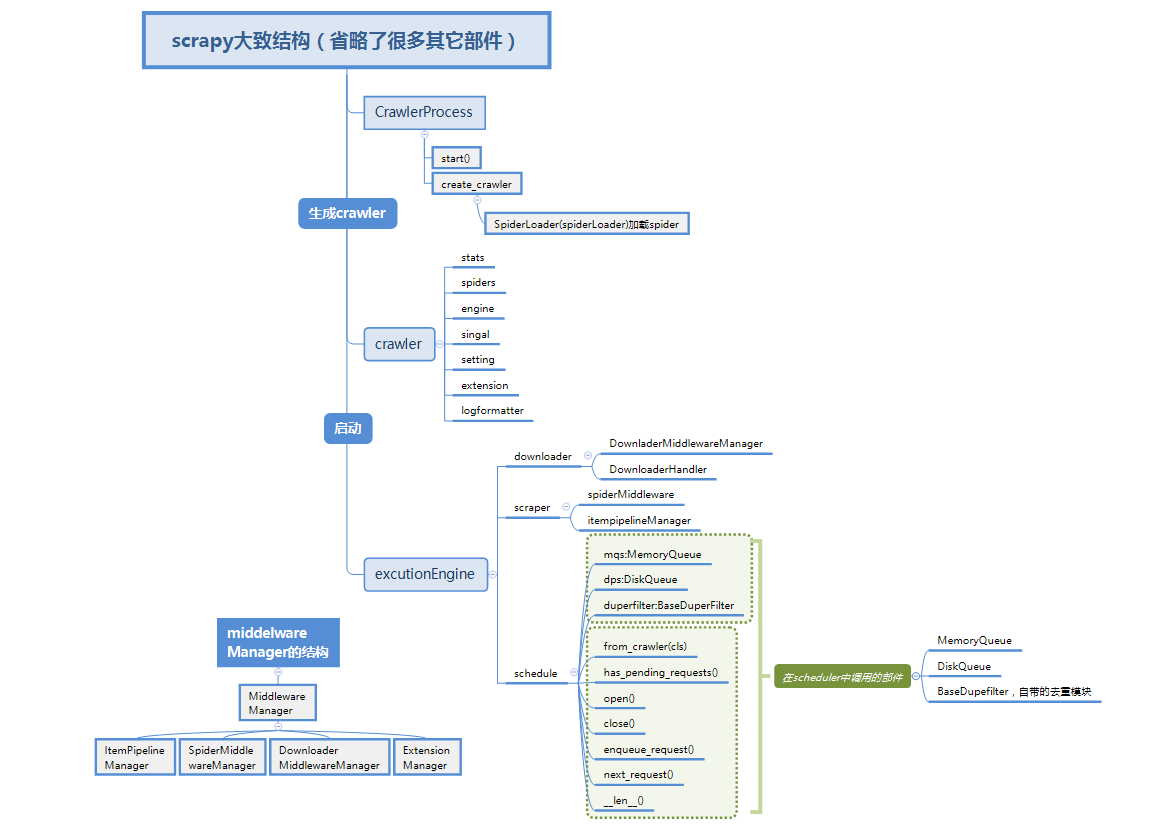
**Scrapy的启动流程：**

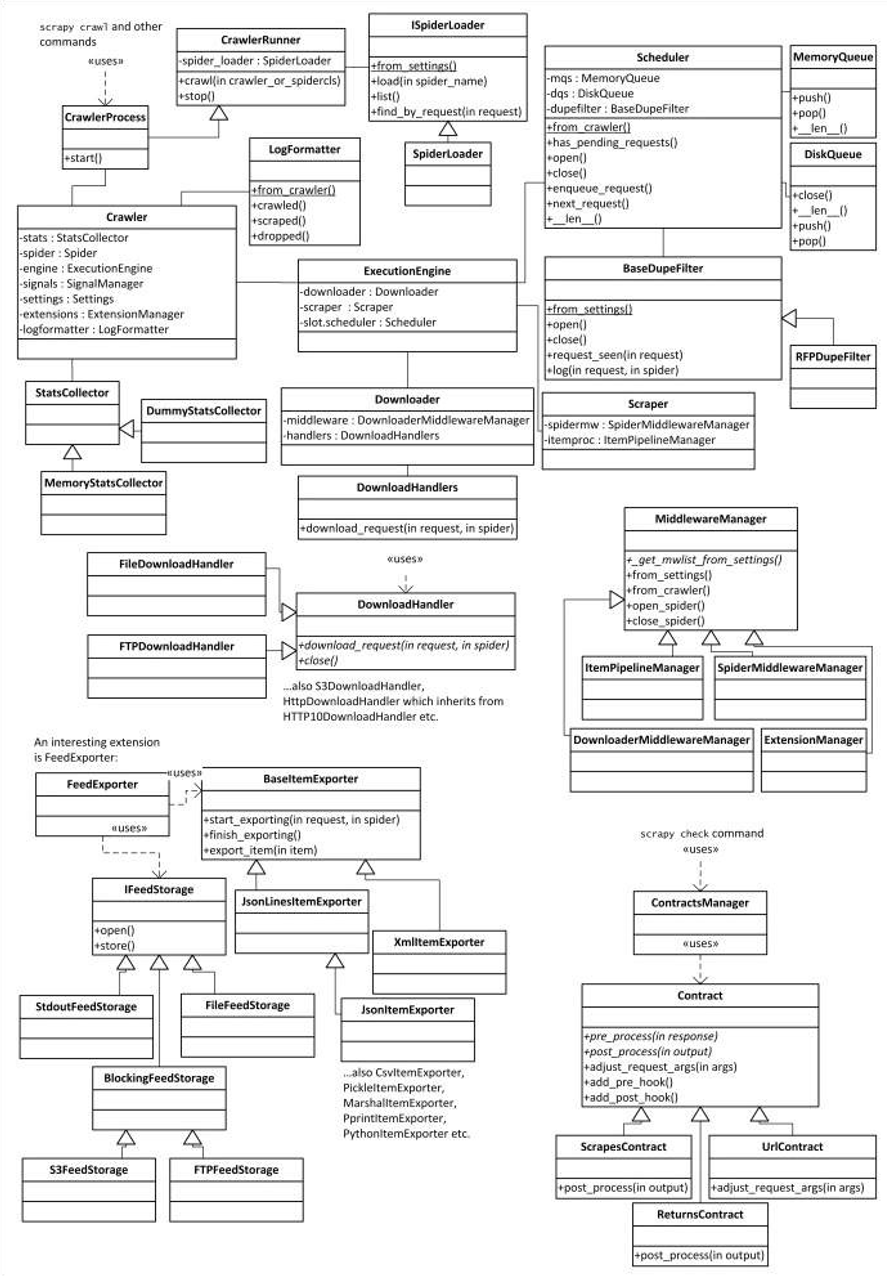
一般来说scrapy的本身的结构是足够满足威胁情报的爬虫需求的，但是说不定将来也会有一些未知的要求，所以这里简单介绍了一下scrpay的大致框架，并不严谨，如果能起到让大家了解的作用的话，算是抛砖引玉。

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。

下边是一个相对比较完整的scrapy模块之间的关系图：



**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；

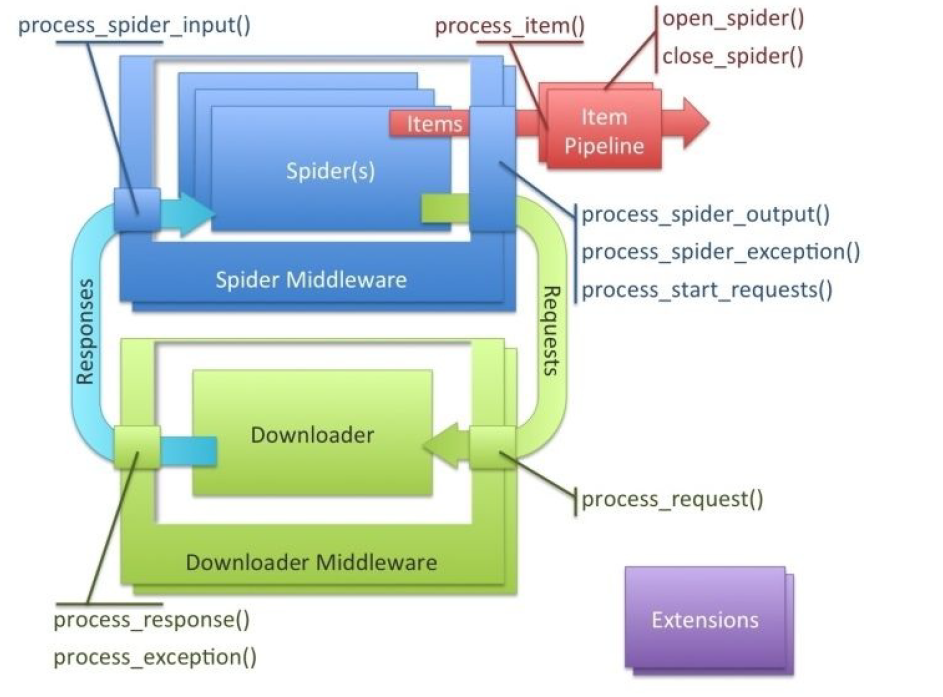
2、去重功能BaseDuperfilter负责，通过采取request的指纹，放在set集合中，每次调用next\_request的时候，检查一下请求是否已经存在了，但是这个去重的set好像生命周期不长。不过没有使用scrapy-redis框架，一般scrapy的去重功能会放在middleware中，对request做去重处理；

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来来定位类所在的位置（'ThreatArticleCollector.pipelines.ThreatcollectPipeline':71），通过laod\_object()方法（scrapy的）来加载对应的类，之后通过其中的from\_setting方法来生成middleware实例。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中的类由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。（网上有较详细的介绍）

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，要求客户端用带密码的访问方式，所以只有写成redis\_url这种，因为REDIS\_HOST和REDIS\_PORT这中写法，没有写链接授权码的地方。

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）

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助文档。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！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！