**监控工具开发文档**

1. **项目环境**

系统：Window

程序：python3.6

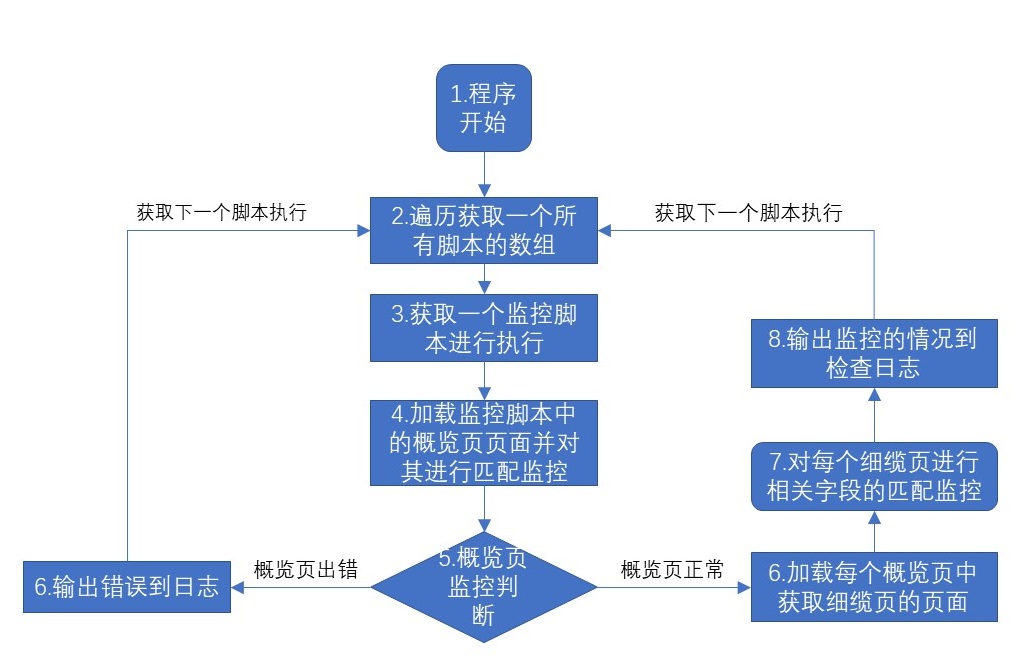
框架：scrapy1.5

1. **监控的原理**

监控主要的原理是通过脚本验证网页中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概览页和细缆页达到监控的目的。概览页解析其包含的细缆链接（看概览页是不是能正常访问，能不能取到细缆页的链接，只要不能就判断概览页错误），细缆页解析每篇新闻所需的特定字段（如果某个字段的标签不能取到就说明该细缆页的该字段的网页结构发生改变）。 监控的解析的方式有xpath路径，css样式，正则表达式。把所有采集点的监控情况输出到当天的汇总日志，再把汇总检查日志的错误采集点收集到当天的错误日志中。

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：**读取excel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，概览页无法访问，概览页结构改变）
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错误，细缆页未检查。获取下一个监控脚本执行。

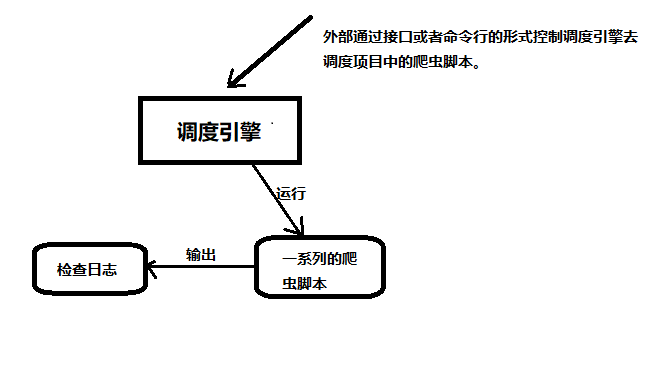
**正常：**进行后面的流程

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。
4. **筛选监控日志中的错误:** 如果该采集点的监控中发现是错误的，输出该采集点的错误情况到错误日志。

**10. 所有脚本运行完成：**所有的采集点监控脚本运行结束，查看NewsCrawl\result\里的 所有采集点监控情况。查看NewsCrawl\disable\_script\_list.txt 发现哪些脚本编写出错，进行修改。

注：针对多模板的网页，采用采集多个if判断分支【例：author、author1...在判断是利用if判断，只要一个if分支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. 调度引擎一个包含所有爬虫脚本接口，被外部调用，控制项目中爬虫脚本的运行。
2. 爬虫脚本：每个采集点进行具体检查的脚本。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：为所有采集点爬虫脚本输出的检查日志汇总。

**4．目录结构**

NewsCrawl为项目的根目录。

NewsCrawl\disable\_script\_list.txt: 为显示配置错误的脚本文件，通过这个文件可以查看那些监控脚本书写错误。

NewsCrawl\NewsCrawl\:为项目的代码所在目录，代码内容包括每个采集点的监控脚本，日志格式控制，日志显示字段的提取规则，项目的整体设置，每个脚本的个性化设置，监控过程中的一些中间件设置。

NewsCrawl\logs\：为程序的运行日志目录，显示程序运行过程的错误，和脚本书写相关的错误。

NewsCrawl\result\: 为程序对采集点监控的日志目录，此日志为汇总日志。

NewsCrawl\result\wrong\: 为程序对采集点检查的错误日志目录，此目录的日志中出现的采集点均为结构发生改动的采集点。

NewsCrawl\work\ ：放excel的目录，有两种excel。 gen\_spiders\_from\_excel用来生成脚本模板的excel，start\_check\_from\_excel 需要进行监控采集点汇总的excel

NewsCrawl\gen\_spider.py : 用NewsCrawl\work\ 里的excel自动生成模板（需要再配置即可运行）

NewsCrawl\scrapyd\_server.bat : 启动scrapyd服务，该服务确保了程序的调度运行。

NewsCrawl\scrapyd\_task.py : scrapyd任务相关类，实现了推任务脚本到scrapyd，进行任务调度，查看任务的执行状况。其它目录和文件：为项目自带，不可删除，与程序的逻辑无关。

NewsCrawl\start.py ： 执行excel清单中的待检查的采集点脚本，开始检查。

NewsCrawl\watch.py : 查看脚本的执行状况

**其它目录和文件**：为项目自带，不可删除，与程序的逻辑无关。

**5.日志部分**

**日志综述**：

日志文件的命名为当天日期的txt文件，每天都会以当天日期命名，日志的分隔符为^ ,每个运行监控的采集点的都在汇总日志文件中有自己的一行日志。汇总日志中的每行的最后一个错误综述字段为监控情况综述，该字段为正常说明该采集点为正常，否则为错误的采集点。汇总日志中的错误采集点会被收集到错误日志中。日志为txt格式，可以用excel导入阅读更加直观。日志命名规则如下图

**C:\Users\lim\AppData\Local\Temp\1534127487(1).png**

**日志目录** ：

* + - 1. NewsCrawl\result\(检查汇总)
      2. NewsCrawl\result\wrong\(错误的采集点)

**字段格式**：

*1.汇总日志的格式*：

该日志一共16个字段。第一行为字段名（表头），第二行开始是采集点的监控情况。错误总数字段为正常，这个采集点是正常的。下面是具体字段的说明：



a.脚本名（为采集点名字的拼音首字母），网站名，栏目名，栏目别名，网址 5个为该采集点脚本的基本信息。

b.开始时间，用时：2个为监控时该脚本的运行情况，什么时间开始的，用时多少秒。

c:概览页检查、标题、正文、来源、作者、时间、引题、副题 8个为具体字段的检查情况，每个字段有两种结果（正常，错误并包含错误的url）

d:错误综述：为前面的综述字段。若前面的（概览页检查、标题、正文、来源、作者、时间、引题、副题）都为正常，那么该字段为正常，只要前面的字段有一个不正常那么该字段为不正常。

*2错误日志格式*：

该日志一共12个字段。第一行为字段名（表头），第二行开始是采集点的监控情况。错误日志为出错的采集点，下面出现的采集点都是报错。下图中有一个错误的采集点，他的其它字段都正常，来源字段的网页结构发生变化。故报来源错误，并贴出了错误所在页面的地址。 下面是具体字段的说明情况。

C:\Users\lim\AppData\Local\Temp\1534734795(1).png

1. 网站名，栏目名，栏目别名，网址 4个为该采集点脚本的基本信息。
2. 概览页检查、标题、正文、来源、作者、时间、引题、副题 8个为具体字段的检查情况，每个字段有两种结果（正常，错误并包含错误的url）

**6.使用方法**

**监控脚本配置**：

1.把待生成脚本的excel清单放入NewsCrawl\work\gen\_spiders\_from\_excel\ 目中的

（注意格式，按照该目录中的excel例子一样）。

2.python 运行 gen\_spider.py即可批量生成脚本。生成的脚本是需要再配置的。

3.配置脚本中链接提取规则，字段的提取规则，完成配置的脚本再下次检查中就会运行

**运行检查**：

1.先把要检查采集点清单的excel放入 NewsCrawl\work\start\_check\_from\_excel\ 目录中（注意格式，按照该目录中的excel例子一样）

2.先点击运行scrapyd\_server.bat，启动scrapyd服务，不要关闭服务的窗口。

3.使用python 运行 start.py 按照NewsCrawl\work\start\_check\_from\_excel\ 目录

中excel的清单 调度爬虫程序的运行，运行清单中的所有脚本。

4.python 运行 watch.py 查看scrapyd中脚本的执行状况，全部脚本运行结束了再检查监控结果。

5.监控结束，查看NewsCrawl\result\ 里的日志查看运行脚本采集点的监控状态。

6.查看NewsCrawl\disable\_script\_list.txt 文件有哪些脚本没有运行，对照NewsCrawl\logs\里的程序运行日志把脚本修改正确。

**7.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。