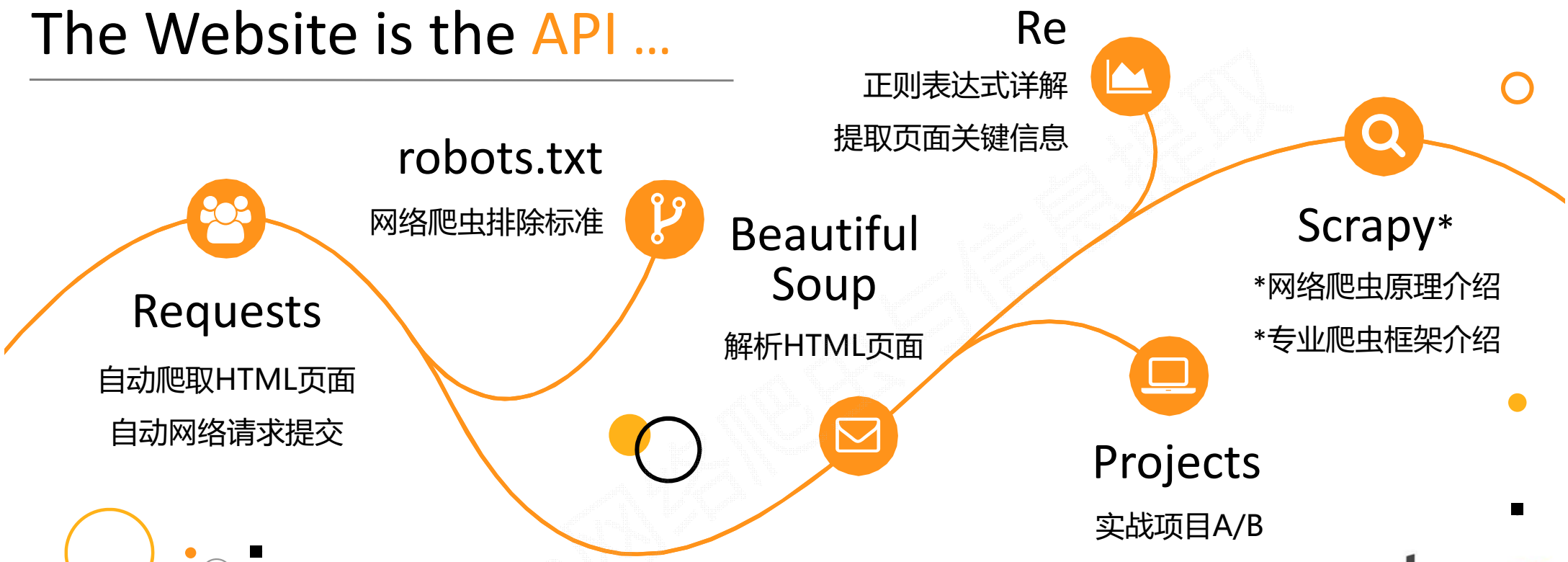
实例4：股票数据Scrapy爬虫

WS12

嵩天

www.python123.org

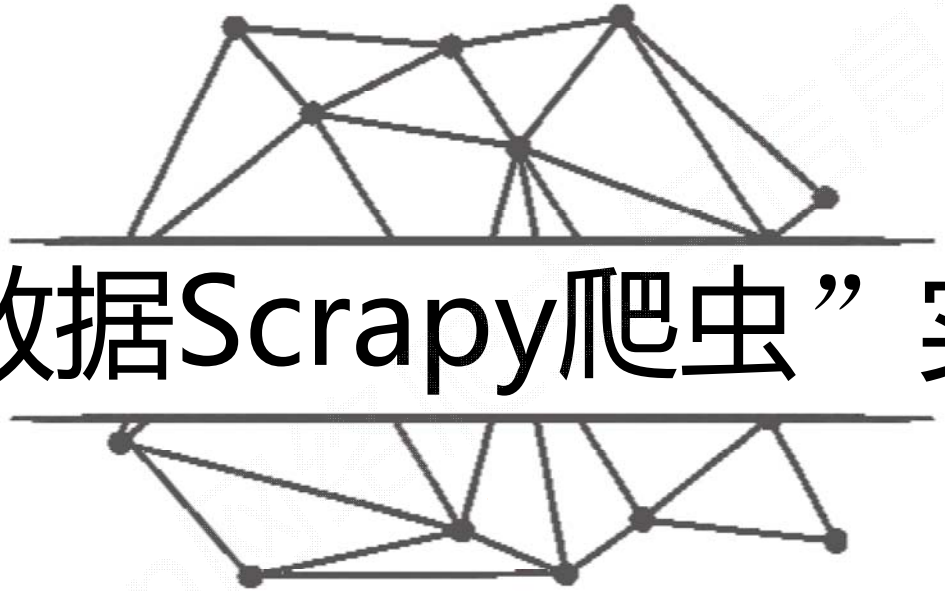


掌握定向网络数据爬取和网页解析的基本能力

python
弹指之间 · 享受创新

Python网络爬虫与信息提取

O4X -Tian



“股票数据Scrapy爬虫”实例介绍

功能描述

目标：获取上交所和深交所所有股票的名称和交易信息

输出：保存到文件中

技术路线：`scrapy`

数据网站的确定

获取股票列表：

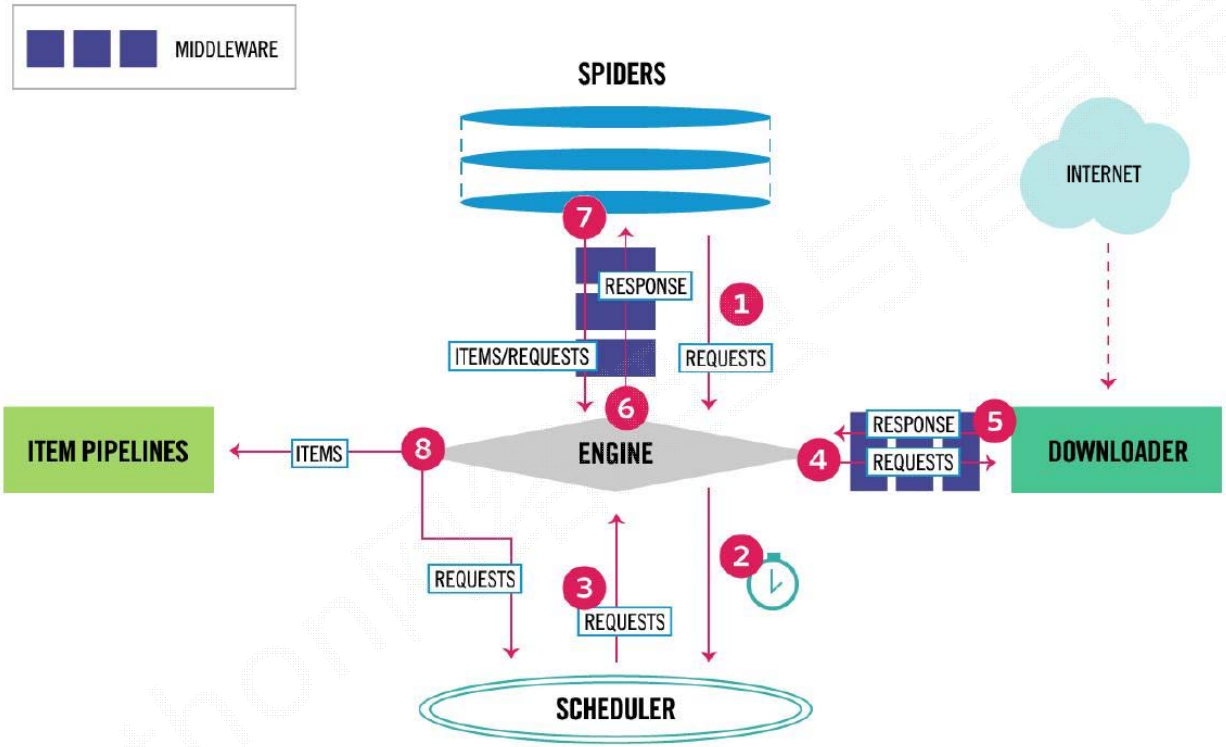
东方财富网：http://quote.eastmoney.com/stocklist.html

获取个股信息：

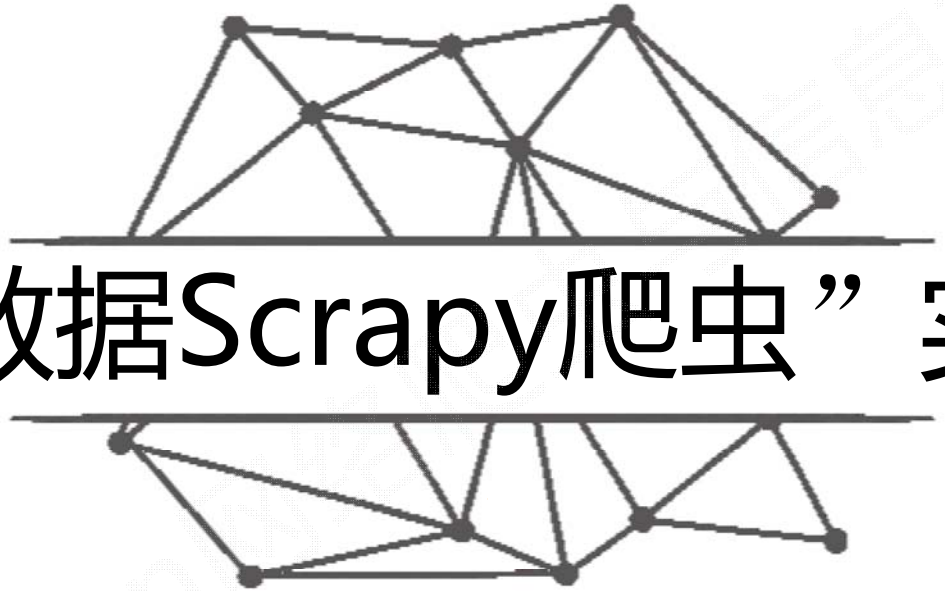
百度股票：https://gupiao.baidu.com/stock/

单个股票：https://gupiao.baidu.com/stock/sz002439.html

程序框架



编写spider处理链接爬取和页面解析，编写pipelines处理信息存储



“股票数据Scrapy爬虫”实例编写

步骤

步骤1：建立工程和Spider模板

步骤2：编写Spider

步骤3：编写ITEM Pipelines

Python网络爬虫与信息提取

步骤1：建立工程和Spider模板

```
\>scrapy startproject BaiduStocks
\>cd BaiduStocks
\>scrapy genspider stocks baidu.com
```

进一步修改spiders/stocks.py文件

Python网络爬虫与信息提取

步骤2：编写Spider

配置stocks.py文件

修改对返回页面的处理

修改对新增URL爬取请求的处理

Python网络爬虫与信息提取

stocks.py（修改前）

```
# -*- coding: utf-8 -*-
import scrapy


class StocksSpider(scrapy.Spider):
    name = "stocks"
    allowed_domains = ["baidu.com"]
    start_urls = ['http://baidu.com/']

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

stocks.py（修改中）

```
import scrapy
import re

class StocksSpider(scrapy.Spider):
    name = "stocks"
    start_urls = ['http://quote.eastmoney.com/stocklist.html']

    def parse(self, response):
        for href in response.css('a::attr(href)').extract():
            try:
                stock = re.findall(r"[s][hz]\d{6}", href)[0]
                url = 'https://gupiao.baidu.com/stock/' + stock + '.html'
                yield scrapy.Request(url, callback=self.parse_stock)
            except:
                continue
```

stocks.py（修改后）

```
def parse_stock(self, response):
    infoDict = {}
    stockInfo = response.css('.stock-bets')
    name = stockInfo.css('.bets-name').extract()[0]
    keyList = stockInfo.css('dt').extract()
    valueList = stockInfo.css('dd').extract()
    for i in range(len(keyList)):
        key = re.findall(r'>.*</dt>', keyList[i])[0][1:-5]
        try:
            val = re.findall(r'\d+\.?.*</dd>', valueList[i])[0][0:-5]
        except:
            val = '--'
        infoDict[key] = val

    infoDict.update(
        {'股票名称': re.findall('\s.*\(', name)[0].split()[0] + \
         re.findall('\>.*\<', name)[0][1:-1]})

    yield infoDict
```

步骤3：编写Pipelines

配置pipelines.py文件

定义对爬取项（Scraped Item）的处理类

配置ITEM_PIPELINES选项

pipelines.py（修改前）

```
# -*- coding: utf-8 -*-

# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: http://doc.scrapy.org/en/latest/topics/item-pipeline.html


class BaidustocksPipeline(object):
    def process_item(self, item, spider):
        return item
```

pipelines.py（修改后）

```
# -*- coding: utf-8 -*-

# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: http://doc.scrapy.org/en/latest/topics/item-pipeline.html

class BaidustocksPipeline(object):
    def process_item(self, item, spider):
        return item

class BaidustocksInfoPipeline(object):
    def open_spider(self, spider):
        self.f = open('BaiduStockInfo.txt', 'w')

    def close_spider(self, spider):
        self.f.close()

    def process_item(self, item, spider):
        try:
            line = str(dict(item)) + "\n"
            self.f.write(line)
        except:
            pass
        return item
```

配置ITEM_PIPELINES选项

配置settings.py文件

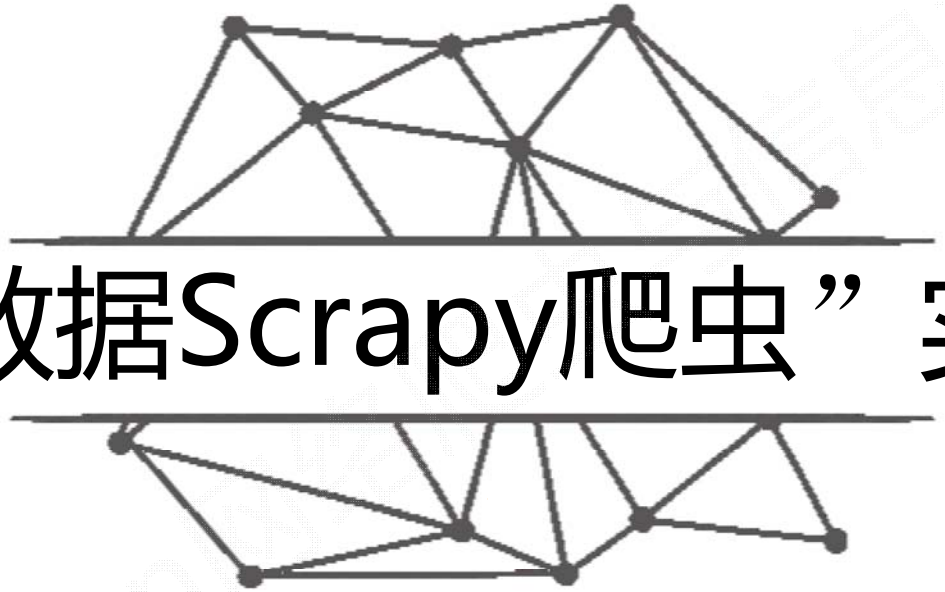
```
# Configure item pipelines
# See http://scrapy.readthedocs.org/en/latest/topics/item-pipeline.html
ITEM_PIPELINES = {
    'BaiduStocks.pipelines.BaidustocksInfoPipeline': 300,
}
```

Python爬虫从入门到放弃

程序的执行

```
\>scrapy crawl stocks
```

Python爬虫入门教程



“股票数据Scrapy爬虫”实例优化

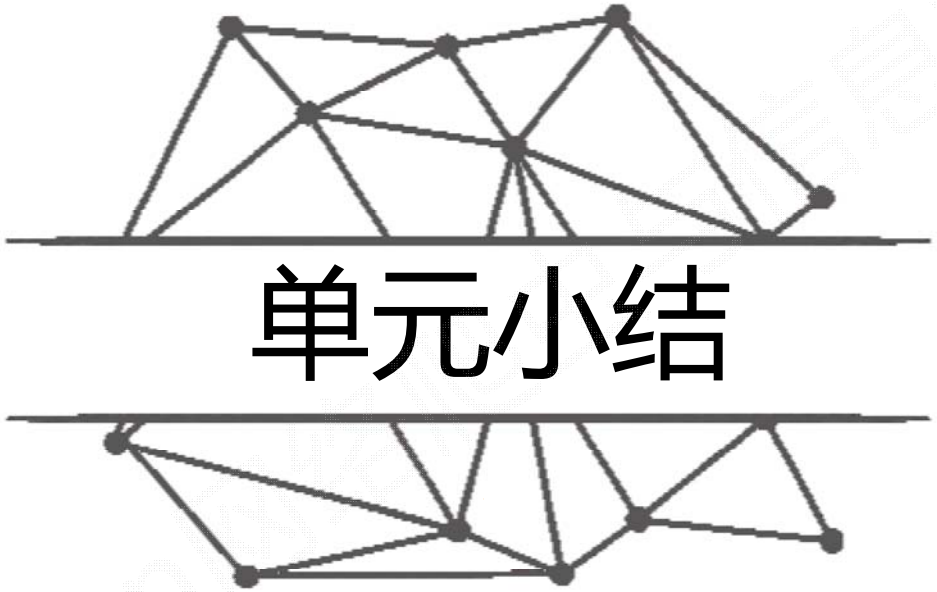
如何进一步提高scrapy爬虫爬取速度？

Python网络爬虫与信息提取

配置并发连接选项

settings.py文件

选项	说明
CONCURRENT_REQUESTS	Downloader最大并发请求下载数量，默认32
CONCURRENT_ITEMS	Item Pipeline最大并发ITEM处理数量，默认100
CONCURRENT_REQUESTS_PER_DOMAIN	每个目标域名最大的并发请求数量，默认8
CONCURRENT_REQUESTS_PER_IP	每个目标IP最大的并发请求数量，默认0，非0有效



单元小结

实例4：股票数据Scrapy爬虫

完整配置并实现Scrapy爬虫的过程

1 建立工程和Spider模板

2 编写Spider

3 编写Pipeline

4 配置优化