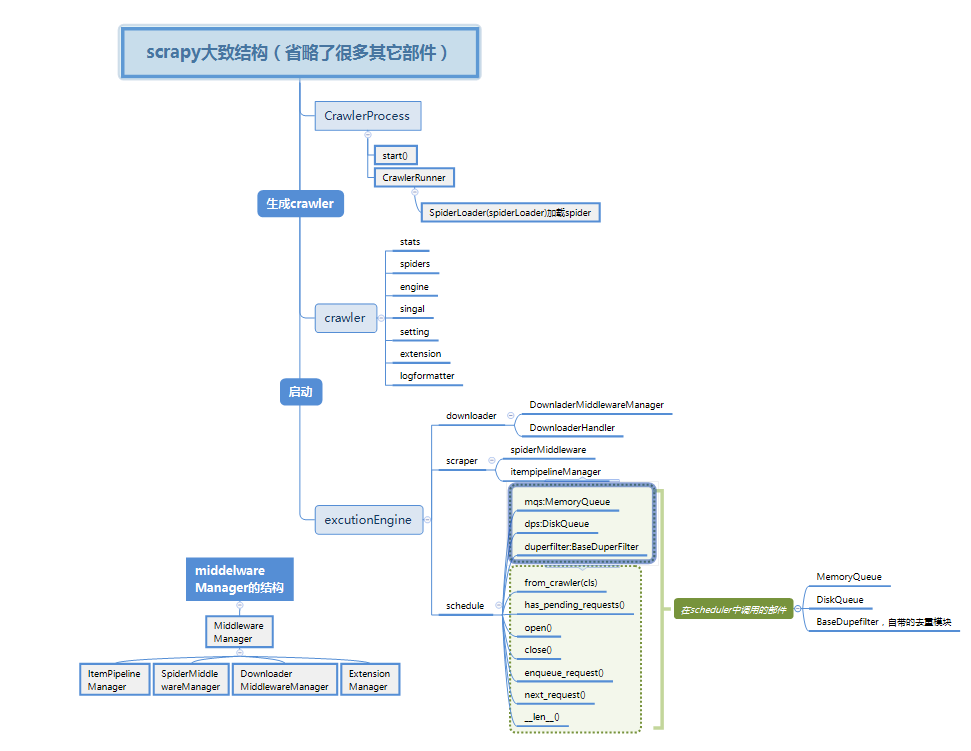
Scrapy的启动流程：

一般来说scrapy的本身的结构是足够满足威胁情报的爬虫需求的，但是说不定将来也会有一些未知的要求，所以这里简单介绍了一下scrpay的大致框架，并不严谨，如果能起到让大家了解的作用的话，我也满意了。

执行scrapy crawl spider命令之后，生成一个crawler实例，由这个crawler实例生成一个crawlerprocess对象，之后由这个crawlerprocess启动loader\_spider来启动一系列的爬虫，之后初始化schedule，downloader，scraper：



其中schedule负责任务调度，比如对请求的任务的队列进行管理，去重处理等的功能。Scraper负责数据的后期处理，比如对后期得到的item的处理，downloader负责对请求进行下载。

**Schedule**

1、任务的队列管理由两个操作负责，是enqueue\_request()和next\_request()，分别负责任务的入队和出队；

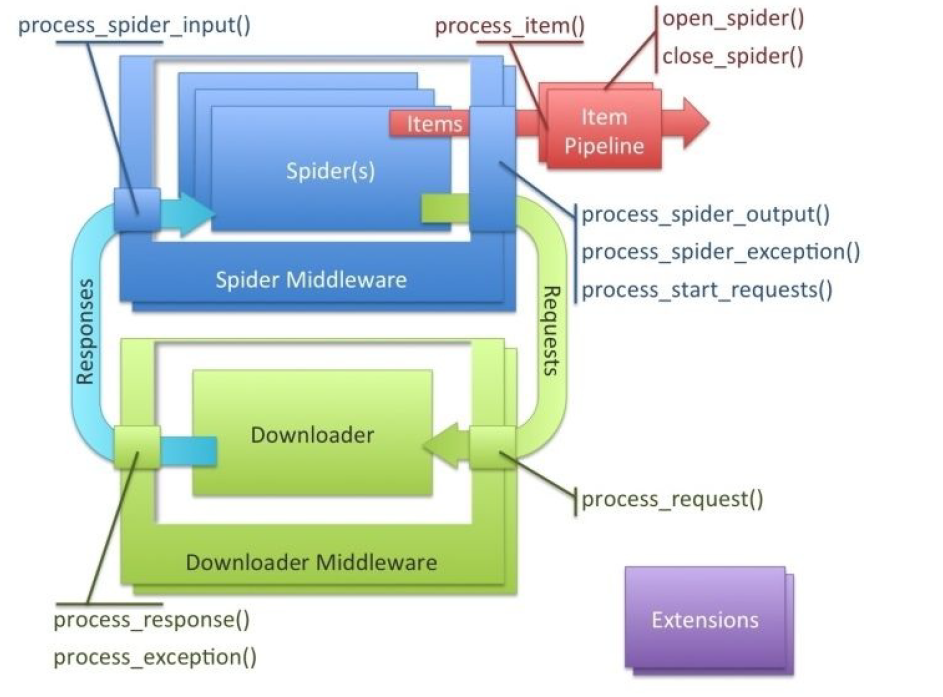
2、去重功能BaseDuperfilter负责，通过采取request的指纹，放在set集合中，每次调用next\_request的时候，检查一下请求是否已经存在了，但是这个去重的set好像生命周期也就是next\_request的一个周期；

3、scrapy的统计功能由StatsCollector对象来负责，在crawler生成时指定。

**Scraper：**  
spiderMiddleware由middlewareManager这个类来调用，初始化方法是from\_setting，里边通过setting.py文件中的对相应的字符串描述定位类所在的位置，通过laod\_object()方法（scrapy的）来加载对应的类，之后通过其中的from\_setting方法（千篇一律）来生成middleware实例。（为什么scrapy框架中总喜欢用from\_setting方法），调用是这个类中的download。

方法：process\_spider\_input(response, spider)： 处理经过spiderMiddleware处理后的response，下一步就是交给spider来处理。

方法：process\_spider\_output(response, result, spider): 处理spider执行完后扔出来的结果，里边的result可以是dict，也可以是item



Scrapy里边有很多默认的中间件，在default\_setting.py文件中，官方api也可以查阅到，包括一些404，302跳转等网络情况的处理

**Pipeline：**

Pipeline中的方法由ItemPipelineManager来调用，ItemPipelineManager继承自middlewareManager：

方法：process\_item(self,item,spider)常见的item处理都放在这里边。

方法：open\_spider(self,spider)开启一个spider后默认会执行的操作，常用来写入数据库初始化操作。

方法：close\_spider(self,spider)常用来写入数据库关闭操作

问题：pipeline中的的数据库是每次初始化的时候建立，还是说每次运行到pipeline中的对应的方法的时候，都会重新生成一次。（明天测试一下）

Pipeline中也有很多系统自带的pipeline，比如imagePipeline,可以用来下载图片。

当由item被drop掉了之后，如果想得到被丢弃掉的item，可以在middleware初始化的时候将dropitem与一个信号绑定在一起。（需要详细描述）

Scrapy-redis简介：

大致上模块的名字和功能都与scrapy中相应的模块是一样的，不过将去重放在了redis中，同时任务队列也都放在了redis中。Response和item的处理也都放在了redis服务器中，这些实例对象使用的是pickle方法。

我在远程访问reids的时候要求用带密码的访问方式，所以有了下边这个redis\_url

使用redis需要在scrapy项目的setting中设置：

REDIS\_URL='redis://user:asd123456@127.0.0.1:6379'#其中个asd123456是redis的权限密码

# REDIS\_HOST="127.0.0.1"

# REDIS\_PORT='6379'

在没有redis\_url的时候，可以用上述的redis\_host(redis的服务器ip),redis\_port(redis的服务器端口)

SCHEDULER="scrapy\_redis.scheduler.Scheduler"

DUPEFILTER\_CLASS="scrapy\_redis.dupefilter.RFPDuperFilter"

ITEM\_PIPELINES={

'scrapy\_redis.pipelines.RedisPipeline':300

}

重新申明调度器位置，去重处理的位置，以及将数据推送到redis中的设置

（需要详细补充）

Scrapy的一些高级用法：

Itemloader的用法：

Itemloader的add(xpath)和add(css)的语法

这样写的好处：不会出现extract之后的strip方法出错或者其他什么的，没有取到就没有这个字段，也可以添加自定义函数，来设置默认值,缺点：不能使用re功能。（可以使用re，可以在pipeline中设置default值），itemloader中的函数时以\*args的方式传入的，所以很多个函数时以列表的形式传入，执行顺序也是按照list的顺序来。

Itemloader中设置函数的方式，compose和mapcompose的简介，以及scrapy。Field中写方法的技巧，这样可以单独对某一个字段来出来，pipeline会处理所有的爬虫扔出来的item，而在scrpy。Field中泽可以抓门对某一个爬虫的item做特殊处理。Mapcompose貌似并没有太多需要注意的地方，其实也是一个list中存放一些函数。

Scrapy.Field()的特殊写法。

Item.get()方法，这样可以设置默认值

爬虫中的try和except的标准写法，log模块的设计

当所有爬虫写好之后，需要使用scrapyd来部署scrapy，操作起来比较简单，网上都有比较详细的帮组。我在linux下实现没有问题，在windows下因为路径名编码的原因导致出错，没能部署成功，后续我会将windows的部署方案给上。

文件保存操作！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！