专栏：016：功能强大的“图片下载器”

E - 爬虫技术 爬虫知识

> **用理工科思维看待这个世界**

系列爬虫专栏

初学者，尽力实现最小化学习系统

如何实现项目图片的下载

0：学习理念

- 推荐阅读
  简书：学习方法论
  我觉得对我有帮助，多问自己为什么从来不是什么坏毛病。
- 学习理念

> 作为初学者，独自在摸索中的过程中，往往会遇到各种各样的问题，
> **第一遍**的学习往往就算呈现的是正确答案，往往也不能全部理解，这歌层次需要知道：是什么？；
> **第二遍**的学习需要知道：怎么做？；
> **第三遍**的学习需要知道：如何实现已知的？；
> **第四步**的学习需要知道：如何实现自己的？。
>
> ---
>
> 实现了自己的这步是实现最小可行性系统的关键，但往往容易陷入误区，**错把最后一步的操作当做完全正确的答案**，为避免陷入误区，应该在实现了最小可行性系统上，**再次查阅最接近正确答案的文档，尤其是自学的过程中，推荐阅读官方文档**
>
> ---
>
> 写你明白，差不多也就没学明白；
> 写的自己明白，差不多只学到了7成；
> 写的别人都能明白，差不多学到了8.5成；
> 剩下的是知识盲区。需要持续不断的精进。

- 学习动机
  某动漫爱好者知道我会爬虫，想要我给写个程序抓取某网站图片。当然我不可能错过这个**装X**的机会。所以就使用多线程实现了网页图片链接的下载，总共6万个左右。存在很大的bug，时间紧，就草草结束。后来回过头想要使用Scrapy框架实现，于是有了你看到的这篇文章。

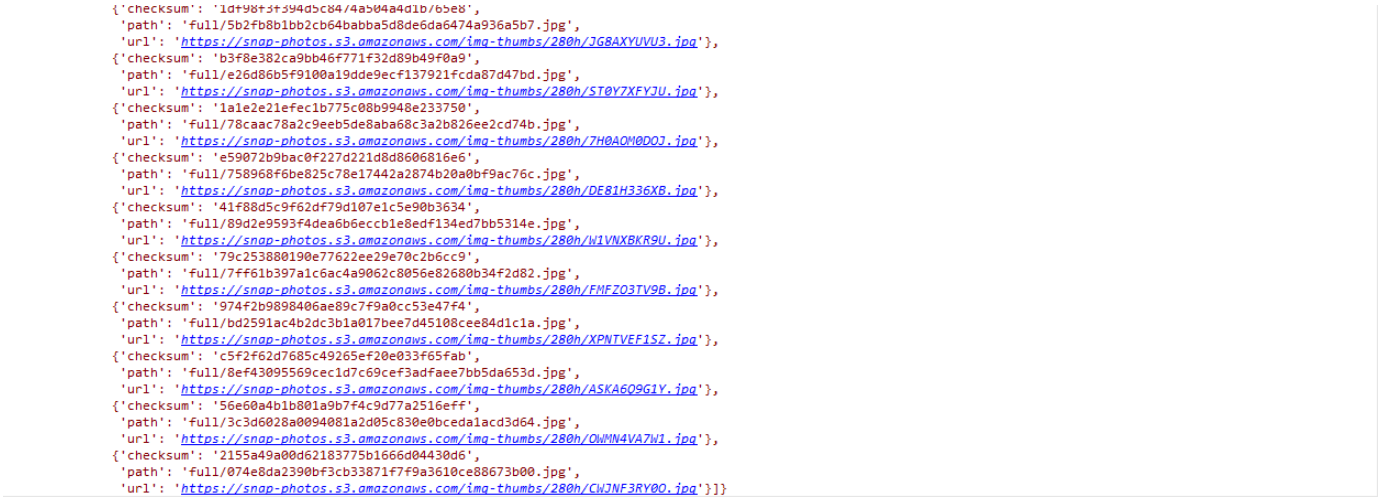
1：原理分解

使用Scrapy的ImagePipeline类提供的一种方便的方式来下载和存储图片，需要PIL库的支持，图片管道，在 ImagesPipeline 类中实现，提供了一个方便并具有额外特性的方法，来下载并本地存储图片:

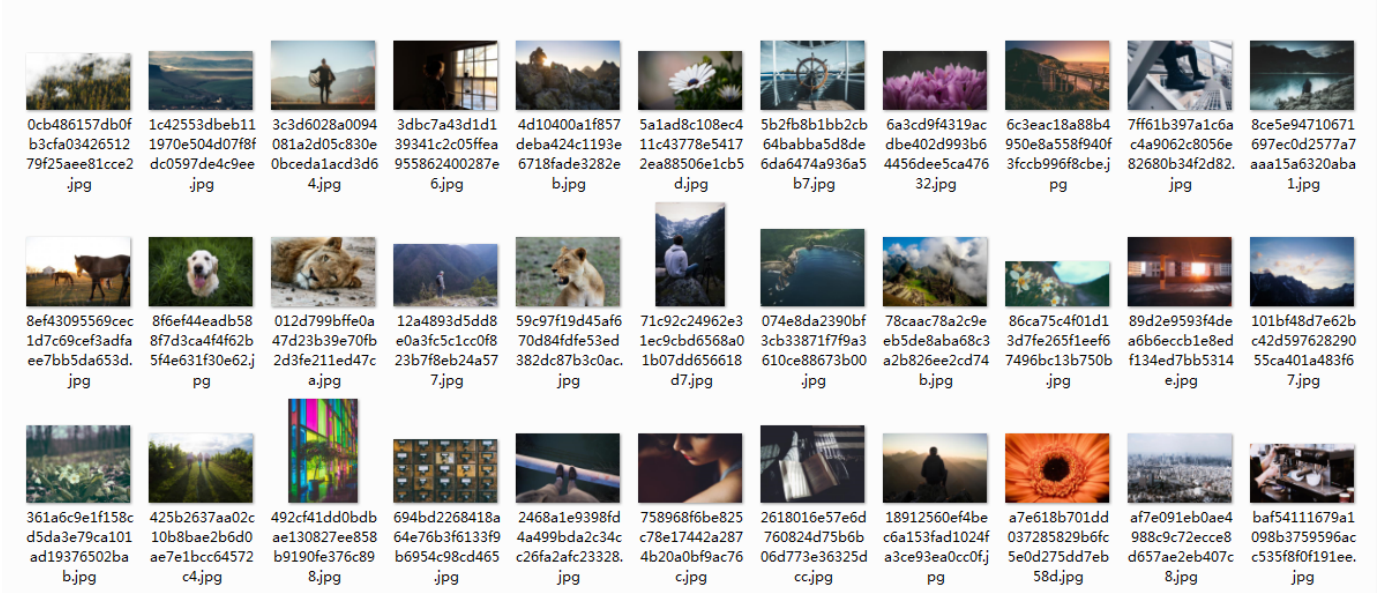
- 主要特征：(可以实现对图片进行怎样的操作)
    - 转换格式
    - 避免重复下载
    - 缩略图下载
    - 指定过滤大小的图片

- 工作流程：(ImagesPipeline类是如何实现图片下载的)
    - Scrapy 爬取的大致步骤是：items.py 设置抓取目标；Spiders/ 实现抓取的代码；pipelines.py 实现对抓取内容的处理
    - 爬取一个Item，将图片的链接放入`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。

如图示：下载成功界面显示：



最终效果：



2：实际操演

- 目标网站
  是它，是它，就是它
  网站采用了异步加载，那就抓取一页先好了，具体的异步加载处理以后写
- 图片`url`的xpath：首页存在30张图片
  `//div[@class="photo-item"]//img/@src`

- `items.py`文件: 定义Item

```
class ImagesItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    image_urls = scrapy.Field()
    images = scrapy.Field()
    image_paths = scrapy.Field()
    pass
```

- `spider/image_spider.py`文件：爬取规则

```
# -*- coding:utf-8 -*-
from scrapy.spiders import CrawlSpider,Rule
from images.items import ImagesItem
from scrapy.selector import Selector

class Download(CrawlSpider):
    name = "image"
    allowed_domains = ["https://stocksnap.io/"]
    start_urls = ["https://stocksnap.io/",]

    def parse(self, response):
        print(response)
        hxs = Selector(response)
        imgs = hxs.xpath('//div[@class="photo-item"]//img/@src').extract()
        item = ImagesItem()
        item['image_urls']=imgs
        return item
```

- 设置`settings.py`
  可以设置：
  - 开启图片管道：`ITEM_PIPELINES = {'scrapy.contrib.pipeline.images.ImagesPipeline': 1}`
  - 存储路径：`IMAGES_STORE = '/path/to/valid/dir'`
  - 还可以设置一些图片失效：`IMAGES_EXPIRES = 90`；缩略图生成：`需要设置IMAGES_THUMBS 字典`,这时会创建缩略图格式的文件夹`<IMAGES_STORE>/thumbs/<size_name>/<image_id>.jpg`；设置过滤小图片`IMAGES_MIN_HEIGHT,IMAGES_MIN_WIDTH`

```
ITEM_PIPELINES = {'scrapy.contrib.pipeline.images.ImagesPipeline': 1} # 开启图片管道
IMAGES_STORE=r"C:\Users\Wuxiaoshen\Desktop\history\tupian"# 存储路径
IMAGES_EXPIRES = 90 # 图片失效日期
```

- 实现定制图片管道
  主要处理的是：ImagePipeline类下的`get_media_requests(item, info)` 和`item_completed(results, items, info)`方法

“

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`url`：原始URL `path`：本地存储路径`checksum`：校验码。失败则包含一些出错信息。

”

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

class ImagesPipeline(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
```

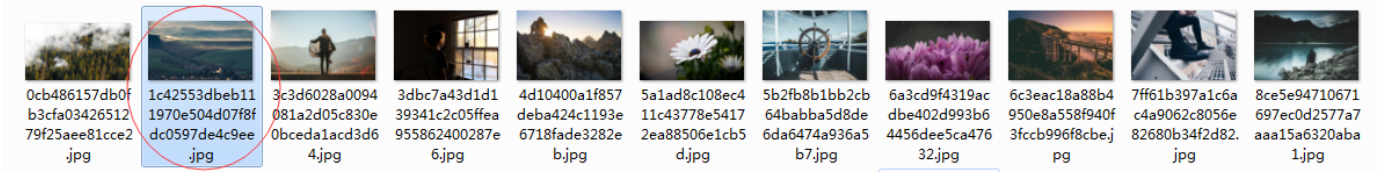
运行效果：

```
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/WJX4VCVKAS.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/I5M7YGSETN.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/P59BNODGE2.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/2VCOAPP524.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/JG8AXYUVU3.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/ST0Y7XFYJU.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/7H0AGM0DOJ.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/DE81H336XB.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/W1VNXBKR9U.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/FMFZO3TV9B.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/XPNTVEF1SZ.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/ASKA6O9G1Y.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/OWMN4VA7W1.jpg',
            u'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/CWJNF3RY0O.jpg'],
 'images': [{'checksum': '41c10b6961656fbbf4525f09accd5c20',
             'path': 'full/1c42553dbeb111970e504d07f8fdc0597de4c9ee.jpg',
             'url': 'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/07EK7KIUOV.jpg'},
            {'checksum': 'c4fd5c24843e7d7cfbfb6c73fcce7ffb',
             'path': 'full/5a1ad8c108ec411c43778e54172ea88506e1cb5d.jpg',
             'url': 'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/60LMGOXWCY.jpg'},
            {'checksum': '30c59cfc9085cb10f761f11d7c992980',
             'path': 'full/71c92c24962e31ec9cbd6568a01b07dd656618d7.jpg',
             'url': 'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/PHIVVGZGP0.jpg'},
            {'checksum': 'db9284cadc85f68212901d10a63a9351',
             'path': 'full/2468a1e9398fd4a499bda2c34cc26fa2afc23328.jpg',
             'url': 'https://snap-photos.s3.amazonaws.com/img-thumbs/280h/ABC06Z73WV.jpg'},
```

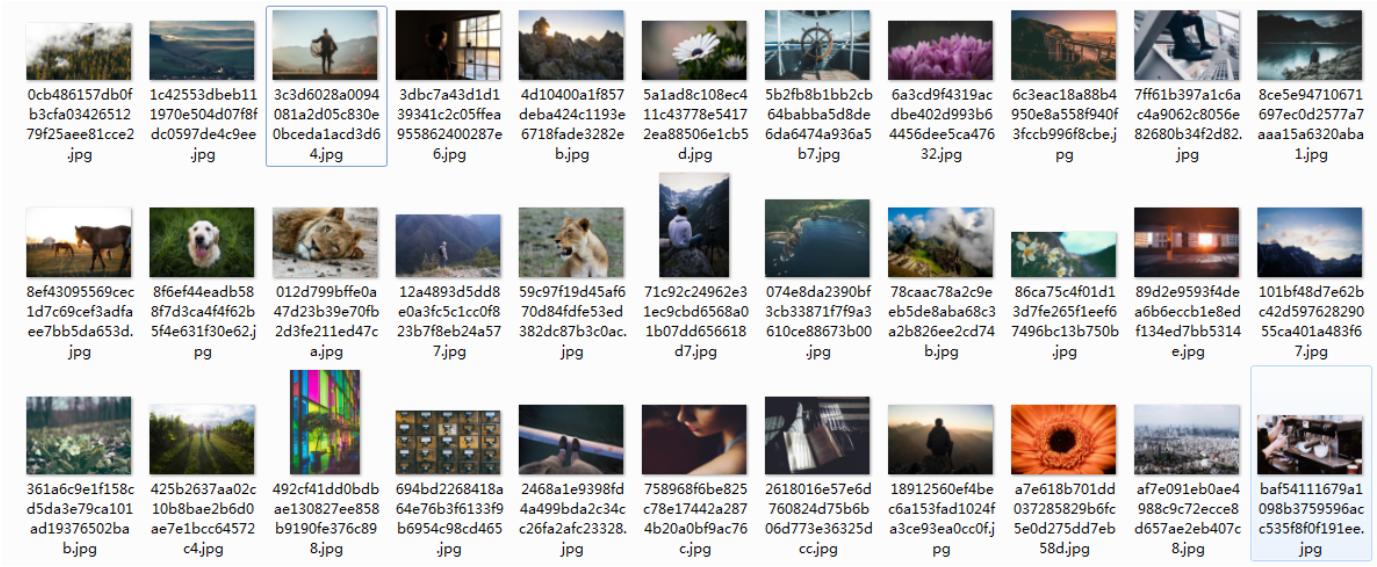
本地图片显示：存储在本地设置的路径下full文件下，图片的名字使用图片url的SHA1 hash(这样的值很少会重复，所以可以实现重复判断，数据库中的去重操作的主键也常使用消息摘要算法)
示例：图片的url = `https://snap-photos.s3.amazonaws.com/img-thumbs/280h/07EK7KIUOV.jpg` 网站显示

```
import hashlib
print(hashlib.sha1(b"https://snap-photos.s3.amazonaws.com/img-thumbs/280h/07EK7KIUOV.jpg").hexdigest())
# 显示：'1c42553dbeb111970e504d07f8fdc0597de4c9ee'
```

图片显示：



全部图片：



完整版代码

3：总结与参考

第一次接触，就算是正确答案，你也不能完全的明白，所以参考文献的多次重复可以让你渐渐的明白原理和操作

- 参考列表
  - 列表1
  - 列表2
  - 列表3
  - 列表4：官方文档

> **任何实用性的东西都解决不了你所面临的实际问题，但为什么还要看？为了经验，为了通过阅读抓取别人的经验，虽然还需批判思维看待**
>
> 如果你忍不住的想要和我交朋友：email: 1156143589@qq.com