|  |  |  |  |
| --- | --- | --- | --- |
| 文件编号 |  | 作　　者 | 张淼 |
| 文档版本 | 1.0 | 最后修改日期 | 2019/03/15 |

**数据中心**

**版本号1.0**

**技术文档**

**（scrapy 去重机制源码剖析）**

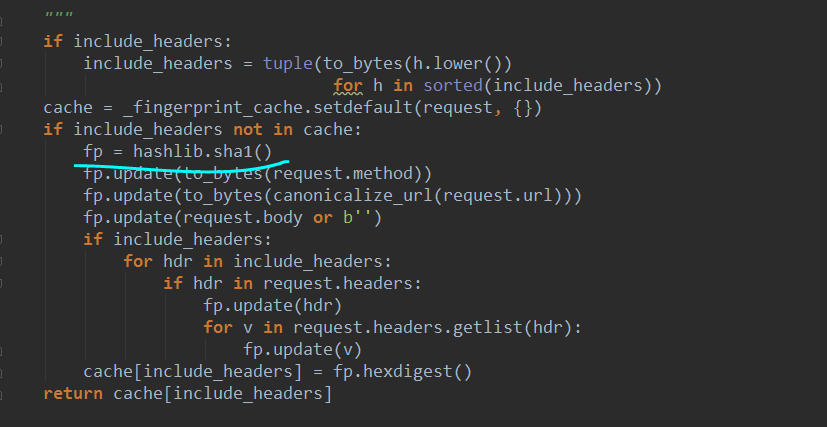
**编写人：张淼**

**编写时间：2019/3/15**

写在前面的

之前在编写Scrapy爬虫的时候，可能是对源码和一些高级机制不太熟悉，说实在，我是有些信不过scrapy的，我总怕scrapy框架会骗我，但毋庸置疑的，scrapy肯定是个优良的框架，无论是设计思想还是实现，所以我想，如果能把scrapy的优良特点加到我们的爬虫系统中，岂不是一件不错的事情。

Request\_fingerprint是去重机制中最关键的一步，这就是传说中的数据指纹，我划青线的地方，是hash加密算法不是创建某个复杂的对象，关于这一点这个算法，说白了它会比md5更好，他是基本上无法被反向破解的，但有个缺点就是生成时间会比较长，所以非常适合被用于做数据指纹。



在去重机制中，数据指纹起的作用，是对比，对比是否抓取到了相同的url，而这里就涉及到了另一个问题，就比如以下的两个链接，

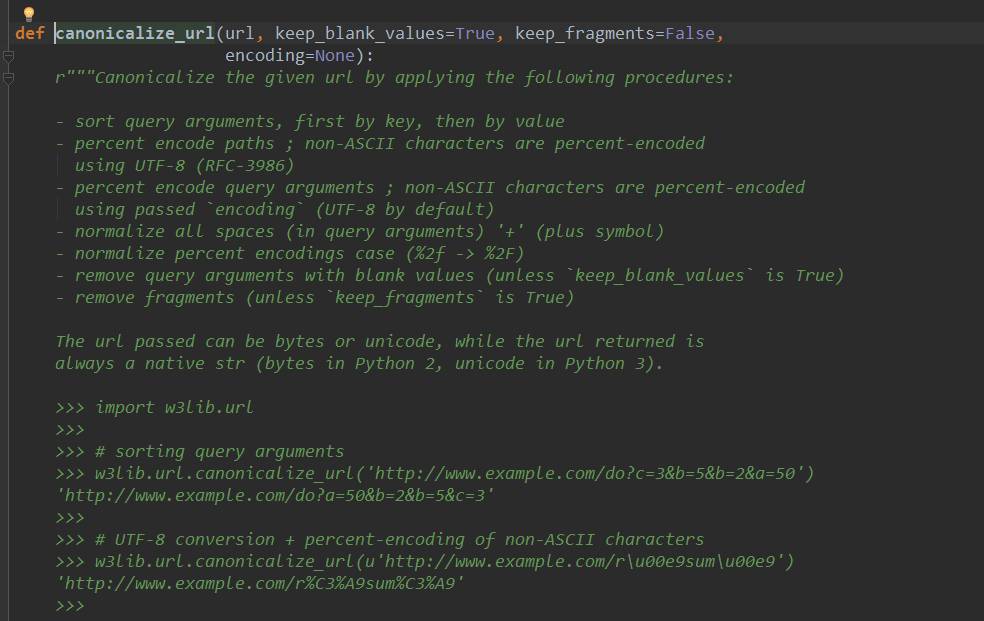
http://www.example.com/query?id=111&cat=222

<http://www.example.com/query?cat=222&id=111>

这是同一个链接，但如果只是简单的去重那么同一个页面就会被重抓一次

因此，scrapy这里的处理是

1. 将请求方式转换成二进制码，在通过hash算法加密
2. 将url地址按标准拆分，并重新组合再通过hash算法加密



1. 如果存在body就通过hash算法加密
2. 如果该请求存在headers 那么拆分再通过hash算法加密headers的信息

然后就是存储了，把经过加密后的数据存储起来以供以后对比使用

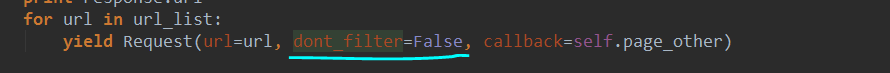
存储scrapy这里使用的是弱引用类型创建的字典，好处自然不言而喻，我之前还在担心，如果存储于内存用于对比的数据多了，内存会不会爆掉，这么看来应该不会

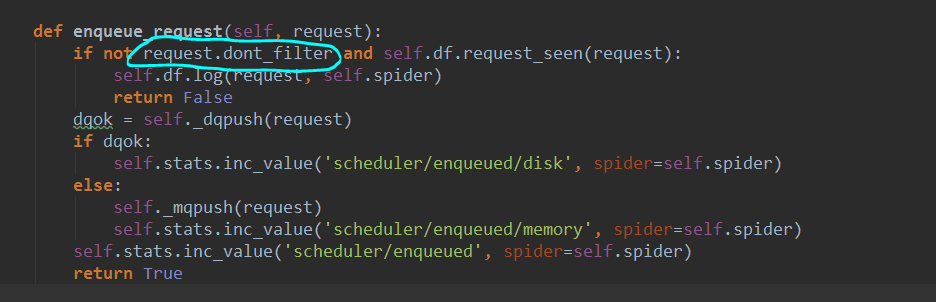


但是去重的精度，可能会受到影响，关于这一点，scrapy\_redis给出了另外一个答案，这一点还待探究

关键部分说明了，但请求是如何过来的呢?

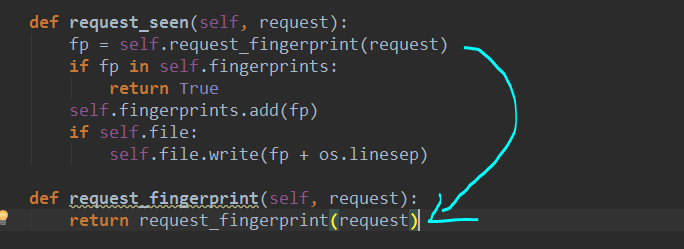
主要是通过这个方法， 在请求的过程中如果dont\_filter=False, 当请求访问enqueue\_request





这个方法的时候就会调用到去重控制模块dupefilters中的request\_seen

有了request\_seen ，然后就有了光....



这样通过request\_seen逐步调用到了数据指纹的主方法，主方法返回一个弱类型字典和已有的set()中的内容进行对比，这一步设计的真够精妙，如果内存不够，会自动清除字典内容，如果存在则返回True,告诉前面的方法，该url信息已经被抓取过

当然我相信scrapy\_redis有更好的实现，因为这种方式虽好，但不能持久化存储已经抓过的链接，万一需要断点续爬，那这个内存机制就失效了

不过说到断点续爬，据说scrapy也有不错的实现，我也非常期待scrapy带给我新的惊喜，那么断点续爬就且听下回分解了。