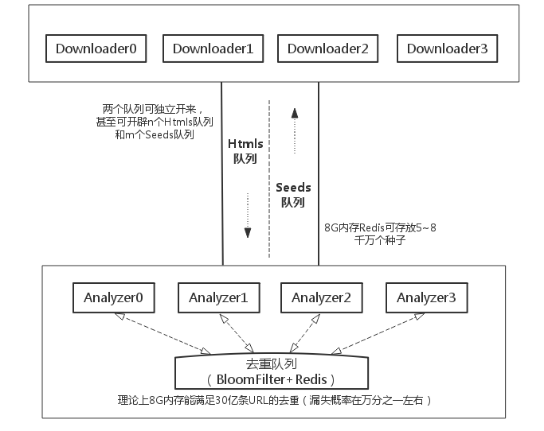
**架构：**



**说明：**

1. MSpiders框架主要是针对分布式、多爬虫扩展而设计的。
2. 框架分下载器(MSpiders\_Downloader)和解析器(MSpiders\_Analyzer)两部分，两者通过Htmls队列和Seeds队列进行数据互通。
3. 爬虫流程：下载器从Seeds队列拿到一条种子（主要包含URL等信息），根据种子信息调用相应的下载规则进行网页抓取，拿到网页内容后封装成html包（主要含下载到的网页内容等信息），加入到Htmls队列；解析器从Htmls队列拿到一个html包，根据包里所带信息，调用相应的解析规则，将目标字段入库，并封装新的待爬URL，作为一条种子加入到Seeds队列。
4. 分布式：下载器和解析器的运行是相互独立的，可以开启不同数量的下载器和解析器，也可以各自部署到不同机器。下载器主要耗带宽资源，解析器主要耗CPU，这样的设计方便分布式扩展，也可以最大化地利用各机器资源。甚至可以白天增加下载器，晚上增加解析器，合理规避网站反爬虫机制。
5. 多爬虫扩展：新增一个爬虫，只需要在下载器里写一个下载规则，在解析器里写一个解析规则，即可。其他公用的像去重、入库、DNS解析缓存、日志监控等模块都已经封装好，节省大量的重复开发，同时也方便管理。
6. 其他功能：多爬虫框架中各爬虫公用队列，减缓了单个网站的抓取频率，降低了反防爬的压力。其外的，框架可以支持优先级抓取，对某个网站设置优先级，可优先抓取。
7. Htmls队列和Seeds队列同样也可以独立拆分，部署在不同机器上的Redis，各自的数量也是没有限制的。需要注意的是：所有解析器必须共用一个去重队列。

**如何新增一个爬虫：**

1. MSpiders目录下有一个 builder.py 文件，命令行执行 ”python builder \*\*\*” 新建一个爬虫(\*\*\*为爬虫名字)。
2. 下载规则编写：
3. 进入 Spider\_\*\*\* 文件夹，里面包含settings.py 和 spiders.py 两个文件，在 settings.py 里设置网络延迟和网络超时时间，在 spiders.py 里编写下载规则。
4. self.visit()方法负责网络请求，可带上各种参数，例如表头、Cookie、代理、超时等。默认为get请求，如果设置了data参数，则会改为post请求。如果请求返回的结果为空，表示请求失败；否则请求结果为一个字典，包含”html”和”status\_code”两个键，即网络请求返回的网页内容和状态码。
5. self.logger可以输出日志信息，建议编写下载规则时注意做好异常监控。
6. 下载规则要返回一个字典，里面至少包含”html”键和”parser”键，即抓取到的网页内容和要调用的解析规则名称。”parser”的值可以为”类名.方法名”，也可为”类名”(默认调用parse0()方法)。
7. 如果要加代理和Cookie，每个爬虫要自己维护好。
8. 解析规则编写：
9. 进入 Spider\_\*\*\* 文件夹，里面包含 settings.py 和 parsers.py 两个文件，在 settings.py 里设置数据库信息，在 parsers.py 里编写解析规则。
10. 解析规则要返回两个列表，第一个列表放待入库的目标数据，第二个放待爬种子。每一条目标数据/待爬种子，为一个字典。
11. 每一条待入库的目标数据，至少要包含键”pipline\_dbType”(值为mongo/es/redis/hbase/ssdb/mysql取其一)，可以带上键 ”pipline\_keyName”、”pipline\_collection”、”pipline\_index”、”pipline\_doc\_type”，如果没有带上这几个配置信息，程序会从 settings.py 中读取，如果 settings.py 里面也没有设置，则使用 “default” 。
12. 每一条待爬种子，至少要包含键”url”和”spider”，分别代表待爬URL和对应的下载规则。”spider”的值可以为”类名.方法名”，也可为”类名”(默认调用crawl0()方法)。
13. 建议做好异常处理和日志监控。
14. 下载器拿到种子才能抓取网页，产生html包；解析器要根据html包才能解析生成新的待爬种子。两者是互相依赖的，所以加入一个爬虫，我们还需要手动插入一个起始种子。