kaokao2011 VIP
2017-01-08 10:41 更新 阅读 2次

scrapy图片抓取

1.ImagePipeline

Scrapy用ImagesPipeline类提供一种方便的方式来下载和存储图片。需要PIL库支持。

2.主要特征

- 将下载图片转换成通用的JPG和RGB格式
- 避免重复下载
- 缩略图生成
- 图片大小过滤

3.工作流程

- 爬取一个Item，将图片的URLs放入 `image_urls` 字段
- 从 `Spider` 返回的Item，传递到 `Item Pipeline`
  当Item传递到ImagePipeline，将调用Scrapy 调度器和下载器完成 `image_urls` 中的url的调度和下载。ImagePipeline会自动高优先级抓取这些url，于此同时，item会被锁定直到图片抓取完毕才被解锁。
  图片下载成功结束后，图片下载路径、url和校验和等信息会被填充到images字段中。

3.1 定义Item

使用 `ImagePipeline` 需要在配置文件中配置ITEMPIPELINES属性，并定义一个Item包含imageurls和images字段

```
1.  class ImageItem(Item):
2.      image_urls = Field()
3.      images = Field()
4.      image_paths = Field()
```

3.2 设置条件和属性settings.py

```
ITEM_PIPELINES = ['demo.pipelines.MyImagesPipeline']  # ImagePipel
IMAGES_STORE = 'D:\\dev\\python\\scrapy\\demo\\img'   # 图片存储路径
IMAGES_EXPIRES = 90                                   # 过期天数
IMAGES_MIN_HEIGHT = 100                               # 图片的最小高
IMAGES_MIN_WIDTH = 100                                # 图片的最小宽
# 图片的尺寸小于IMAGES_MIN_WIDTH*IMAGES_MIN_HEIGHT的图片都会被过滤
```

3.3 图片爬虫ImageSpider

```
from scrapy.spider import BaseSpider
from scrapy.selector import HtmlXPathSelector
from demo.items import ImageItem

class MyImageSpider(BaseSpider):
    name = "image_spider"
    allowed_domains = ["http://topit.me/"]
    start_urls = [
        "http://topit.me/",
    ]

    def parse(self, response):
        hxs = HtmlXPathSelector(response)
        imgs = hxs.select('//img/@src').extract()
        item = ImageItem()
        item['image_urls']=imgs
        return item
```

3.4 ImagePipeline

需要在自定义的ImagePipeline类中重载的方法： `get_media_requests(item, info)` 和 `item_completed(results, items, info)`

正如工作流程所示，Pipeline将从item中获取图片的URLs并下载它们，所以必须重载 `get_media_requests` ，并返回一个 `Request` 对象，这些请求对象将被Pipeline处理，当完成下载后，结果将发送到 `item_completed` 方法，这些结果为一个二元组的list，每个元祖的包含 `(success, image_info_or_failure)` 。

- `success` ：boolean值， `true` 表示成功下载
- `image_info_or_error` ： 如果success=true， `image_info_or_error` 词典包含以下键值对。失败则包含一些出错信息。
    - `url` ： 原始URL

- path ： 本地存储路径
- checksum ： 校验码

```
from scrapy.contrib.pipeline.images import ImagesPipeline
from scrapy.exceptions import DropItem
from scrapy.http import Request

class MyImagesPipeline(ImagesPipeline):

    def get_media_requests(self, item, info):
        for image_url in item['image_urls']:
            yield Request(image_url)


    def item_completed(self, results, item, info):
        image_paths = [x['path'] for ok, x in results if ok]
        if not image_paths:
            raise DropItem("Item contains no images")
        item['image_paths'] = image_paths
        return item
```

运行结果

```
$ scrapy crawl image_spider
.....
2017-01-06 16:49:48+0800 [image_spider] DEBUG: Scraped from <200 ht
    {'image_paths': ['full/4285ec809413767e8dd5daf4a57fbfe97d964e2e
                     'full/b8088f68ba1d569c96b04a3664e115d4833544be
                     ....
                     'full/97f2272a06191cda15abda57e03ec29eb5c5195
                     'full/9afc3ed6b6cf8259b4065a0d2d79f49b6670c51b
     'image_urls': [u'http://img.topit.me/logo.gif',
                    u'http://fe.topit.me/e/a3/43/117387157785b43a3e
                    ...
                    u'http://i.topit.me/9/v/3LOGi2v9m.jpg',
                    u'http://www.topit.me/img/link/mobile.jpg']}
2017-01-06 16:49:48+0800 [image_spider] INFO: Closing spider (finis
....
```