**监控工具开发文档**

1. **项目环境**

系统：Window

程序：python3.6

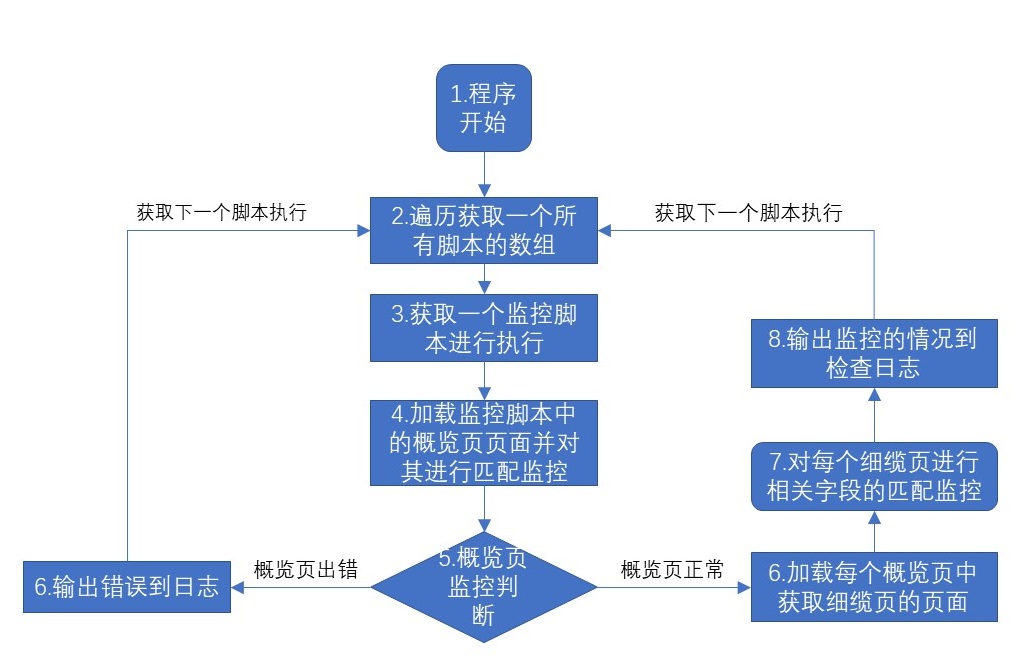
框架：scrapy1.5

1. **监控的原理**

目前监控主要的原理是通过验证包裹字段的标签是否存在，来判断网页结构是否发生变化，进而达到监控的目的。对于新增的标签内容无法监控。

通过为每个采集点写一个爬虫脚本，用脚本去解析采集点的首页和细缆页达到监控的目的。首页解析其包含的细缆链接，细缆页解析每篇新闻所需的特定字段。 解析的方式有xpath路径，css样式，正则表达式。原网页与该字段相关的网页结构发生变化后，通过日志输出即可发现网页的结构变化。最后把所有采集点的监控日志汇总到一个当天的日志文件。

1. **程序流程和项目结构 流程**

****

1. **开始：**通过脚本或者接口的方式启动监控程序。
2. **获取所有脚本：**获取监控程序中的所有监控脚本的一个数组，数组中的脚本按照字母顺序排序。
3. **获取一个脚本执行：**按照顺序获取一个监控脚本进行执行。
4. **概览页监控：**使用脚本中的xpath（标签的路径定位）、css（标签的样式定位）或书写正则表达式 去匹配概览页，看概览页是否正常。监控中能出现的情况有：正常，细缆页无法访问，细缆页结构改变）
5. **对概览页的监控情况进行判断：**

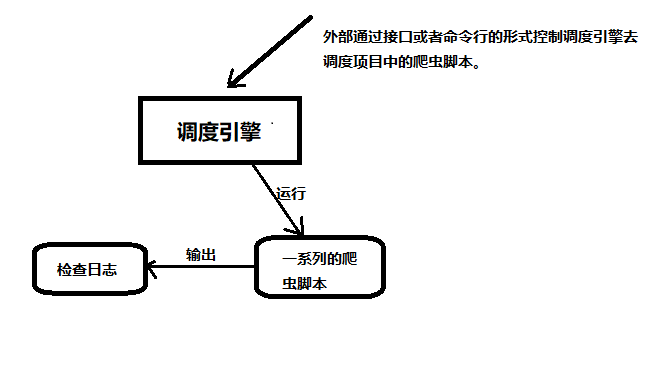
**不正常**（结构改变，网络异常无法访问），导致后面的细缆检查无法进行，直接输出错误到检查日志，输出的内容为概览页异常，细缆页未检查。获取下一个监控脚本执行。

**正常：**进行后面的流程

1. **获取所有细缆页的链接**：在概览页正常的情况下，获取到了所有细缆页的链接。
2. **加载细缆页并进行监控**：监控最多不超过20个细缆页。使用xpath、css或正则去匹配细缆页中包裹（标题，时间，来源，作者，正文，引题，副题）字段的标签。监控中能出现的情况有：正常，细缆页无法加载访问，细缆页结构改变）
3. **监控结果输出到日志文件**：把针对这个采集点的监控情况输出到日志。进行下一个脚本。

注：针对多模板的网页，采用采集多个特定点【例：author、author1...在判断是利用or只要一项成立即可判断为原网页未变化（存在一定误报）】。对于外链链接直接在采集过程中排除即可。对于无法用xpath定位的数据，例如网页源码中用js加载的数据，可采用正则的方式定位所要监控内容的位置。

**结构**



1. 调度引擎为一个脚本汇总的框架接口，控制项目中爬虫脚本的运行。
2. 爬虫脚本：每个采集点有其相应的一个脚本，脚本的命名不可重复，建议以采集点的全拼进行命名。
3. 检查日志：为所有采集点的检查日志汇总。

**4．目录结构**

NewsCrawl为项目的名字。

NewsCrawl\NewsCrawl\items.py：定义了要取什么字段

NewsCrawl\NewsCrawl\spiders\：采集点监控脚本的文件夹(里面放置抓取采集点内容的.py文件)

NewsCrawl\log.txt；error\_log.txt ：日志文件分为正确日志以及报错日志

NewsCrawl\NewsCrawl\pipelines.py：对采集点特定字段进行判断的文件，并输出判断结果到日志文件。

NewsCrawl\NewsCrawl\settings.py：设置文件，可以设置ip代理，请求头，下载延迟，请求重试次数，并发量等等设置。

NewsCrawl\NewsCrawl\write.py：日志输出格式

**5.日志部分**

a.日志目录：NewsCrawl\log.txt;error\_log.txt

b.log.txt用于写入正确日志，error\_log.txt用于写入报错日志。日志输出为所有细缆页的监控情况。

c.字段说明:

log.txt：网站名|栏目别名+url+特定字段（中文）

error\_log.txt：网站名|栏目别名+url+时间+特定字段（中文）

**6.爬虫脚本的书写规范**

Spider是由自己来定义的Class类，Scrapy用它来从网页里抓取内容，并将抓取的结果进行解析。不过这个Class必须要继承Scrapy提供的Spider类scrapy.Spider，并且还要定义Spider的名称和起始请求以及怎样处理爬取后的结果的方法。

，并返回结果  
 item = {}  
 indexLen = len(response.xpath('//ul[@class="news\_list"]/li/a').extract())

#监控起始点（概览页）变化  
 item['indexLen'] = indexLen

#概览页获取链接的条数  
 return item  
  
 def parse\_item(self, response):

#用来处理每个请求下载的响应的方法，解析页面，将抓到的数据提取为字典，#达到监控细览页标题、引题、副题、正文、时间、来源、结构是否变化的目

#的，并返回结果  
 item = {}  
 item['title'] = response.xpath('//div[@class="news\_con"]/h1').extract\_first()

#监控标题变化  
 item['content'] = response.xpath('//div[@class="con"][/@class').extract\_first](mailto:/@class').extract_first)()

#监控正文变化  
 item['source'] = response.xpath('//div[@class="time"][/span/@class').extract\_first](mailto:/span/@class').extract_first)()

#监控来源变化  
 item['author'] = None

#监控作者变化  
 item['time'] = response.xpath('//div[@class="time"]/@class').extract\_first()

#监控时间变化  
 item['topTitle'] = ''

#监控引题变化  
 item['bottomTitle'] = ''

#监控副题变化  
 return item

### Xpath提取方法说明：XPath 使用路径表达式来选取网页的节点或节点集。节点是通过沿着路径 (path) 或者path+节点属性定位来选取的。使用xpath+节点属性时：//ul[@class="news\_list"]/li表示：所有符合class属性为news\_list的ul标签下的li标签是我们要提取的内容，css和正则提取方法与xpath提取类似。

### 脚本提取字段的xpath或者css的规则写的越详细越好（即定位标签的时候加上标签属性和父节点的标签），这样原网站一稍稍变动，检测就能报错，起到更好的检测作用。

### Python中注意代码缩进，否则程序会报错。