middlewares.HuabanMiddleware': 100
}
HUABAN_USR="您在花瓣网上的用户名"
HUABAN_PWD="你在花瓣网上注册的用户密码"
```

有了这个中间件你就可以像写普通蜘蛛一样来编写花瓣蜘蛛的逻辑。

Chrome通用中间件

Chrome 无头浏览器仿真中间件。让爬虫用Chrome来访问目标URL，完美解决富JS页面的问题。

> 但仅可以对不需要进行登录的网站应用此中间件。

```
SELENIUM_TIMEOUT = 30 # 设置页面打开的超时秒数
CHROMEDRIVER = "/path/to/chrome" # Chrome浏览器驱动地址
DOWNLOADER_MIDDLEWARES = {
    'scrapyplus.middlewares.ChromeMiddleware': 800
}
```

Splash渲染中间件

基于scrappy_splash进行扩展，简化splash的用法，使蜘蛛可以不发出`SplashRequest`就能使用splash进行javascript页面的渲染。

模块位置

```
scrapy_plus.middlewares.SplashSpiderMiddleware
```

Splash 中间件，可将请求转发至指定的Splash服务，使蜘蛛具有浏览器仿真功能。

```
WAIT_FOR_ELEMENT = "选择器" # 等待该元素被加载成功才认为页面加载完成
DOWNLOADER_MIDDLEWARES = {
}

DUPEFILTER_CLASS = 'scrapy_splash.SplashAwareDupeFilter'

DOWNLOADER_MIDDLEWARES = {
    'scrapy_splash.SplashCookiesMiddleware': 723,
    'scrapy_splash.SplashMiddleware': 725,
    'scrapy_plus.middlewares.SplashSpiderMiddleware': 800,
    'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware': 810
}

SPIDER_MIDDLEWARES = {
    'scrapy_splash.SplashDeduplicateArgsMiddleware': 100,
}
```

随机UA中间件

为爬取请求随机分配UA。具体原理请参考本专栏第5章第5节反爬初步之客户端仿真。

> 使用前需要将`scrapy.downloadermiddlewares.useragent.UserAgentMiddleware`禁用。

模块位置

```
scrapyplus.middlewares.RandomUserAgentMiddleware
```

使用方法

在`settings.py`添加以下的配置：

```
DOWNLOADER_MIDDLEWARES = {
    'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,
    'scrapyplus.middlewares.RandomUserAgentMiddleware': 500
}

## 可随机增加更多的UA，中间件会进行自动随机选择
USER_AGENTS = [
    'Mozilla/5.0 (Windows NT 10.0; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0',
    'Mozilla/5.0 (Linux; U; Android 2.2) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1',
    'Mozilla/5.0 (Windows NT 5.1; rv:7.0.1) Gecko/20100101 Firefox/7.0.1',
    'Mozilla/5.0 (Linux; Android 6.0.1; SM-G532G Build/MMB29T) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.83 M
obile Safari/537.36',
    'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/604.5.6 (KHTML, like Gecko) Version/11.0.3 Safari/604.5.6'
]
```

随机代理中间件

当爬虫向目标网站发出请求时在配置文件的`HTTP_PROXIES`列表中随机选择一个地址。

模块位置

```
scrapyplus.middlewares.RandomProxyMiddleware
```

在`settings.py`文件内添加以下的配置:

```
DOWNLOADER_MIDDLEWARES = {
    'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware': None,
    'scrapyplus.middlewares.RandomProxyMiddleware': 750
}
# 以下为代理列表
HTTP_PROXIES=[
'203.11.43.22:8080'
]
```

Tor 中间件

模块位置

```
scrapyplus.middlewares.TorProxyMiddleware
```

洋葱头代理中间件,让你的蜘蛛不停地更换IP地址,化身万千。需要先安装 tor 与 privoxy 具体配置方法请参考《虫术——python绝技》

```
# Tor代理
TOR_PROXY = 'http://127.0.0.1:8118'  # 8118是Privoxy的默认代理端口
TOR_CTRL_PORT = 9051
TOR_PASSWORD = 'mypassword'
TOR_CHANGE_AFTER_TIMES = 50 # 在发出多少次请求之后更换IP地址。
```

管道

Scrapy+的中间件放置于`scrapy_plus.pipelines`包内。

MongoDB数据存储管道

将`Item`数据项目直接写入到MongoDB。使用此管道前需要先安装MongoDB或起动MongoDB的Docker实例。

模块位置

```
scrapy_plus.piplines.MongoDBPipeline
```

可以将Item直接写入MongoDB数据库中。

使用方法

在`settings.py`文件内加入以下配置项：

```
ITEM_PIPELINES = {'scrapy_plus.pipelines.MongoDBPipeline':2}

MONGODB_SERVER = "localhost"    # mongodb服务器地址
MONGODB_PORT = 27017            # mongodb服务端口
MONGODB_DB = "数据库名"          # 数据库名
MONGODB_COLLECTION = "表名"     # 表名
```

可支持阿里云的OSS图片管道

scrapy 搭载的图片管道`ImagesPipeline`只能将图片保存到本地或者S3、Google这些中国不能用的云储存，在第6章第5节将花瓣爬虫采访的图片存储于阿里云一节中我就介绍过这个管道的实现过程与原理。

模块位置

```
scrapy_plus.pipelines.ImagesPipeline
```

使用方法

需要在配置文件中`settings.py`加入以下的配置项

```
IMAGE_STORE='oss://<Bucket名称>.<外网EndPoint>.aliyuncs.com/<子目录>'
OSS_ACCESS_KEY = 'OSS上访问公钥'
OSS_ACCESS_SECRET = 'OSS的访问私钥'
# 在Item中存储目标图片下载地址的字段名
IMAGES_URLS_FIELD = 'img_urls'
# 当下载完后将下载结果对象写入到Item对象的字段名
IMAGES_RESULT_FIELD = 'img_files'
# 加载图片存储管道
ITEM_PIPELINES = {
  'huaban.pipelines.images.ImagesPipeline': 2
}
```

存储端

SQL存储端

将数据一次性地写入SQL数据库存储端。可以支持sqlite, postgresql和mysql多种SQL类型的数据库。

模块位置

```
scrapy_plus.extensions.SQLFeedStorage
```

使用方法

`SQLFeedStorage`的使用方法有点复杂，在使用前需要基于SQLAlchemy进行数据建模，具体方法可以参考本专栏第5章第4节基于SQL的数据导出机制。

在`settings.py`需要加入以下的配置项

```
# 数据存储
ORM_MODULE = 'movies.entities'
ORM_METABASE = 'Base'
ORM_ENTITY = 'Movie'

FEED_FORMAT = 'entity'
FEED_EXPORTERS = {
    'entity': 'scrapyplus.extensions.SQLItemExporter'
}

FEED_URI = 'dialect+driver://username:password@host:port/database'  # 默认后端存储文件的名称
FEED_STORAGES = {
    'sqlite': 'scrapyplus.extensions.SQLFeedStorage',
    'postgresql': 'scrapyplus.extensions.SQLFeedStorage',
    'mysql': 'scrapyplus.extensions.SQLFeedStorage'
}
```

输入/输出处理器

Scrapy+的中间件放置于`scrapy_plus.processors`包内。我在本专栏第五章第一、第二节都有介绍过输入输出处理器的用法以及如何开发自定义的输入/输出处理器。在Scrapy+中则更进一步提供了8个最常用的输入/输出处理器。他们分别是:

- `Text` - 提取并输出文字字符串
- `Number` - 提取并输出数字格式字符串
- `Price` - 提取出价格格式的字符串
- `Date` - 提取日期格式的字符串
- `Url` - 提取并输出URL格式字符串
- `Image` - 提取并输出图片地址格式字符串
- `SafeHtml` - 提取并输出移除所有可被执行的HTML代码后的HTML格式化字符串
- `Regex` - 提取并输出符合正规表达式的字符串

模块位置

```
scrapy_plus.processors
```

蜘蛛

Scrapy+将本专栏中两个蜘蛛放到了`scrapy_plus.spiders`包内，方便读者可以在不编写蜘蛛的情况下只配`settings.py`就可以直接使用。

Scrapy+总共提供了三个蜘蛛类，分别是:

类名	蜘蛛名	说明
`BookSpider`	`'doubanbook'`	用于拉取豆瓣图书的专用蜘蛛，该蜘蛛将会返回`scrapy_plus.items.BookItem`的数据项对象。
`NeteaseSpider`	`'netease'`	用于拉取网易新闻的专用蜘蛛，该蜘蛛将会返回`scrapy_plus.items.NewsItem`数据项对象
`TaobaoSpider`	`'taobao'`	用于拉取淘宝网搜索页面的专用蜘蛛，该蜘蛛将会返回`scrapy_plus.items.ProductItem`数据项对象

它们的使方法如下，在`settings.py`内修改

```
BOT_NAME = '蜘蛛名'
SPIDER_MODULES = ['scrapy_plus.spiders']  # 指定爬虫类所在的包
NEWSPIDER_MODULE = 'scrapy_plus.spiders'  # 指定爬虫模块
```

淘宝蜘蛛是在《Python绝技——虫术》一书中的一个经典例子，要使用这个蜘蛛需要使用慢速爬虫与自动登录的Chrome中间件，同时这个蜘蛛需要被继承并重新实现`gen_keywords`方法后才能使用：

```
from scrapy_plus.spider import TaobaoSpider

class MyTaobaoSpider(TaobaoSpider):
  def gen_keywords(self):
        return ["小米","红酒"]
```

这个蜘蛛是利用淘宝搜索页进行爬取的，所以你需要返回需要搜索的关键字，该蜘蛛就会对这些关键字下搜索出来的产品进行爬取。

小结

Scrapy+还将会继续更新，如果你有可以促进此项目的蜘蛛或其它的扩展可以在github上将需求request我。这样将我们共同知识分享给更多有需要的朋友。用爬虫技术打破现在国内数据封闭、各成孤岛的困境，数据共享才能促进更多的可能，Enjoy。

> 注：你可以去 GitHub 获取源代码。

Scrapy架构总结与经验补充

写在最后

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论