Scrapy源码解读一

关于scrapy，基本的安装，指令就说了，虽然标题是源码解读，但也只是大致谈下工作中对scrapy的理解。

平常我们听到关于scrapy，最多就是可能就是，这是一个基于Twisted的异步网络爬虫框架，效率较高等等。在介绍之前，我们确实是需要了解一些关于Scrapy框架的基础知识，但是不是Twised这种网络异步框架，只是针对Scrapy的核心模块，工作流程，等了解足够深并仍然有兴趣的，可以再研究研究这个Twisted框架，也是个巨无霸。

核心模块，按照正常工作流程，从前往后分别是：引擎、爬虫、调度器、(处理请求的)下载中间件、下载器、(处理回复的)下载中间件、(处理回复的)爬虫中间件、爬虫、(处理结果的)爬虫中间件、管道。

其中管理整个流程的模块就是引擎，对上面流程进行大致讲解：引擎激活后，从爬虫spider中获取目标url，然后把他传递到调度器进行统一管理，当初始定义的目标url都推到调度器之后，引擎会从调度器中推出目标url，将其传递给下载器，在与下载器交互的过程中，会通过一个中间件，对数据进行一些基本的判断处理。当确实拿到了正确的回复response的时候，再将结果传递到爬虫，进行解析工作，当然了，中间又少不了一些中间件，当我们解析出结果后，就进入到了管道，接着就是对数据的相关序列化处理。

不知道在上面流程中，会不会有一些看不大懂的词，我们可以在详细的解剖下这个流程，模拟下scrapy的工作。

首先是引擎的管理，不知道你对这个是怎么想的，是不是第一次听都会感觉很厉害！虽然这个写的确实牛，我们不一定能完整的实现出来，但他的大致思路，我们还需可以了解的，我们举个不太恰当的例子：

有一件这样的事：男人外出买菜 -> 女人煮饭 -> 男人开吃

那么我们现在就需要定义三个对象：Man、Women、Kitchen，分布表示男人、女人、厨房，那我们现在来大致描述下这个流程

Class Kitchen:

@staticmethod

Def process\_food(food):

Return food

Class Man:

Def shopping(self):

Return [1] // 数字就表示事物把

Def eat(self):

Return None

Class Women:

Def cooking(self, food):

Return Kitchen.process\_food(food)

我们现在要来实现，去”管理”这个流程，我们就把管理者叫做引擎。这里的管理，是对已有模块的一次再封装，实现通过”一个”模块，能够实现对整个模块的处理。下面就是，我只需要work一下，这个流程或许就可以跑起来了，或许也跑不起来，，，不过影响不大。

Class Engine；

Def \_\_init\_\_(self, man, women, kitchen):

Self.man = man

Self.women = women

Self.kitchen = kitchen

Self.list = []

Def work(self):

Food = Self.man.shopping()

Self.list.extend(Food)

For f in self.list:

goodFood = self.women.cooking(food)

Self.man.eat(goodFood )

这个例子不恰当，也没有什么意义啊，只是我曾经被”管理”一词吓到过，，，所以，，，瞎写的，哈哈哈。