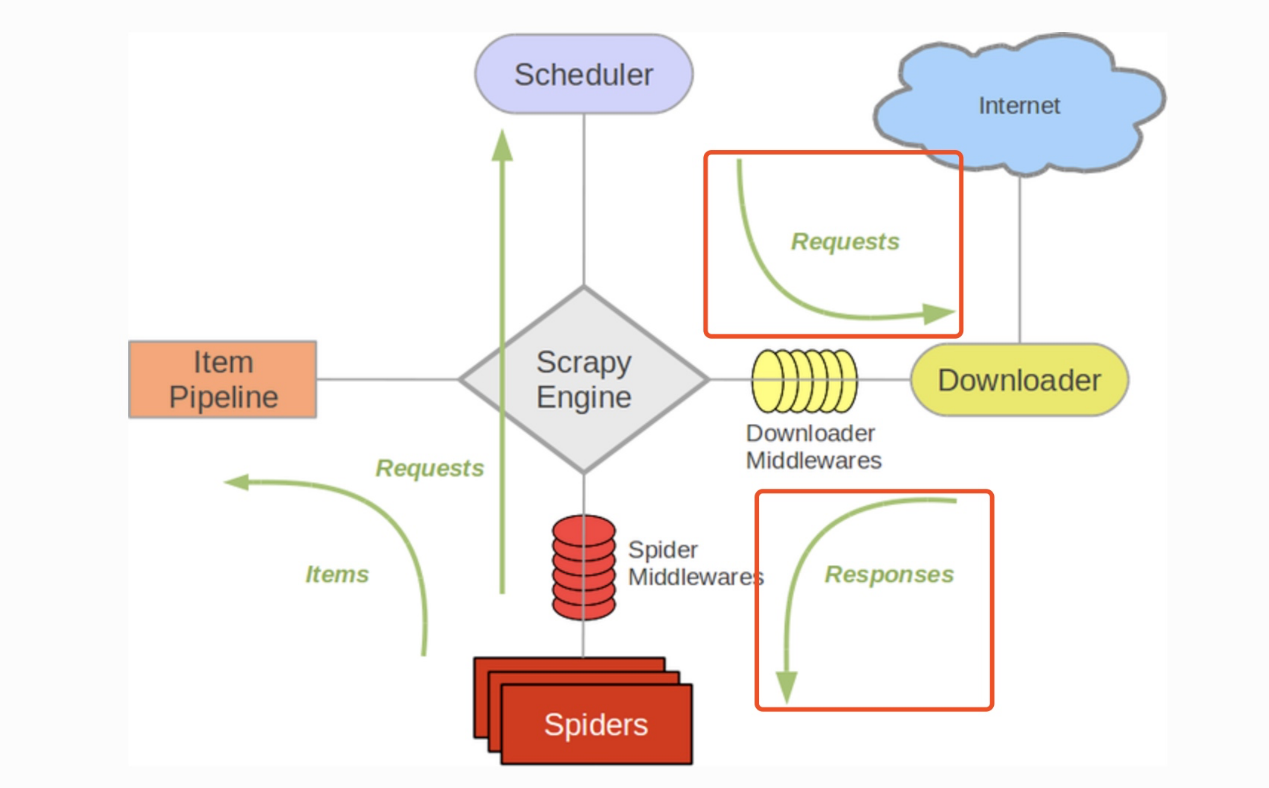
# Downloader Middlewares的用法

下载中间件是处于引擎(Engine)和下载器（DownLoader）)之间的一层组件，可以有多个下载中间件被加载运行。



* 当引擎传递请求给下载器的过程中，下载中间件可以对请求进行处理 （例如增加http header信息，增加proxy信息等）；
* 在下载器完成http请求，传递响应给引擎的过程中， 下载中间件可以对响应进行处理（例如进行gzip的解压等）

## 使用说明

需要说明的是Scrapy其实已经提供了很多Downloader Middlewares，比如失败重试、自动重定向等，他们被定义在DOWNLOADER\_MIDDLEWARES\_BASE里面。

{"scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware":100, "scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware":300, "scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware":350, "scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware":400, "scrapy.downloadermiddlewares.useragent.UserAgentMiddleware":500, "scrapy.downloadermiddlewares.retry.RetryMiddleware":550, "scrapy.downloadermiddlewares.ajaxcrawl.AjaxCrawlMiddleware":560, "scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware":580, "scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware":590,

"scrapy.downloadermiddlewares.redirect.RedirectMiddleware":600, "scrapy.downloadermiddlewares.cookies.CookiesMiddleware":700, "scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware":750, "scrapy.downloadermiddlewares.stats.DownloaderStats": 850,

"scrapy.downloadermiddlewares.httpcache.HttpCacheMiddleware": 900}

可以通过在某项目下的终端输入：

**scrapy settings --get=DOWNLOADER\_MIDDLEWARES\_BASE**

来查看该项目开启的内置Downloader Middlewares的名称和优先级。

要激活自定义的下载器中间件组件，将其加入到 DOWNLOADER\_MIDDLEWARES 设置中。 该设置是一个字典(dict)，键为中间件类的路径，值为其中间件的顺序(order)。

这里是一个例子:

DOWNLOADER\_MIDDLEWARES = {

'mySpider.middlewares.MyDownloaderMiddleware': 543,

}

## 核心方法

Scrapy内置的Downloader Middleware为Scrapy提供了基础的功能，但在项目实战里面我们往往需要单独自定义Downloader Middleware。

每个Downloader Middleware都定义了一个或多个类方法，核心方法主要有三个：

* process\_request(request, spider)
* process\_response(request, response, spider)
* process\_exception(request, exception, spider)

### 2.1 process\_request(request, spider)

当每个request通过下载中间件时，该方法被调用。该方法主要有两个参数：

* request (Request 对象) – 被处理的request
* spider (Spider 对象) – 该request对应的spider

process\_request() 必须返回以下其中之一：一个 None 、一个 Response 对象、一个 Request 对象或 raise IgnoreRequest，返回的类型不同，产生的效果也不一样:

* 如果其返回 None ，Scrapy将继续处理该request，执行其他的中间件的相应方法，直到合适的下载器处理函数(download handler)被调用， 该request被执行(其response被下载)。
* 如果其返回 Response 对象，Scrapy将不会调用 任何 其他的 process\_request() 或 process\_exception() 方法，或相应地下载函数； 其将返回该response。 已安装的中间件的 process\_response() 方法则会在每个response返回时被调用。
* 如果其返回 Request 对象，Scrapy则停止调用 process\_request方法并重新调度返回的request。当新返回的request被执行后， 相应地中间件链将会根据下载的response被调用。
* 如果其raise一个 IgnoreRequest 异常，则安装的下载中间件的 process\_exception() 方法会被调用。如果没有任何一个方法处理该异常， 则request的errback(Request.errback)方法会被调用。如果没有代码处理抛出的异常， 则该异常被忽略且不记录(不同于其他异常那样)。

### 2.2 process\_response(request, response, spider)

DownLoader获得Response之后，会先经过Downloader Middlewares。所以我们可以通过process\_response()的方法，对Response进行处理。该方法有三个参数：

* request (Request 对象) – response所对应的request
* response (Response 对象) – 被处理的response
* spider (Spider 对象) – response所对应的spider

process\_response() 必须返回以下其中之一: 返回一个 Response 对象、 返回一个 Request 对象或raise一个 IgnoreRequest 异常。

* 如果其返回一个 Response (可以与传入的response相同，也可以是全新的对象)， 该response会被在链中的其他中间件的 process\_response() 方法处理。
* 如果其返回一个 Request 对象，则中间件链停止， 返回的request会被重新调度下载。处理类似于 process\_request() 返回request所做的那样。
* 如果其抛出一个 IgnoreRequest 异常，则调用request的errback(Request.errback)。 如果没有代码处理抛出的异常，则该异常被忽略且不记录(不同于其他异常那样)。

### 2.3 process\_exception(request, exception, spider)

当下载处理器(download handler)或 process\_request() (下载中间件)抛出异常(包括 IgnoreRequest 异常)时，Scrapy调用 process\_exception()。主要有3个参数：

* request (Request 对象) – response所对应的request。
* exception (Exception 对象) – 抛出的异常。
* spider (Spider 对象) – response所对应的spider。

process\_exception() 也是返回三者中的一个: 返回 None 、 一个 Response 对象、或者一个 Request 对象。

* 如果其返回 None，Scrapy将会继续处理该异常，接着调用已安装的其他中间件的 process\_exception() 方法，直到所有中间件都被调用完毕，则调用默认的异常处理。
* 如果其返回一个 Response 对象，则已安装的中间件链的 process\_response() 方法被调用。Scrapy将不会调用任何其他中间件的 process\_exception() 方法。
* 如果其返回一个 Request 对象， 则返回的request将会被重新调用下载。这将停止中间件的 process\_exception() 方法执行，就如返回一个response的那样。 这个是非常有用的，就相当于如果我们失败了可以在这里进行一次失败的重试，例如当我们访问一个网站出现因为频繁爬取被封ip就可以在这里设置增加代理继续访问

## 实战

### 3.1 process\_request()和process\_exception()方法

新建一个项目，命令如下所示：

scrapy startproject httptest

进入项目，新建一个spider，命令如下：

scrapy genspider http httpbin.org

可以修改spider内容如下：

import scrapy

class HttpSpider(scrapy.Spider):

name = 'http'

allowed\_domains = ['httpbin.org']

start\_urls = ['http://httpbin.org/']

def parse(self, response):

self.logger.debug(response.text)

执行此spider，执行如下命令：

scrapy crawl http

获取request的response信息：

{"args":{},

"headers":{"Accept":"text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8",

"Accept-Encoding":"gzip,deflate",

"Accept-Language":"en","Connection":"close","Host":"httpbin.org",

"User-Agent":"Scrapy/1.5.0(+https://scrapy.org)"},

"origin":"116.77.73.255",

"url":"http://httpbin.org/get"}

针对middleware.py文件中，添加process\_request(request, spider)的方法：

class ProxyMiddleware(object):

logger = logging.getLogger(\_\_name\_\_)

def process\_request(self, request, spider):

self.logger.debug('启动代理！')

request.meta['proxy']='http://118.190.95.43:9001'

return None

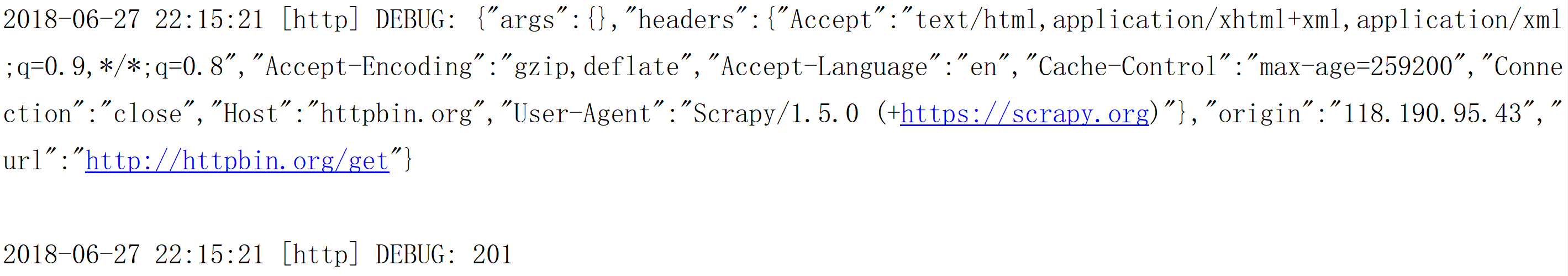
在settings里面，将DOWNLOADER\_MIDDLEWARES注释去掉，并设置如下内容：

DOWNLOADER\_MIDDLEWARES = {  
 'httptest.middlewares.ProxyMiddleware': 543,  
}

执行此spider，执行如下命令：

scrapy crawl http

得到内容如下



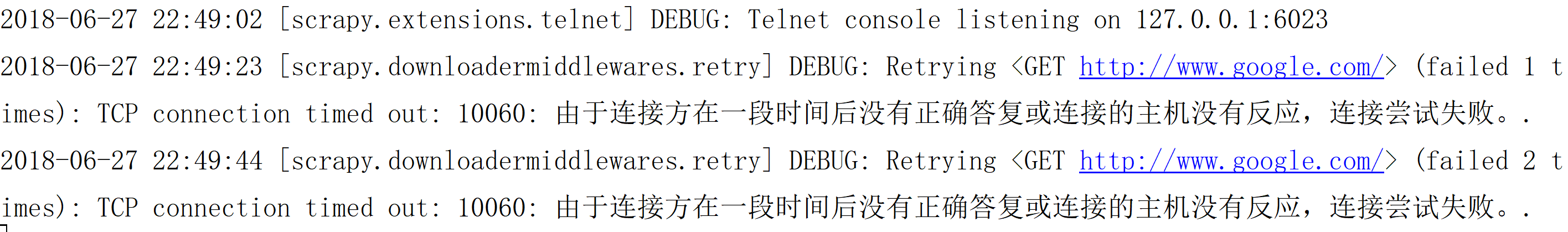
由此可见，代理已经生效，同时修改的状态码，已经生效。

### 3.2 process\_exception()方法

进入项目，新建一个spider，命令如下：

scrapy genspider google www.google.com

输出相应信息：



会进行重试。

为了不影响演示，我们将关于重试的功能关闭掉：

DOWNLOADER\_MIDDLEWARES = {

#'httptest.middlewares.ProxyMiddleware': 543,

'scrapy.downloadermiddlewares.retry.RetryMiddleware':None,

}

接下来，我们考虑用process\_exception()来实现错误重试，那么我们在spider里面修改内容如下：

import scrapy

class GoogleSpider(scrapy.Spider):

name = 'google'

allowed\_domains = ['www.google.com']

start\_urls = ['http://www.google.com/']

def make\_requests\_from\_url(self, url):

return scrapy.Request(url=url,dont\_filter=True,meta={'download\_timeout':10},callback=self.parse)

def parse(self, response):

self.logger.debug(response.text)

接下来实现，在middleware.py文件里面实现process\_exception()：

class ProxyMiddleware(object):

logger = logging.getLogger(\_\_name\_\_)

# def process\_request(self, request, spider):

# self.logger.debug('启动代理！')

# request.meta['proxy']='http://118.190.95.43:9001'

# return None

#

# def process\_response(self, request, response, spider):

# response.status=201

# return response

def process\_exception(self, request, exception, spider):

self.logger.debug('异常处理')

return request

在settings里面，将DOWNLOADER\_MIDDLEWARES设置如下内容：

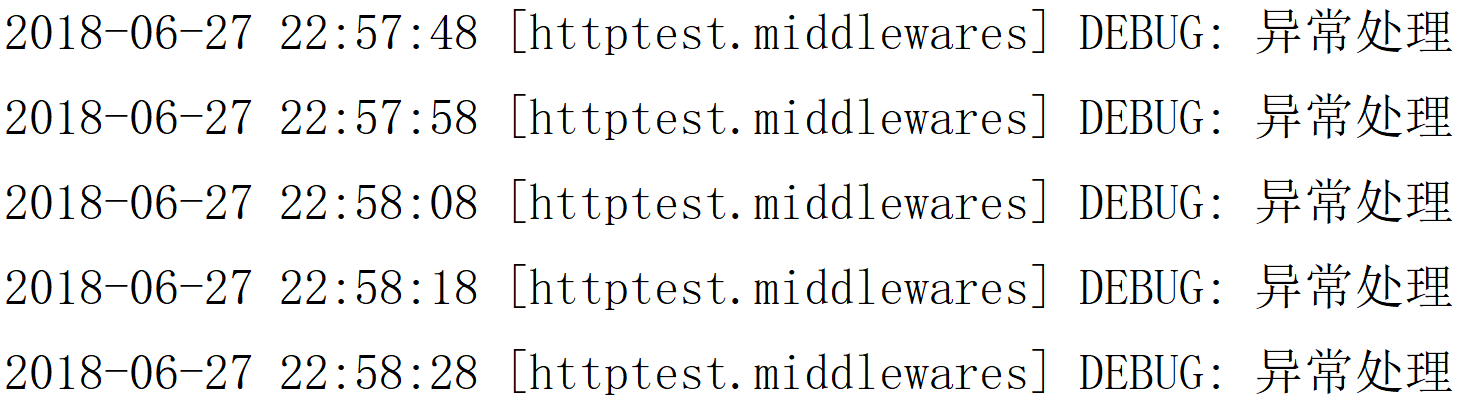
DOWNLOADER\_MIDDLEWARES = {

'httptest.middlewares.ProxyMiddleware': 543,

'scrapy.downloadermiddlewares.retry.RetryMiddleware':None,

}

得到的输出如下：



说明进入了process\_exception()方法里面，会不断进行错误重试。