EDU

CSDN学院　IT实战派

Scrapy框架

讲师：黄老师

本章目标

1. 理解Scrapy架构。
2. 学会Spider爬虫的编写。
3. 学会CrawlSpider爬虫编写。
4. 学会中间件的编写。
5. 学会pipeline保存数据。
6. 学会将Scrapy结合selenium一起使用。
7. 学会在Scrapy中使用IP代理。

Scrapy框架介绍

写一个爬虫，需要做很多的事情。比如：发送网络请求、数据解析、数据存储、反反爬虫机制（更换ip代理、设置请求头等）、异步请求等。这些工作如果每次都要自己从零开始写的话，比较浪费时间。因此Scrapy把一些基础的东西封装好了，在他上面写爬虫可以变的更加的高效（爬取效率和开发效率）。因此真正在公司里，一些上了量的爬虫，都是使用Scrapy框架来解决。

安装Scrapy框架

1. 安装：通过pip install scrapy即可安装。
2. Scrapy官方文档：http://doc.scrapy.org/en/latest
3. Scrapy中文文档：http://scrapy-chs.readthedocs.io/zh_CN/latest/index.html

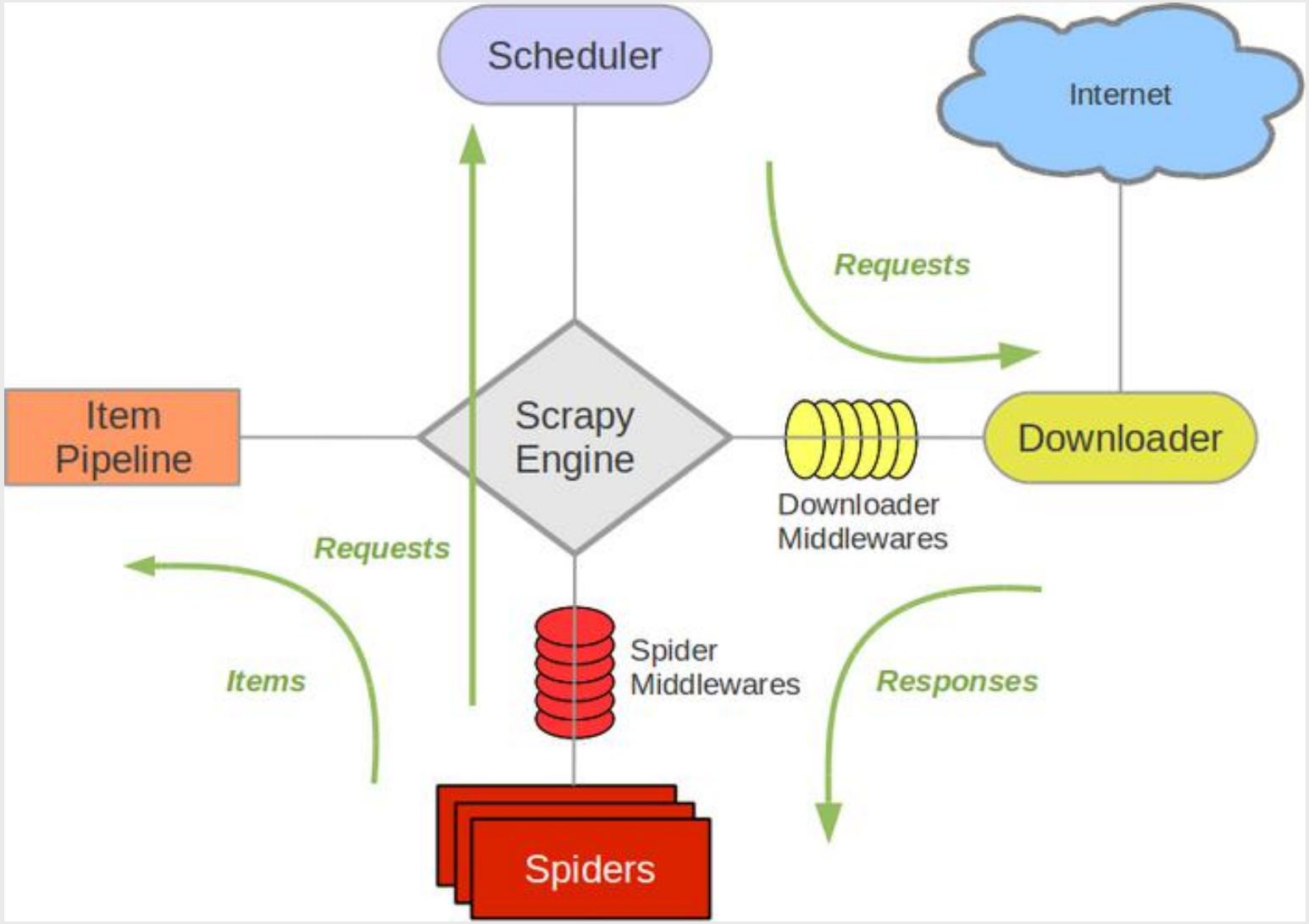
注意：

1. 在ubuntu上安装scrapy之前，需要先安装以下依赖：

sudo apt-get install python3-dev build-essential python3-pip libxml2-dev libxslt1-dev zlib1g-dev libffi-dev libssl-dev，然后再通过pip install scrapy安装。

2. 如果在windows系统下，提示这个错误ModuleNotFoundError: No module named 'win32api'，那么使用以下命令可以解决：pip install pypiwin32。
3. 如果安装的时候提示twisted安装有问题，那么可以先到这个网站下载twisted的whl文件：https://www.lfd.uci.edu/~gohlke/pythonlibs/，下载完成后，再使用pip install xxx.whl安装。

Scrapy框架架构



Scrapy框架架构

1. Scrapy Engine（引擎）：Scrapy框架的核心部分。负责在Spider和ItemPipeline、Downloader、Scheduler中间通信、传递数据等。
2. Spider（爬虫）：发送需要爬取的链接给引擎，最后引擎把其他模块请求回来的数据再发送给爬虫，爬虫就去解析想要的数据。这个部分是我们开发者自己写的，因为要爬取哪些链接，页面中的哪些数据是需要的，都是由程序员自己决定。
3. Scheduler（调度器）：负责接收引擎发送过来的请求，并按照一定的方式进行排列和整理，负责调度请求的顺序等。
4. Downloader（下载器）：负责接收引擎传过来的下载请求，然后去网络上下载对应的数据再交还给引擎。
5. Item Pipeline（管道）：负责将Spider（爬虫）传递过来的数据进行保存。具体保存在哪里，应该看开发者自己的需求。
6. Downloader Middlewares（下载中间件）：可以扩展下载器和引擎之间通信功能的中间件。
7. Spider Middlewares（Spider中间件）：可以扩展引擎和爬虫之间通信功能的中间件。

Scrapy快速入门

创建项目：

要使用Scrapy框架创建项目，需要通过命令来创建。首先进入到你想把这个项目存放的目录。然后使用以下命令创建：

scrapy startproject [项目名称]

创建爬虫：

scrapy genspider [爬虫名称] [爬虫作用的域名]

目录结构介绍：

items.py：用来存放爬虫爬取下来数据的模型。

middlewares.py：用来存放各种中间件的文件。

pipelines.py：用来将items的模型存储到本地磁盘中。

settings.py：本爬虫的一些配置信息（比如请求头、多久发送一次请求、ip代理池等）。

scrapy.cfg：项目的配置文件。

spiders包：以后所有的爬虫，都是存放到这个里面。

修改settings.py代码：

在做一个爬虫之前，一定要记得修改setttings.py中的设置。两个地方是强烈建议设置的。

1. ROBOTSTXT_OBEY设置为False。默认是True。即遵守机器协议，那么在爬虫的时候，scrapy首先去找robots.txt文件，如果没有找到。则直接停止爬取。
2. DEFAULT_REQUEST_HEADERS添加User-Agent。这个也是告诉服务器，我这个请求是一个正常的请求，不是一个爬虫。

运行scrapy项目：

需要在终端，进入项目所在的路径，然后scrapy crawl [爬虫名字]即可运行指定的爬虫。如果不想每次都在命令行中运行，那么可以把这个命令写在一个文件中。以后就在pycharm中执行运行这个文件就可以了。比如现在新创建一个文件叫做start.py，然后在这个文件中填入以下代码：

```
from scrapy import cmdline
cmdline.execute("scrapy crawl qsbk".split())
```

CrawlSpider爬虫

之前使用普通的Spider，我们是自己在解析完整个页面后获取下一页的url，然后重新发送一个请求。有时候我们想要这样做，只要满足某个条件的url，都给我进行爬取。那么这时候我们就可以通过CrawlSpider来帮我们完成了。CrawlSpider继承自Spider，只不过是在之前的基础之上增加了新的功能，可以定义爬取的url的规则，以后scrapy碰到满足条件的url都进行爬取，而不用手动的yield Request。

创建CrawlSpider爬虫：

之前创建爬虫的方式是通过scrapy genspider [爬虫名字] [域名]的方式创建的。如果想要创建CrawlSpider爬虫，那么应该通过以下命令创建：

scrapy genspider -t crawl [爬虫名字] [域名]

CrawlSpider爬虫

LinkExtractors链接提取器：

使用LinkExtractors可以不用程序员自己提取想要的url，然后发送请求。这些工作都可以交给LinkExtractors，他会在所有爬的页面中找到满足规则的url，实现自动的爬取。以下对LinkExtractors类做一个简单的介绍：

class scrapy.linkextractors.LinkExtractor(allow = (),deny = (),allow_domains = (),deny_domains = (),deny_extensions = None,restrict_xpaths = (),tags = ('a','area'),attrs = ('href'),canonicalize = True,unique = True,process_value = None)

主要参数讲解：

allow：允许的url。所有满足这个正则表达式的url都会被提取。

deny：禁止的url。所有满足这个正则表达式的url都不会被提取。

allow_domains：允许的域名。只有在这个里面指定的域名的url才会被提取。

deny_domains：禁止的域名。所有在这个里面指定的域名的url都不会被提取。

restrict_xpaths：严格的xpath。和allow共同过滤链接。

CrawlSpider爬虫

Rule规则类：

定义爬虫的规则类。以下对这个类做一个简单的介绍：

class scrapy.spiders.Rule(link_extractor, callback = None, cb_kwargs = None, follow = None,process_links = None, process_request = None)

主要参数讲解：

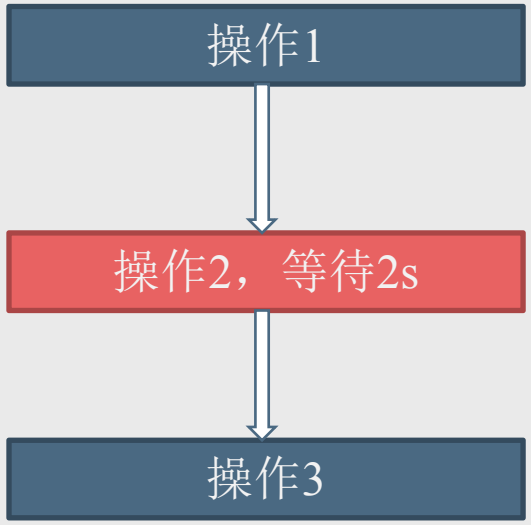
1. link_extractor：一个LinkExtractor对象，用于定义爬取规则。
2. callback：满足这个规则的url，应该要执行哪个回调函数。因为CrawlSpider使用了parse作为回调函数，因此不要覆盖parse作为回调函数自己的回调函数。
3. follow：指定根据该规则从response中提取的链接是否需要跟进。
4. process_links：从link_extractor中获取到链接后会传递给这个函数，用来过滤不需要爬取的链接。

CrawlSpider爬虫实战-猎云网爬虫

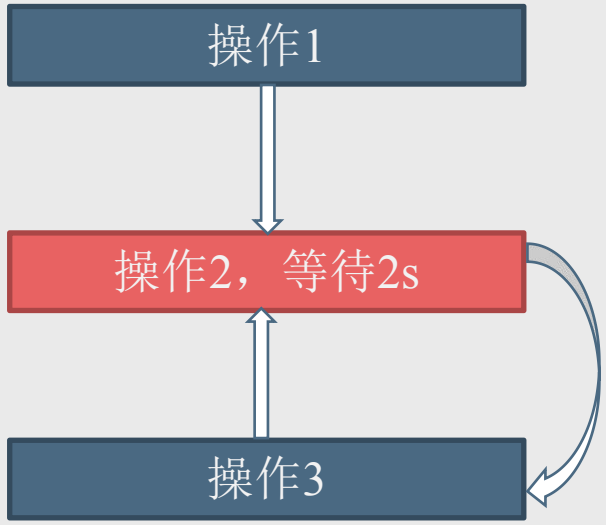
1. 需求：实现猎云网网站的文章数据爬虫。需要保存标题、发布时间、内容、原始url字段，然后异步保存到mysql数据库中。
2. 翻页链接：https://www.lieyunwang.com/latest/p1.html
3. 翻页规则：Rule(LinkExtractor(allow=r'/latest/p\d+.html'), follow=True)
4. 文章详情页规则：Rule(LinkExtractor(allow=r'/archives/\d+'), callback="parse_detail", follow=False)
5. 使用twisted. enterprise.adbapi来异步的保存数据。

异步保存MySQL数据

1. 使用twisted.enterprise.adbapi来创建连接池。
2. 使用runInteraction来运行插入sql语句的函数。
3. 在插入sql语句的函数中，第一个非self的参数就是cursor对象，使用这个对象执行sql语句。



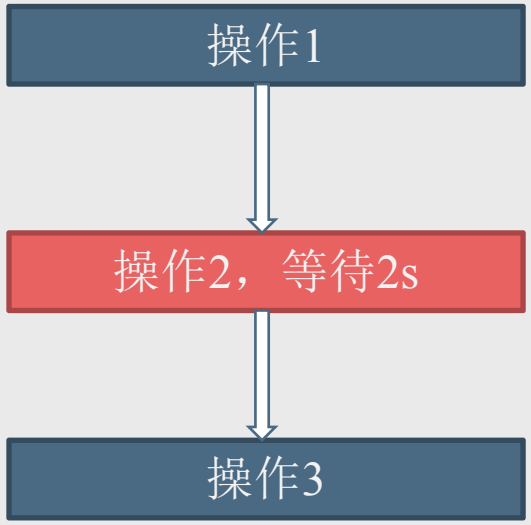
同步执行



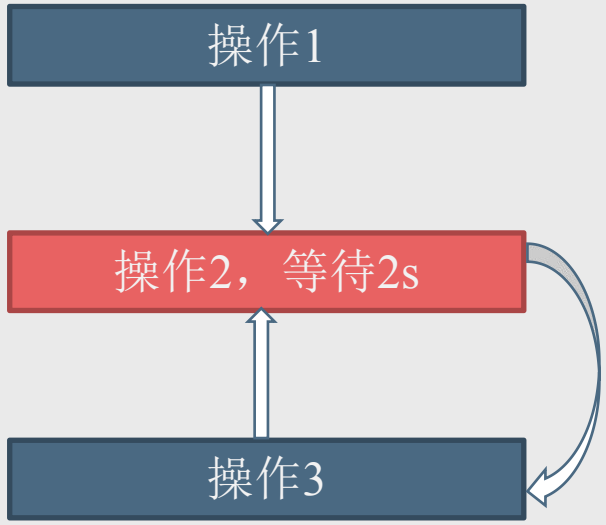
异步执行

异步保存MySQL数据

1. 使用twisted.enterprise.adbapi来创建连接池。
2. 使用runInteraction来运行插入sql语句的函数。
3. 在插入sql语句的函数中，第一个非self的参数就是cursor对象，使用这个对象执行sql语句。



同步执行



异步执行

下载图片文件

1. Scrapy框架下载文件（包括图片）有自己的一套解决方案，比我们直接使用urlretrieve更加有优势。
2. 避免重新下载最近已经下载过的文件。
3. 可以方便的指定文件存储的路径。
4. 可以将下载的图片转换成通用的格式。比如png或jpg。
5. 可以方便的生成缩略图。
6. 可以方便的检测图片的宽和高，确保他们满足最小限制。
7. 异步下载，效率非常高。
8. 更多：https://doc.scrapy.org/en/latest/topics/media-pipeline.html

下载图片文件-站酷图片下载实战

1. 定义一个item，上面有两个字段，一个是image_urls，一个是images。其中image_urls是用来存储图片的链接，由开发者把数据爬取下来后添加的。
2. 使用scrapy.pipelines.images.ImagesPipeline来作为数据保存的pipeline。
3. 在settings.py中设置IMAGES_SOTRE来定义图片下载的路径。
4. 如果想要有更复杂的图片保存的路径需求，可以重写ImagePipeline的file_path方法，这个方法用来返回每个图片的保存路径。

下载器中间件

下载器中间件是引擎和下载器之间通信的中间件。在这个中间件中我们可以设置代理、更换请求头等来达到反反爬虫的目的。要写下载器中间件，可以在下载器中实现两个方法。一个是process_request(self,request,spider)，这个方法是在请求发送之前会执行，还有一个是process_response(self,request,response,spider)，这个方法是数据下载到引擎之前执行。

下载器中间件

process_request(self,request,spider)：

这个方法是下载器在发送请求之前会执行的。一般可以在这个里面设置随机代理ip等。

参数：

1. request：发送请求的request对象。
2. spider：发送请求的spider对象。

返回值：

1. 返回None：如果返回None，Scrapy将继续处理该request，执行其他中间件中的相应方法，直到合适的下载器处理函数被调用。
2. 返回Response对象：Scrapy将不会调用任何其他的process_request方法，将直接返回这个response对象。已经激活的中间件的process_response()方法则会在每个response返回时被调用。
3. 返回Request对象：不再使用之前的request对象去下载数据，而是根据现在返回的request对象返回数据。
4. 如果这个方法中抛出了异常，则会调用process_exception方法。

下载器中间件

process_response(self,request,response,spider)：

这个是下载器下载的数据到引擎中间会执行的方法。

参数：

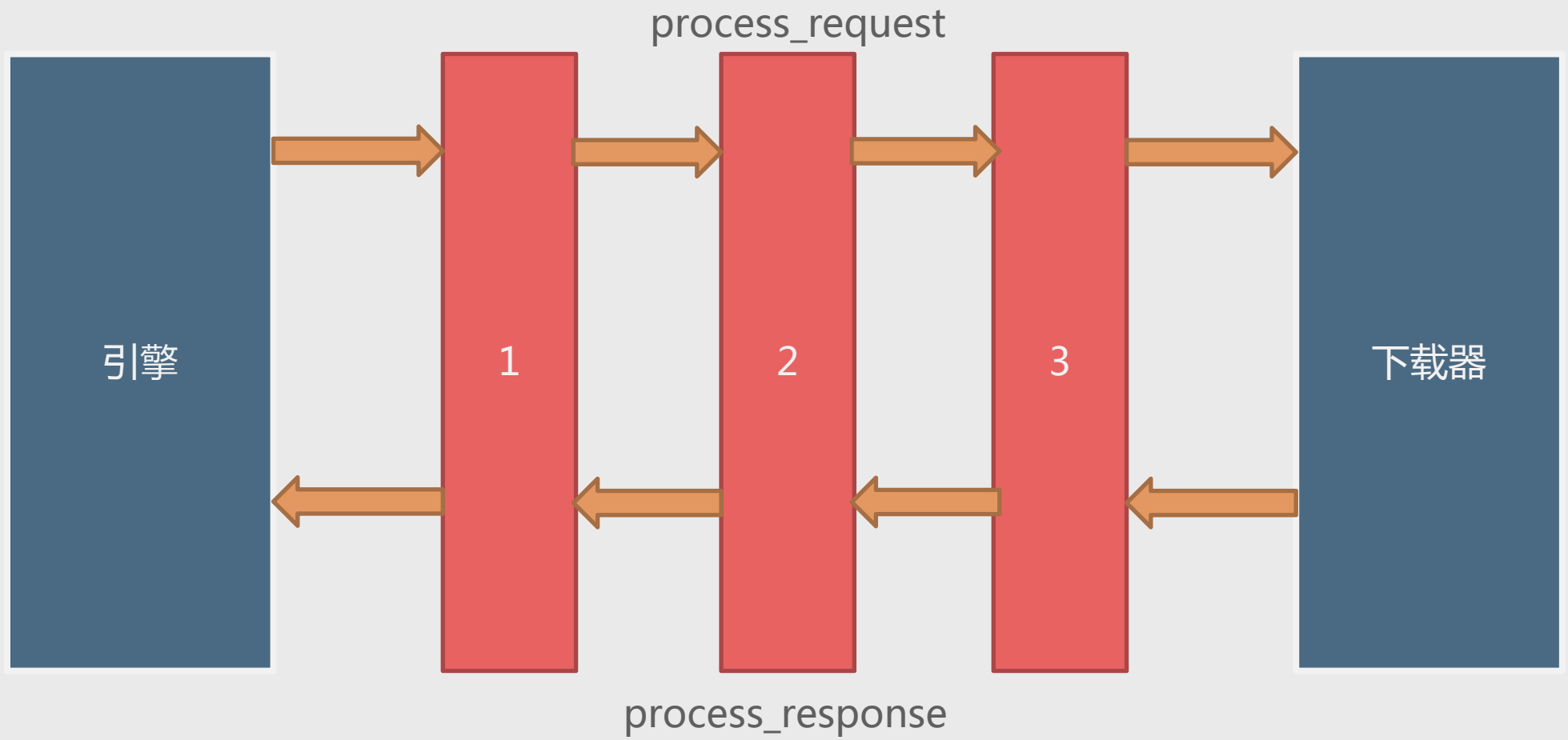
1. request：request对象。
2. response：被处理的response对象。
3. spider：spider对象。

返回值：

1. 返回Response对象：会将这个新的response对象传给其他中间件，最终传给爬虫。
2. 返回Request对象：下载器链被切断，返回的request会重新被下载器调度下载。
3. 如果抛出一个异常，那么调用request的errback方法，如果没有指定这个方法，那么会抛出一个异常。

下载器中间件

中间件



下载器中间件案例

1. 设置随机请求头案例。
2. 设置随机代理IP案例。

代理设置

在以下代理商中购买代理：

1. 芝麻代理：http://http.zhimaruanjian.com/
2. 太阳代理：http://http.taiyangruanjian.com/
3. 快代理：http://www.kuaidaili.com/
4. 讯代理：http://www.xdaili.cn/
5. 蚂蚁代理：http://www.mayidaili.com/
6. 极光代理：http://www.jiguangdaili.com/

等购买代理。

代理设置

开放代理池设置：

```
class IPProxyDownloadMiddleware(object):
    PROXIES = ["5.196.189.50:8080",]
    def process_request(self,request,spider):
        proxy = random.choice(self.PROXIES)
        print('被选中的代理：%s' % proxy)
        request.meta['proxy'] = "http://" + proxy
```

独享代理：

```
class IPProxyDownloadMiddleware(object):
    def process_request(self,request,spider):
        proxy = '121.199.6.124:16816'
        user_password = "970138074:rcdj35xx"
        request.meta['proxy'] = proxy
b64_user_password = base64.b64encode(user_password.encode('utf-8'))
        request.headers['Proxy-Authorization'] = 'Basic ' + b64_user_password.decode('utf-8')
```

部署爬虫

部署scrapy爬虫：

1. 在服务器安装scrapyd：pip install scrapyd。
2. 在客户端安装scrapy-client：pip install scrapyd-client。
3. 服务器使用命令启动服务：scrapyd
4. 在scrapy项目把cfg文件配置好：

```
[settings]
default = lianjia.settings

[deploy]
url = http://192.168.175.129:6800/
project = lianjia
```

5. 在项目中打开cmd，然后使用命令部署项目：scrapyd-deploy default –p lianjia
6. 下载curl客户端：https://curl.haxx.se/windows/
7. 部署爬虫：curl http://192.168.175.129:6800/schedule.json -p project=lianjia -p spider=house

分布式爬虫

简单理解：

之前无论是多线程爬虫，还是scrapy异步爬虫，都是在一台机器上。而分布式爬虫则是将多台主机组合起来，共同完成一个爬取任务，这将大大提高爬取的效率。

分布式爬虫优点：

1. 可以充分利用多台机器的带宽。
2. 可以充分利用多台机器的ip地址。
3. 多台机器做，爬取效率更高。

分布式爬虫必须要解决的问题：

1. 分布式爬虫是好几台机器在同时运行，如何保证不同的机器爬取页面的时候不会出现重复爬取的问题。
2. 同样，分布式爬虫在不同的机器上运行，在把数据爬完后如何保证保存在同一个地方。

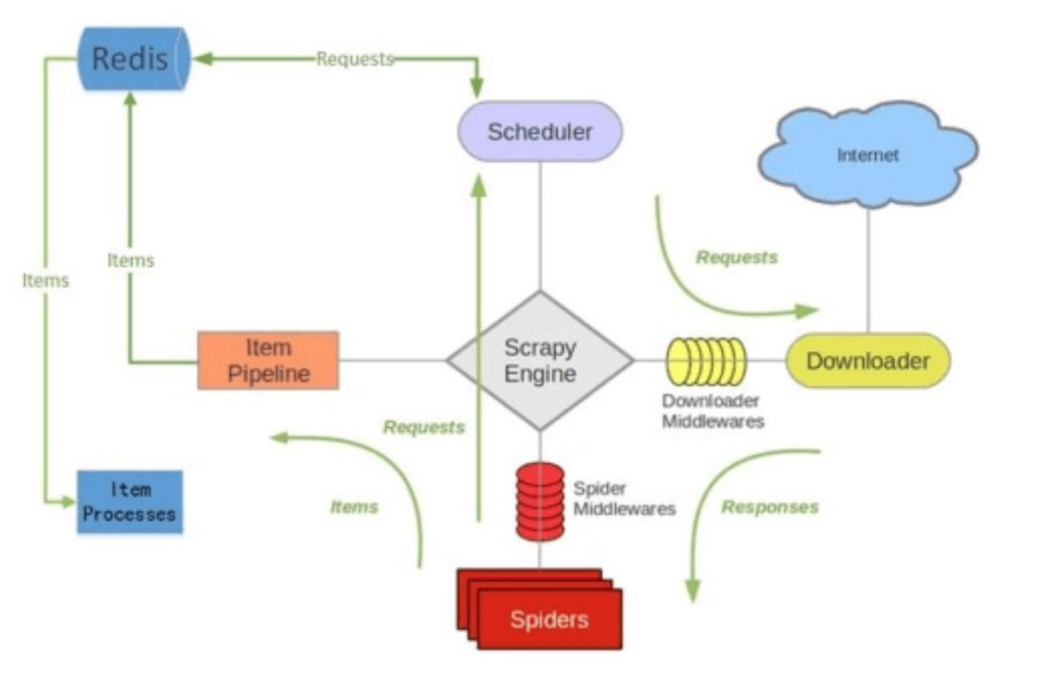
分布式爬虫

Scrapy是一个框架，他本身是不支持分布式的。如果我们想要做分布式的爬虫，就需要借助一个组件叫做Scrapy-Redis，这个组件正是利用了Redis可以分布式的功能，集成到Scrapy框架中，使得爬虫可以进行分布式。可以充分的利用资源（多个ip、更多带宽、同步爬取）来提高爬虫的爬行效率。

安装：

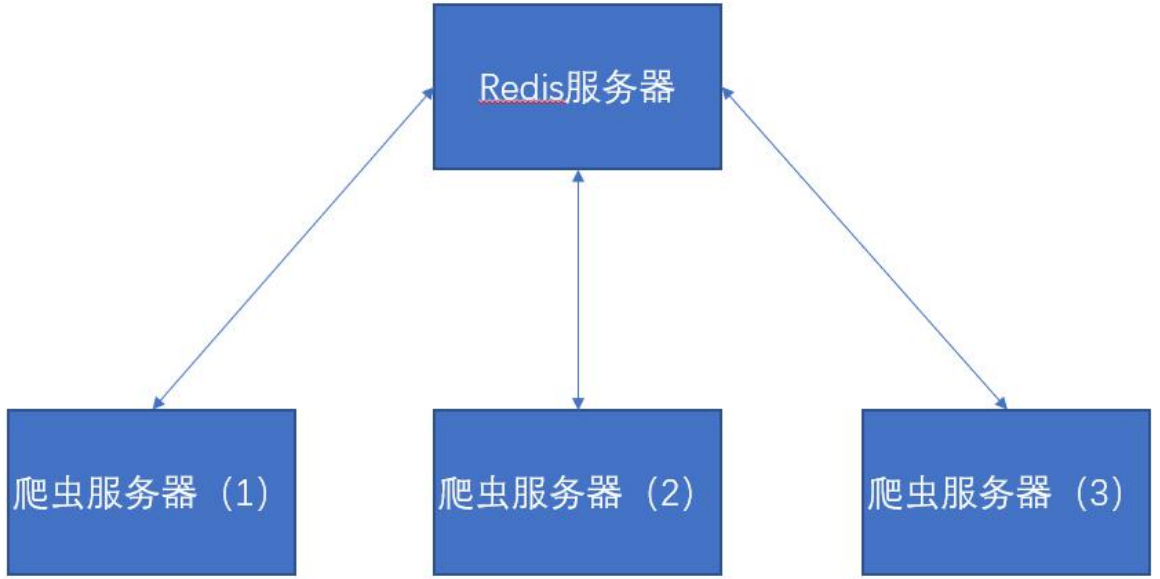
通过pip install scrapy-redis即可安装。

分布式爬虫



一、管理爬虫服务器请求的URL并去重

二、存储爬虫服务器爬下来的数据



1、从Redis获取请求

2、把爬取下来的数据发送给Redis服务器

以上两个图片对比我们可以发现。Item Pipeline在接收到数据后发送给了Redis、Scheduler调度器调度数据也是从Redis中来的、并且其实数据去重也是在Redis中做的。

Redis介绍：

redis是一种支持分布式的nosql数据库,他的数据是保存在内存中，同时redis可以定时把内存数据同步到磁盘，即可以将数据持久化，有丰富的的数据结构(string,list列表[队列和栈],set[集合],sorted set[有序集合],hash(hash表))。相关参考文档：http://redisdoc.com/index.html

在Windows上安装Redis：

1. 下载：redis官方是不支持windows操作系统的。但是微软的开源部门将redis移植到了windows上。因此下载地址不是在redis官网上。而是在github上：https://github.com/MicrosoftArchive/redis/releases。
2. 安装：点击一顿下一步安装就可以了。
3. 运行：进入到redis安装所在的路径然后执行redis-server.exe redis.windows.conf就可以运行了。
4. 连接：redis和mysql以及mongo是一样的，都提供了一个客户端进行连接。输入命令redis-cli（前提是redis安装路径已经加入到环境变量中了）就可以连接到redis服务器了。

分布式爬虫

在Ubuntu上安装Redis：

1. 安装：sudo apt-get install redis-server
2. 卸载：sudo apt-get purge --auto-remove redis-server
3. 启动：redis安装后，默认会自动启动，可以通过这个命令查看：ps aux|grep redis
4. 如果想自己手动启动，可以通过以下命令进行启动：sudo service redis-server start。停止：sudo service redis-server stop
5. 客户端链接redis：redis-cli –h [ip地址] –p [端口号]

其他机器访问本机redis服务器：

想要让其他机器访问本机的redis服务器。那么要修改/etc/redis/redis.conf的配置文件，将bind改成bind [自己的ip地址或者0.0.0.0]，其他机器才能访问。

注意：bind绑定的是本机网卡的ip地址，而不是想让其他机器连接的ip地址。如果有多块网卡，那么可以绑定多个网卡的ip地址。如果绑定到额是0.0.0.0，那么意味着其他机器可以通过本机所有的ip地址进行访问。

分布式爬虫

修改爬虫代码：

1. 将爬虫的类从scrapy.Spider变成scrapy_redis.spiders.RedisSpider；或者是从scrapy.CrawlSpider变成scrapy_redis.spiders.RedisCrawlSpider。
2. 将爬虫中的start_urls删掉。增加一个redis_key="xxx"。这个redis_key是为了以后在redis中控制爬虫启动的。爬虫的第一个url，就是在redis中通过这个发送出去的。

分布式爬虫

配置修改：

```
# 确保request存储到redis中
SCHEDULER = "scrapy_redis.scheduler.Scheduler“
# 确保所有爬虫共享相同的去重指纹
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter“
# 设置redis为item pipeline
ITEM_PIPELINES = {
    'scrapy_redis.pipelines.RedisPipeline': 300
}
# 在redis中保持scrapy-redis用到的队列，不会清理redis中的队列，从而可以实现暂停和恢复的功能。
SCHEDULER_PERSIST = True
# 设置连接redis信息
REDIS_HOST = '127.0.0.1'
REDIS_PORT = 6379
```

分布式爬虫

运行爬虫：

1. 在爬虫服务器上。进入爬虫文件所在的路径，然后输入命令：scrapy runspider [爬虫名字]。
2. 在Redis服务器上，推入一个开始的url链接：redis-cli> lpush [redis_key] start_url开始爬取。

EDU

CSDN学院　IT实战派