

Scrapy爬虫框架

WS10



嵩天

www.python123.org

The Website is the API ...



Requests

自动爬取HTML页面
自动网络请求提交

robots.txt

网络爬虫排除标准



Beautiful Soup

解析HTML页面



Re

正则表达式详解
提取页面关键信息



Scrapy*

*专业爬虫框架介绍
*爬虫框架的基本使用



Projects

实战项目A/B

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

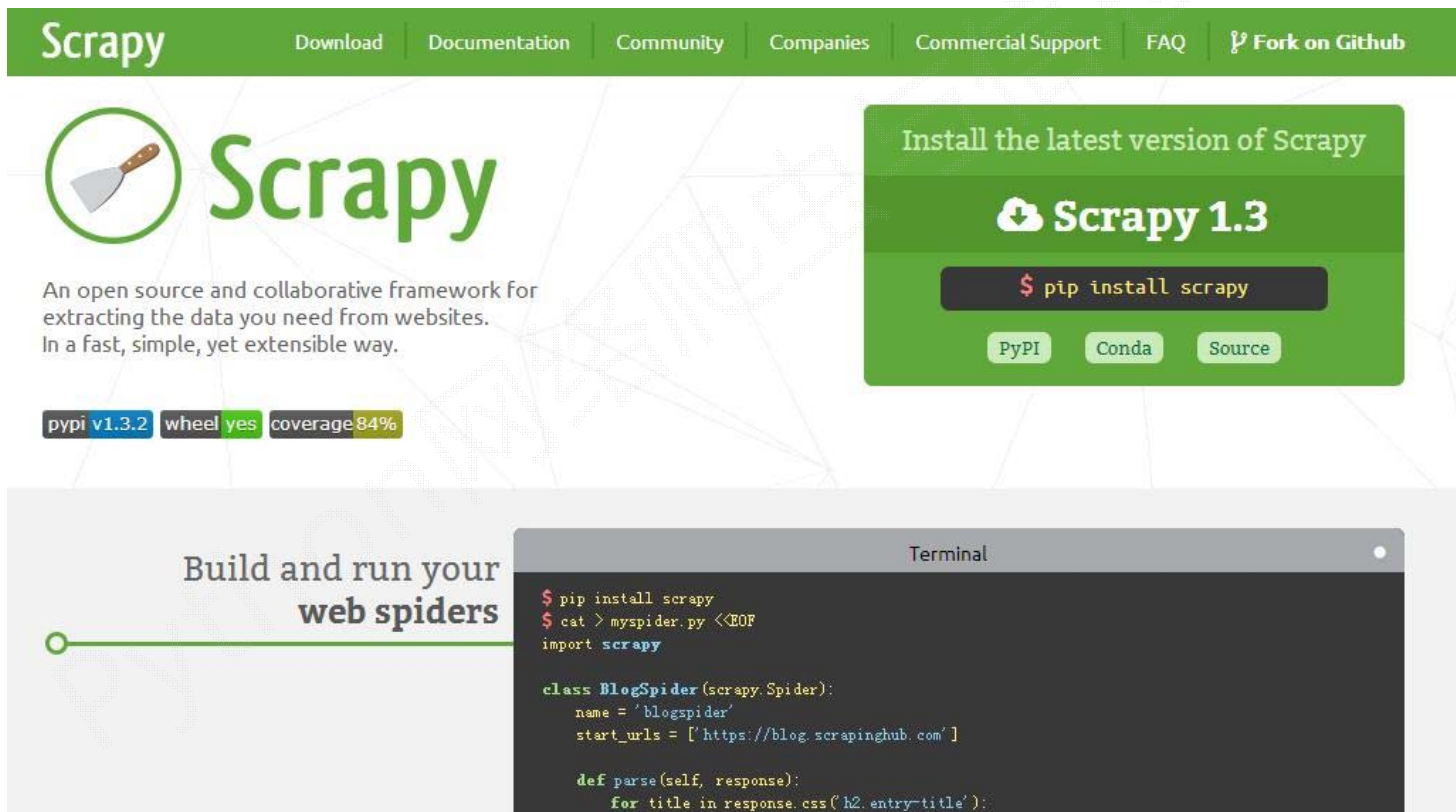
04X -Tian



Scrapy爬虫框架介绍

Scrapy


Scrapy是一个快速功能强大的网络爬虫框架



The image is a screenshot of the Scrapy project's official website. At the top, there is a green navigation bar with the Scrapy logo and links for Download, Documentation, Community, Companies, Commercial Support, FAQ, and Fork on Github. Below the navigation bar, the main content area features the Scrapy logo (a green circle with a scraper icon) and the word 'Scrapy' in large green letters. To the right of the logo, there is a green box with the text 'Install the latest version of Scrapy' and 'Scrapy 1.3'. Below this, there is a dark green button with the command '\$ pip install scrapy' and three smaller buttons for 'PyPI', 'Conda', and 'Source'. Below the main content, there is a section titled 'Build and run your web spiders' with a green circle icon. To the right of this section, there is a terminal window showing the installation and usage of Scrapy. The terminal output includes the command '\$ pip install scrapy', the command '\$ cat > myspider.py <<EOF', the code 'import scrapy', the class definition 'class BlogSpider(scrapy.Spider):', the attributes 'name = 'blogspider'' and 'start_urls = ['https://blog.scrapinghub.com']', and the parse method 'def parse(self, response):' with a loop 'for title in response.css('h2.entry-title'):'.

Scrapy


Download Documentation Community Companies Commercial Support FAQ Fork on Github

 **Scrapy**

An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

pypi v1.3.2 wheel yes coverage 84%

Install the latest version of Scrapy

 **Scrapy 1.3**

`$ pip install scrapy`

PyPI Conda Source

Build and run your web spiders

Terminal

```
$ pip install scrapy
$ cat > myspider.py <<EOF
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://blog.scrapinghub.com']

    def parse(self, response):
        for title in response.css('h2.entry-title'):
```

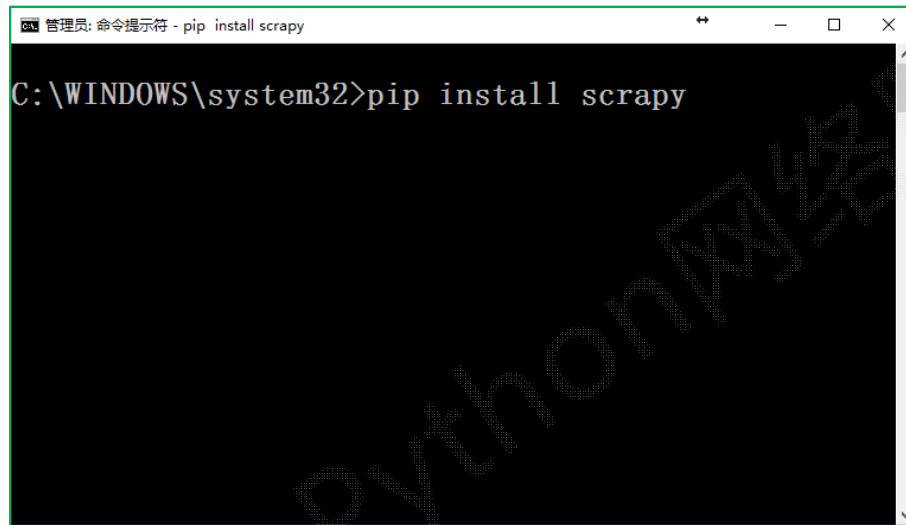
Scrapy的安装

Win平台：“以管理员身份运行” cmd

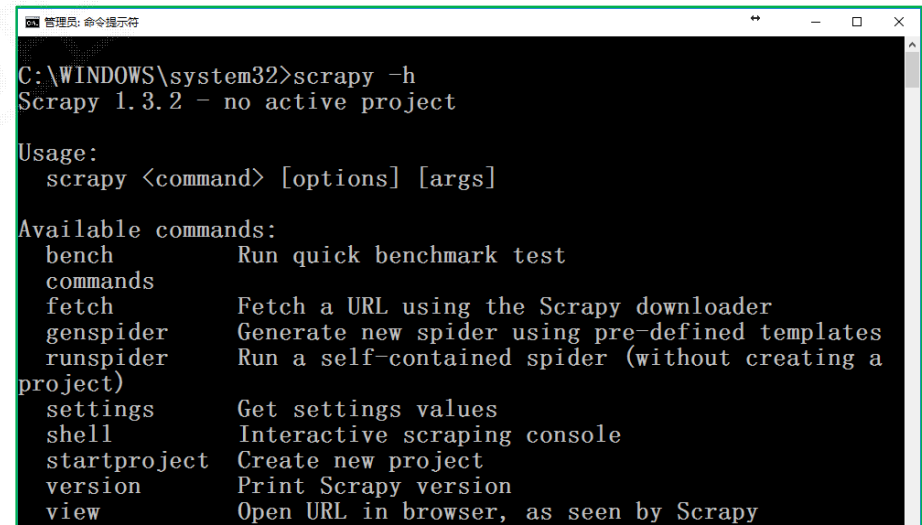
执行 `pip install scrapy`

安装后小测：

执行 `scrapy -h`



```
管理员: 命令提示符 - pip install scrapy
C:\WINDOWS\system32>pip install scrapy
```



```
管理员: 命令提示符
C:\WINDOWS\system32>scrapy -h
Scrapy 1.3.2 - no active project

Usage:
  scrapy <command> [options] [args]

Available commands:
  bench          Run quick benchmark test
  commands
  fetch          Fetch a URL using the Scrapy downloader
  genspider      Generate new spider using pre-defined templates
  runspider      Run a self-contained spider (without creating a
project)
  settings       Get settings values
  shell          Interactive scraping console
  startproject   Create new project
  version        Print Scrapy version
  view           Open URL in browser, as seen by Scrapy
```

Scrapy爬虫框架结构

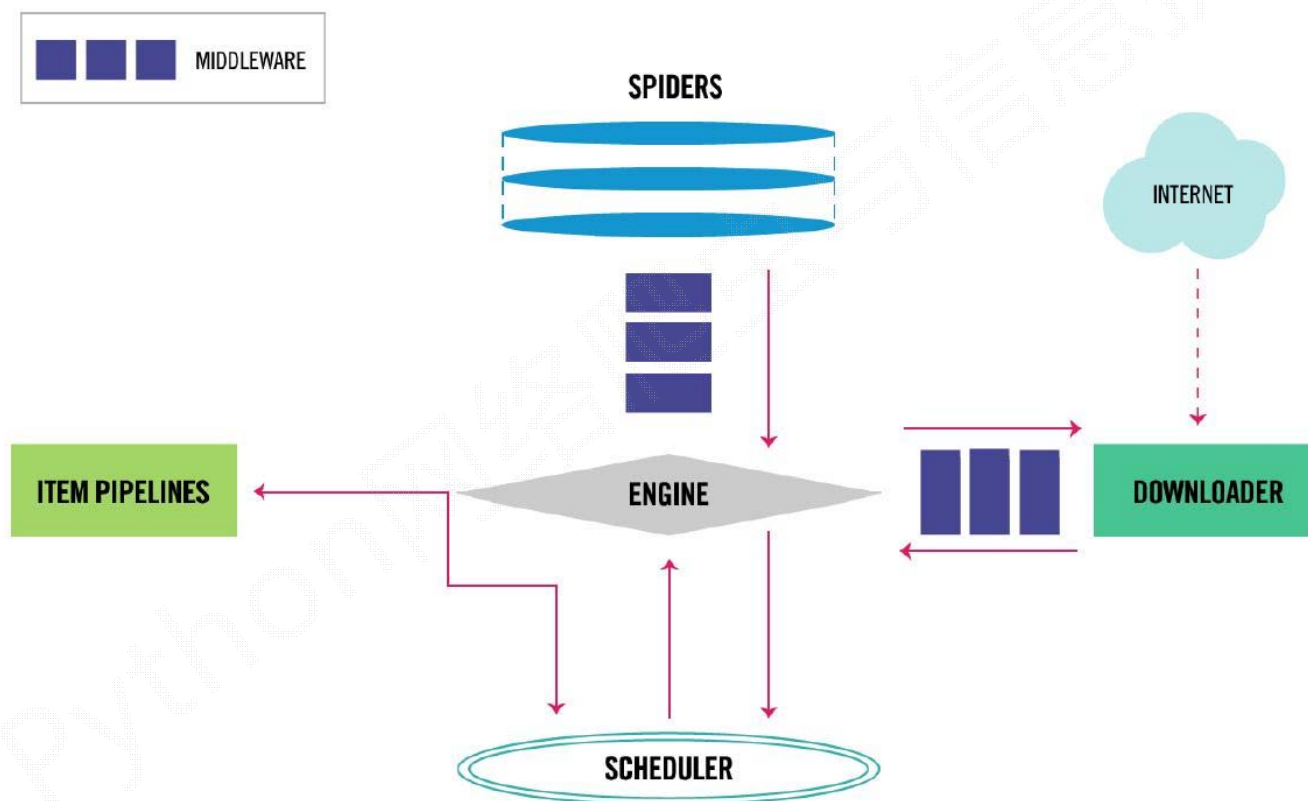
Scrapy不是一个函数功能库，而是一个爬虫框架。

什么是爬虫框架？

爬虫框架是实现爬虫功能的一个软件结构和功能组件集合。

爬虫框架是一个半成品，能够帮助用户实现专业网络爬虫。

Scrapy爬虫框架结构

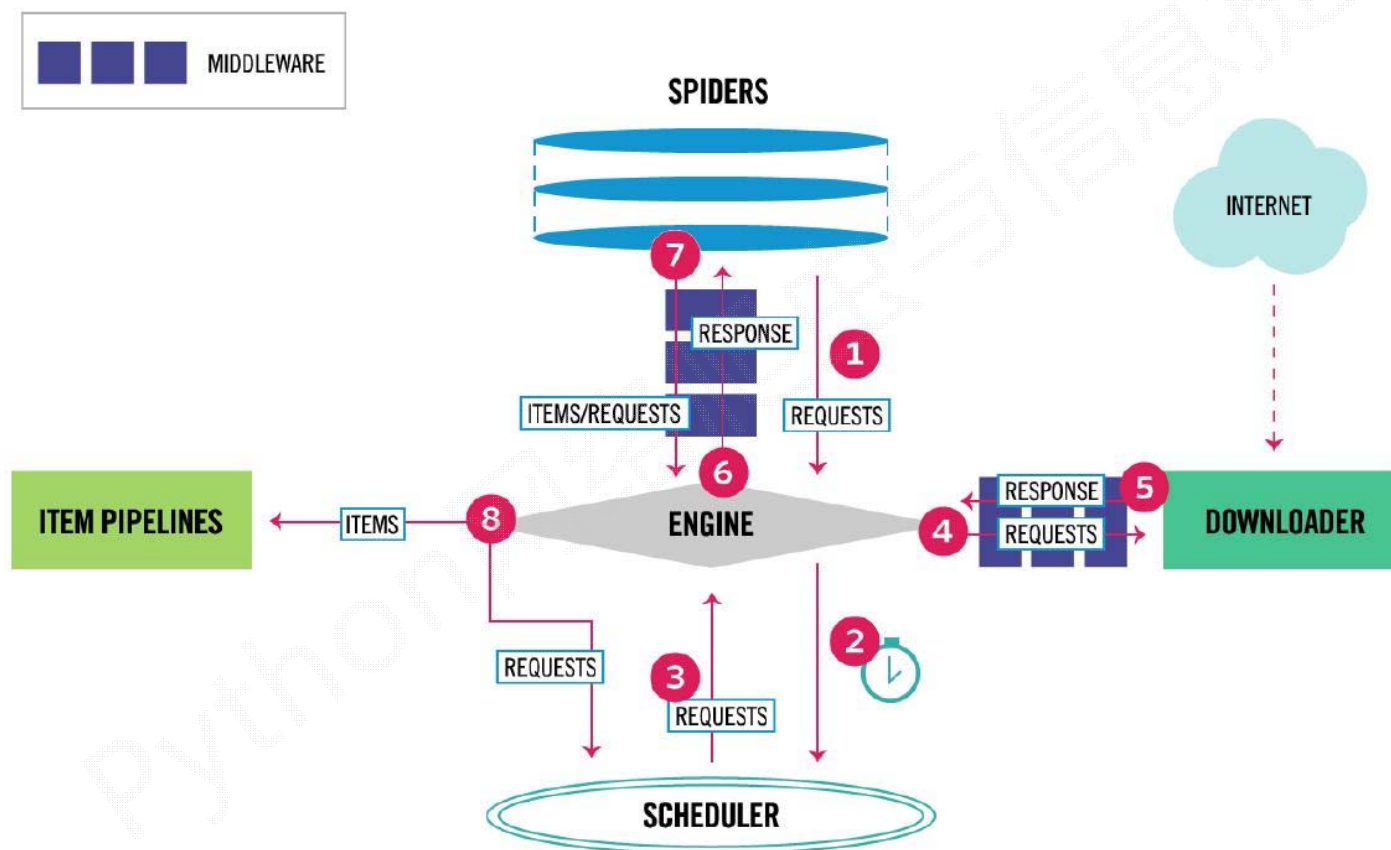


看结构

分布式

“5+2” 结构

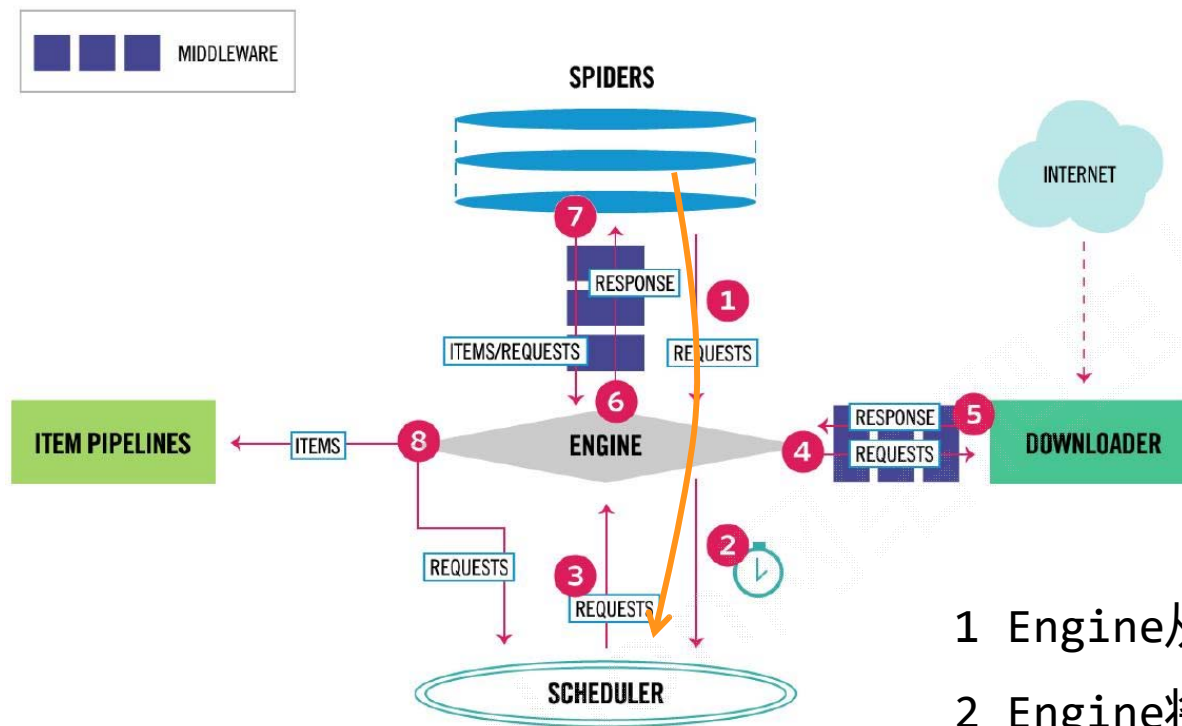
Scrapy爬虫框架结构



看流程

数据流

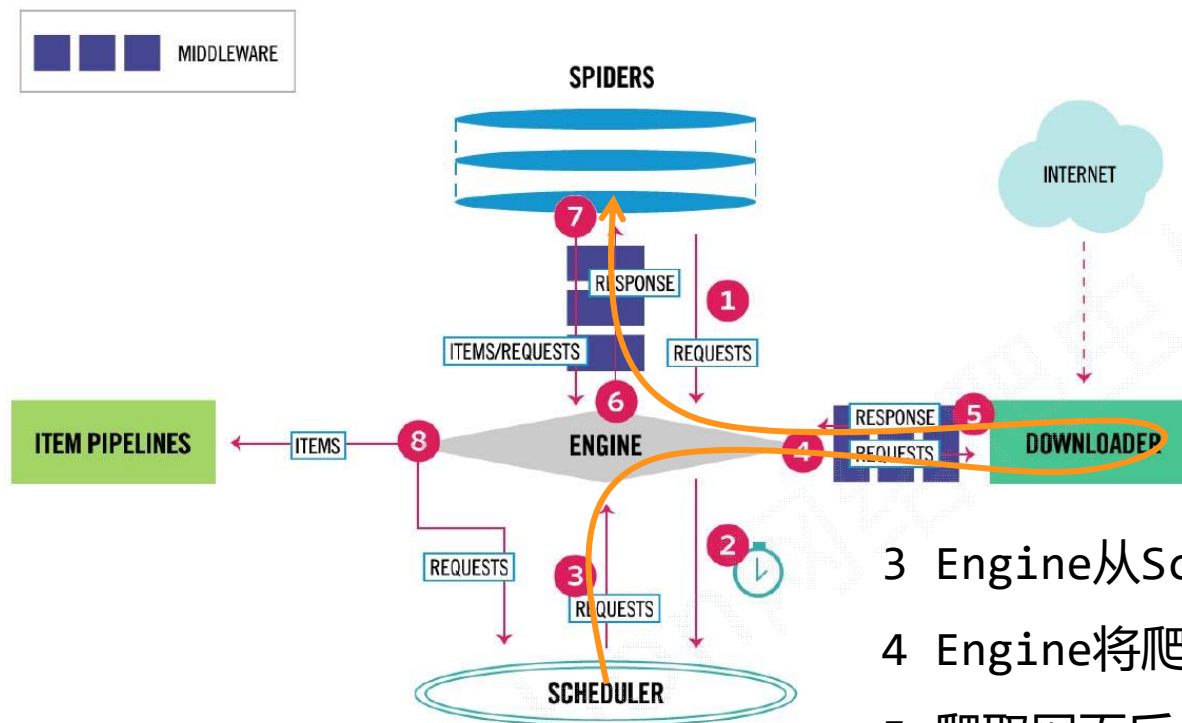
Scrapy爬虫框架结构



数据流的三个路径
(1)

- 1 Engine从Spider处获得爬取请求(Request)
- 2 Engine将爬取请求转发给Scheduler, 用于调度

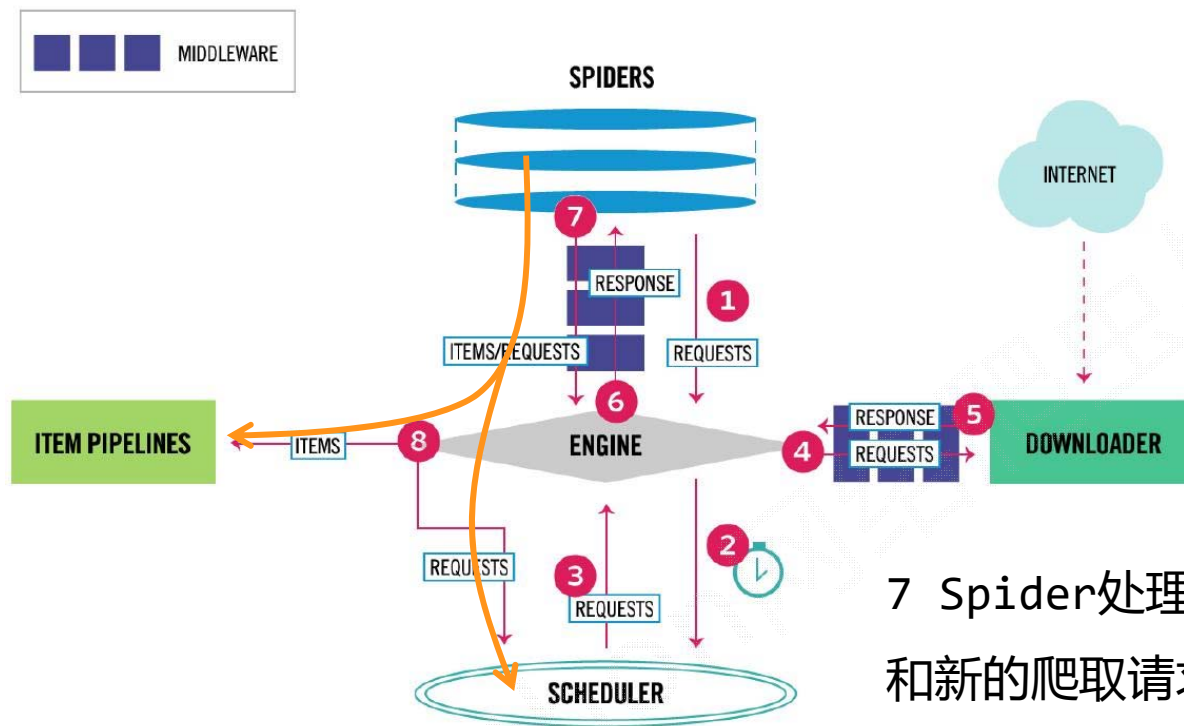
Scrapy爬虫框架结构



数据流的三个路径
(2)

- 3 Engine从Scheduler处获得下一个要爬取请求
- 4 Engine将爬取请求通过中间件发送给Downloader
- 5 爬取网页后，Downloader形成响应（Response）通过中间件发给Engine
- 6 Engine将收到的响应通过中间件发送给Spider处理

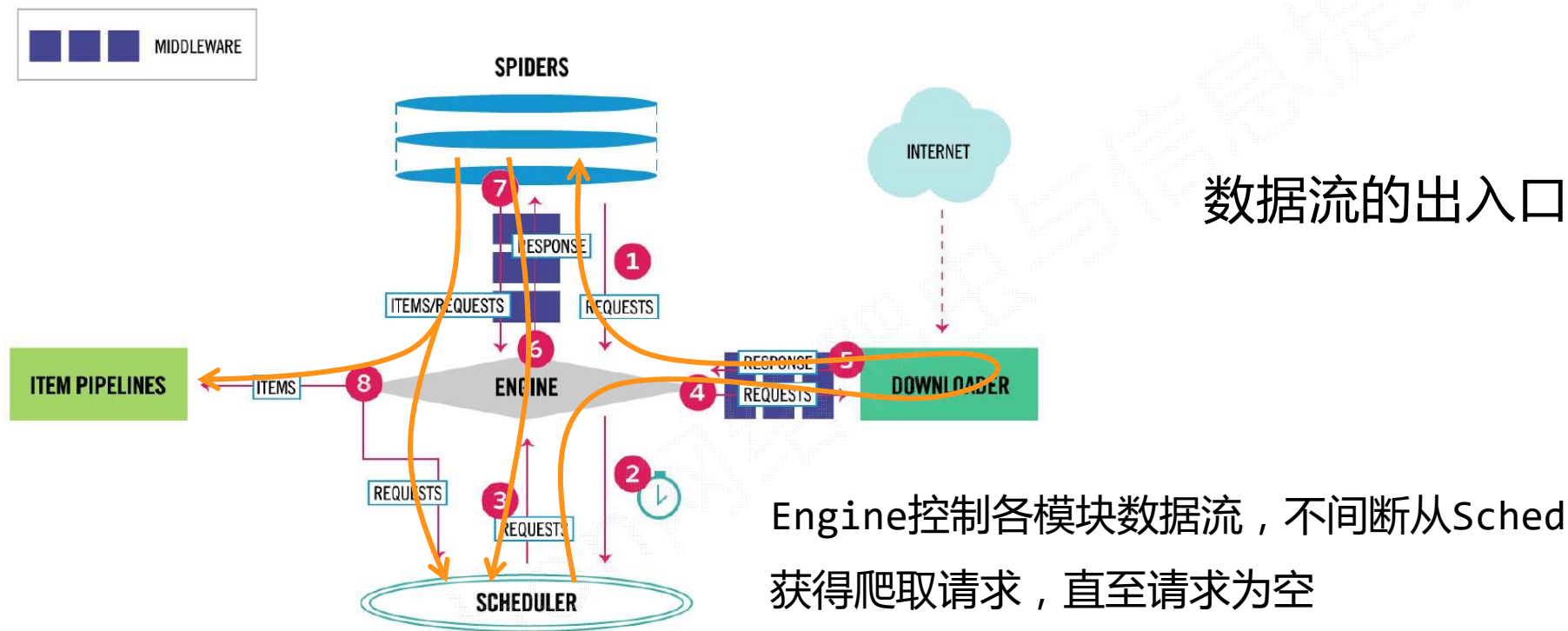
Scrapy爬虫框架结构



数据流的三个路径
(3)

- 7 Spider处理响应后产生爬取项 (scraped Item) 和新的爬取请求 (Requests) 给Engine
- 8 Engine将爬取项发送给Item Pipeline (框架出口)
- 9 Engine将爬取请求发送给Scheduler

Scrapy爬虫框架结构

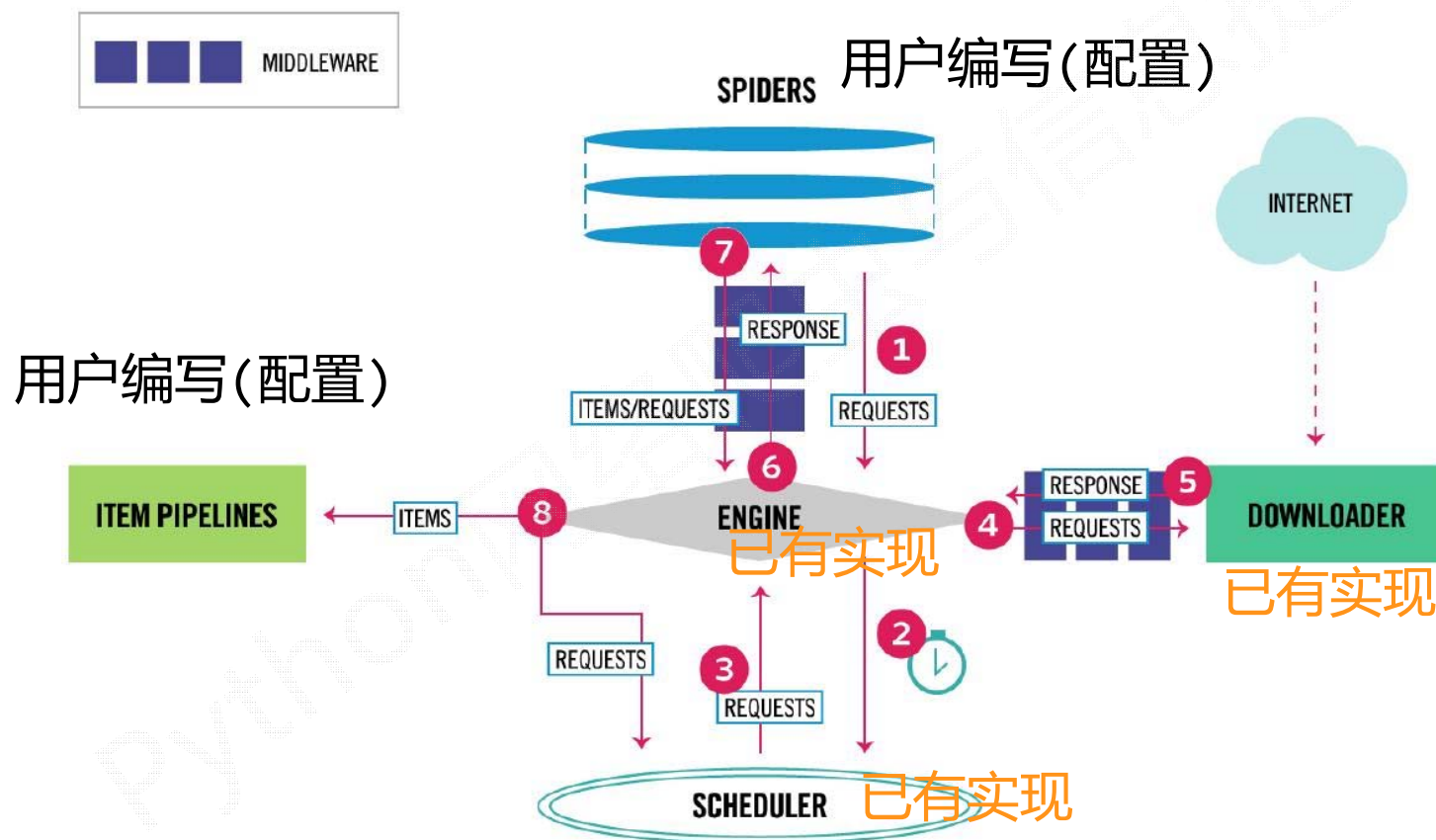


Engine控制各模块数据流，不间断从Scheduler处获得爬取请求，直至请求为空

框架入口：Spider的初始爬取请求

框架出口：Item Pipeline

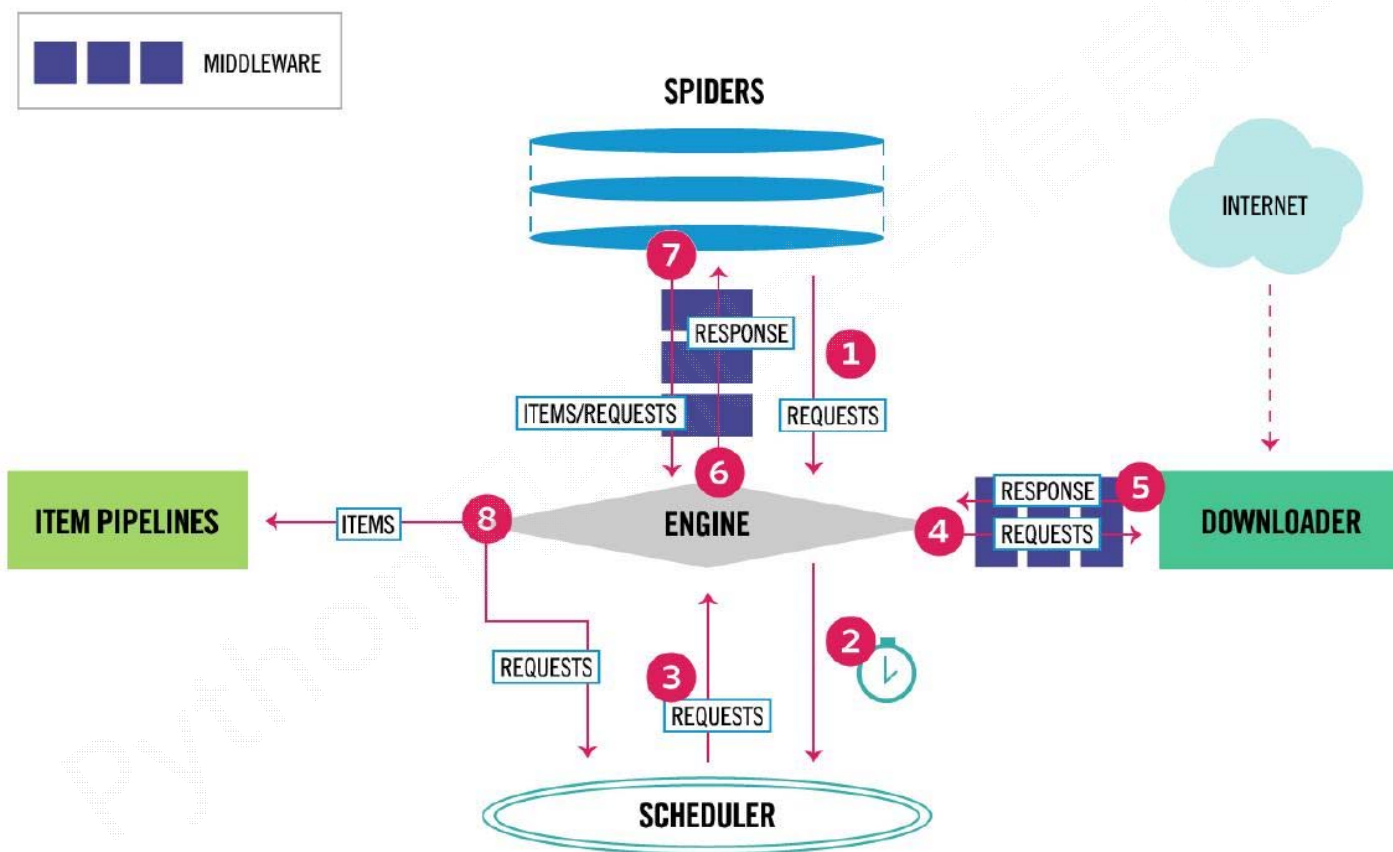
Scrapy爬虫框架结构





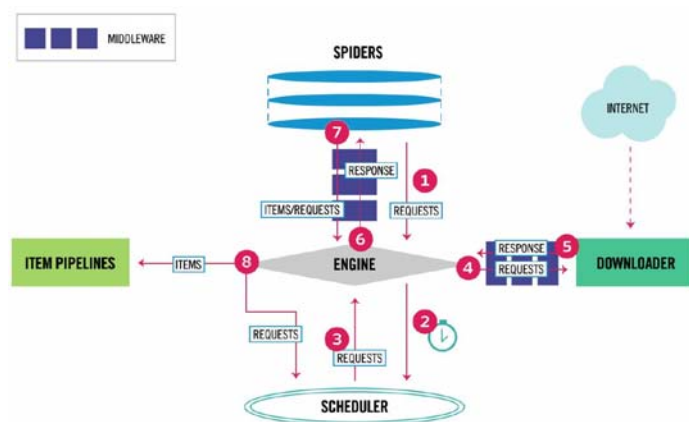
Scrapy爬虫框架解析

Scrapy爬虫框架结构



“5+2” 结构

Scrapy爬虫框架结构



Engine

(1) 控制所有模块之间的数据流

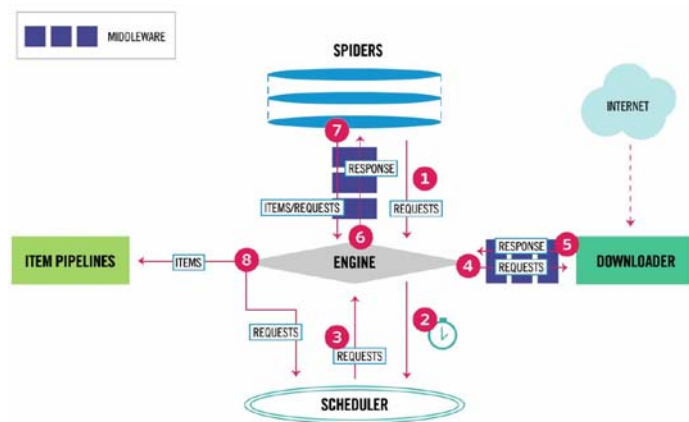
(2) 根据条件触发事件

不需要用户修改

“5+2” 结构

Engine

Scrapy爬虫框架结构



Downloader

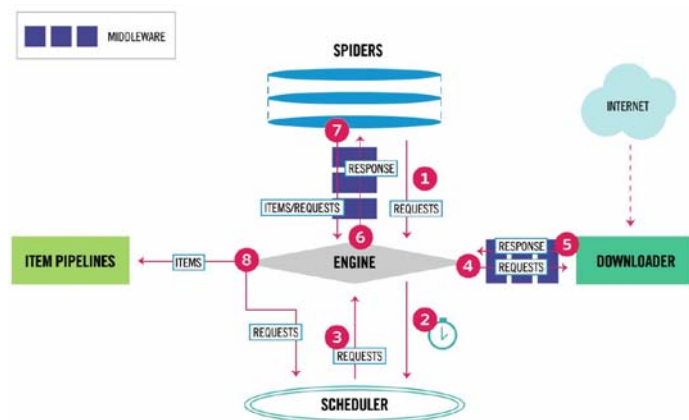
根据请求下载网页

不需要用户修改

“5+2” 结构

Downloader

Scrapy爬虫框架结构



Scheduler

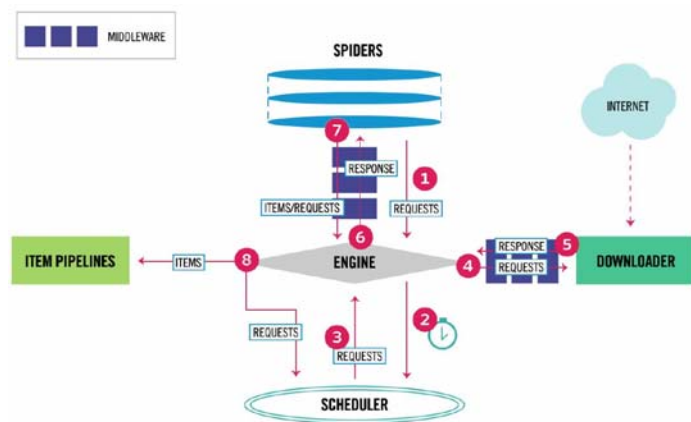
对所有爬取请求进行调度管理

不需要用户修改

“5+2” 结构

Scheduler

Scrapy爬虫框架结构



Downloader Middleware

目的：实施Engine、Scheduler和Downloader之间进行用户可配置的控制

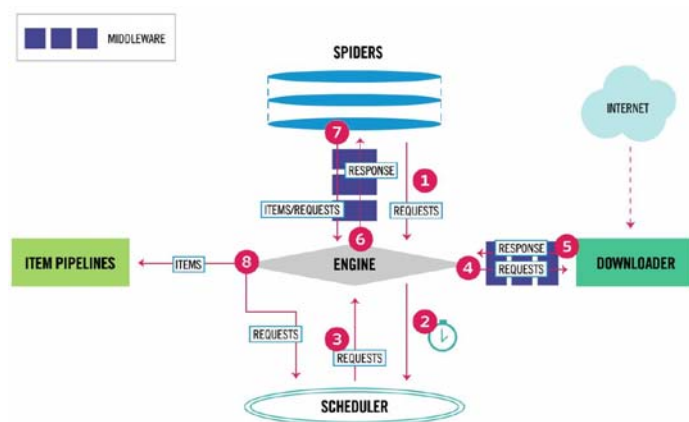
功能：修改、丢弃、新增请求或响应

“5+2” 结构

Downloader Middleware

用户可以编写配置代码

Scrapy爬虫框架结构



Spider

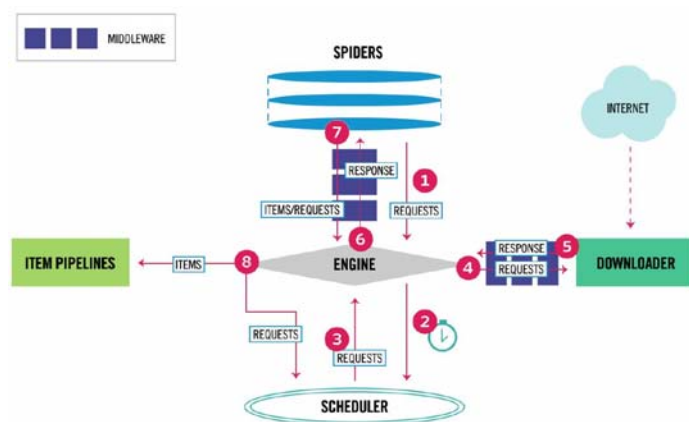
- (1) 解析Downloader返回的响应 (Response)
- (2) 产生爬取项 (scraped item)
- (3) 产生额外的爬取请求 (Request)

“5+2” 结构

Spider

需要用户编写配置代码

Scrapy爬虫框架结构



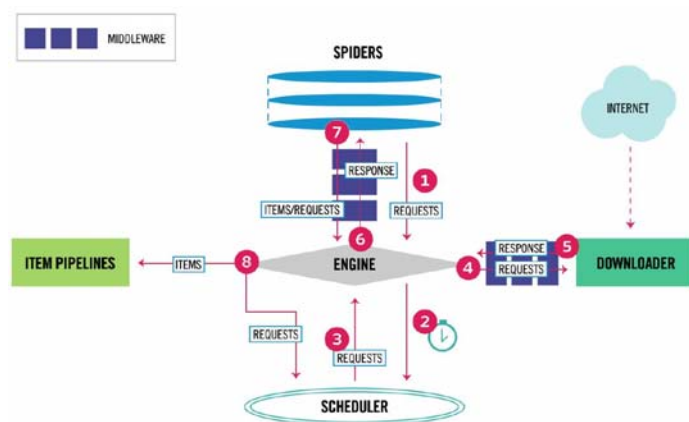
“5+2” 结构

Item Pipelines

Item Pipelines

- (1) 以流水线方式处理Spider产生的爬取项
 - (2) 由一组操作顺序组成，类似流水线，每个操作是一个Item Pipeline类型
 - (3) 可能操作包括：清理、检验和查重爬取项中的HTML数据、将数据存储在数据库
- 需要用户编写配置代码

Scrapy爬虫框架结构



Spider Middleware

目的：对请求和爬取项的再处理

功能：修改、丢弃、新增请求或爬取项

用户可以编写配置代码

“5+2” 结构

Spider Middleware



requests库和Scrapy爬虫的比较

requests vs. Scrapy

相同点：

两者都可以进行页面请求和爬取，Python爬虫的两个重要技术路线

两者可用性都好，文档丰富，入门简单

两者都没有处理js、提交表单、应对验证码等功能（可扩展）

requests vs. Scrapy

页面级爬虫

功能库

并发性考虑不足，性能较差

重点在于页面下载

定制灵活

上手十分简单

网站级爬虫

框架

并发性好，性能较高

重点在于爬虫结构

一般定制灵活，深度定制困难

入门稍难

选用哪个技术路线开发爬虫呢？

看情况

非常小的需求，requests库

不太小的需求，Scrapy框架

定制程度很高的需求（不考虑规模），自搭框架，requests > Scrapy

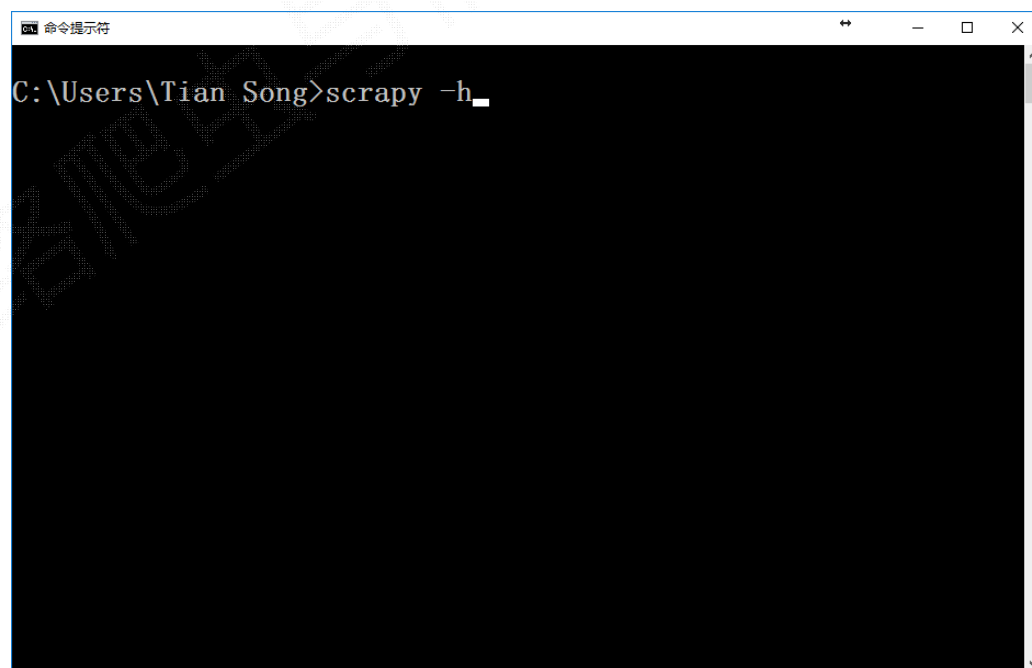


Scrapy爬虫的常用命令

Scrapy命令行

Scrapy是为持续运行设计的专业爬虫框架，提供操作的Scrapy命令行

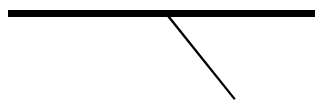
Win下，启动cmd控制台



```
命令提示符
C:\Users\Tian Song>scrapy -h_
```

Scrapy命令行格式

> scrapy <command> [options] [args]



Scrapy命令

Scrapy常用命令

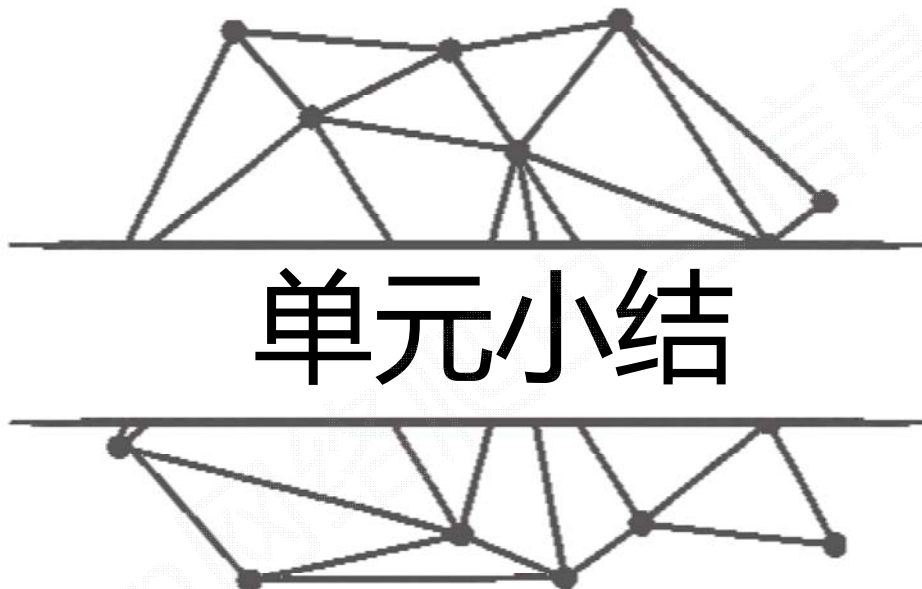
| 命令 | 说明 | 格式 |
|--------------|------------|--|
| startproject | 创建一个新工程 | scrapy startproject <name> [dir] |
| genspider | 创建一个爬虫 | scrapy genspider [options] <name> <domain> |
| settings | 获得爬虫配置信息 | scrapy settings [options] |
| crawl | 运行一个爬虫 | scrapy crawl <spider> |
| list | 列出工程中所有爬虫 | scrapy list |
| shell | 启动URL调试命令行 | scrapy shell [url] |

Scrapy爬虫的命令行逻辑

为什么Scrapy采用命令行创建和运行爬虫？

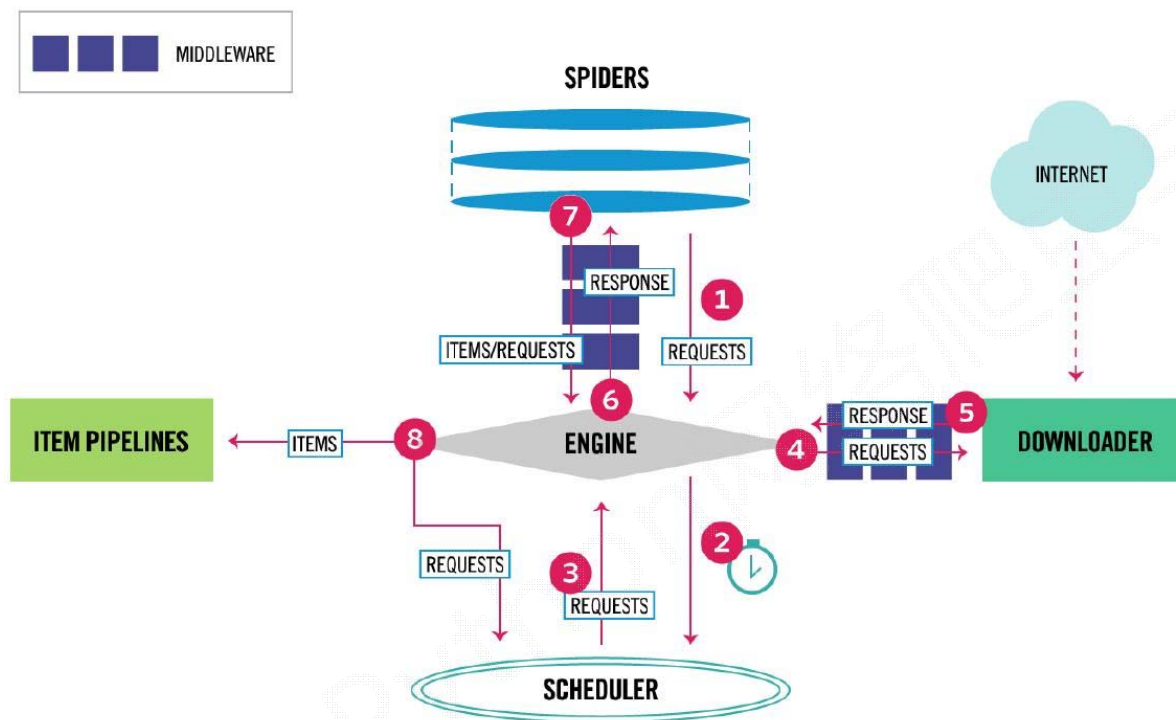
命令行（不是图形界面）更容易自动化，适合脚本控制

本质上，Scrapy是给程序员用的，功能（而不是界面）更重要



单元小结

Scrapy爬虫框架



Scrapy框架

“5+2” 结构

Scrapy命令行的
使用

Scrapy与
requests的不同