**Python中日志类的简单封装-框架内部日志实现**

Python中有一个很强大的日志接口，类似log4j，支持很多场景的日志输出，输出到控制台，输出到文件，发送到服务器等。可以通过logger = logging.getLogger(logger) 来实例化一个logger对象，然后输出 各种级别的日志信息。

在自动化测试框架的设计中，为了更好的定位报错位置和跟踪脚本每一个步骤到底干了什么事情，我们会考虑到把相关操作以日志的形式记录下来，保存到本地文件上。这个时候我们就有必要自己封装一个log类方法，方便自己在框架中不同地方调用。

需求：

1. Log能在控制台输出，也能保存到项目相关文件路径
2. Log文件名称以执行脚本的系统时间命名例如201701101023.log 年月日时分.log
3. 日志文件里每行输出需要格式化
4. 我们主要在框架模块中去处理log，不建议在脚本中处理，也就是说，例如我们有一个浏览器引擎类（BrowserEngine.py）,我们可以在这个类中写上一些log输出或者异常捕获，同样我们在POM中的BasePage.py中封装的那些方法也添加输出，只要脚本调用，都会产生相应的日志，保存在日志文件。至于是INFO还是ERROR日志，脚本开发人员自己提前设计好，正常运行，通过INFO输出消息，异常抱错用ERROR输出.
5. 在framework包下新建一个Logger.py类，代码如下：

*# \_\*\_ coding: utf-8 \_\*\_***import** logging  
**import** os.path  
**import** time  
  
  
**class** Logger(object):  
  
 **def** \_\_init\_\_(self, logger):  
 *'''  
 指定保存日志的文件路径，日志级别，以及调用文件  
 将日志存入到指定的文件中  
 '''  
  
 # 创建一个logger* self.logger = logging.getLogger(logger)  
 self.logger.setLevel(logging.DEBUG)  
  
 *# 创建一个handler，用于写入日志文件* rq = time.strftime(**'%Y%m%d%H%M'**, time.localtime(time.time()))  
 log\_path = os.path.dirname(os.getcwd()) + **'/Logs/'** log\_name = log\_path + rq + **'.log'** fh = logging.FileHandler(log\_name)  
 fh.setLevel(logging.INFO)  
  
 *# 再创建一个handler，用于输出到控制台* ch = logging.StreamHandler()  
 ch.setLevel(logging.INFO)  
  
 *# 定义handler的输出格式* formatter = logging.Formatter(**'%(asctime)s - %(name)s - %(levelname)s - %(message)s'**)  
 fh.setFormatter(formatter)  
 ch.setFormatter(formatter)  
  
 *# 给logger添加handler* self.logger.addHandler(fh)  
 self.logger.addHandler(ch)  
  
 **def** getlog(self):  
 **return** self.logger

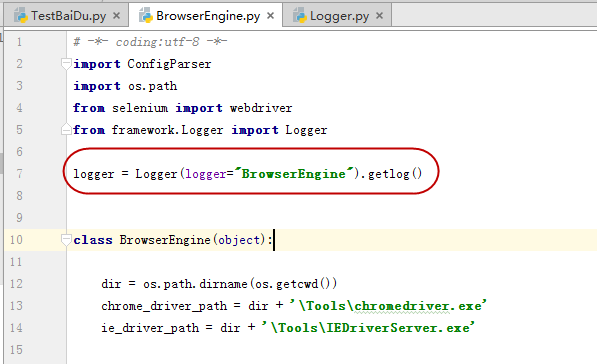
解释下，在这个代码里，我写死了log的保存路径 ：python项目根目录/Logs/xxxx.log

rq = time.strftime(**'%Y%m%d%H%M'**, time.localtime(time.time()))  
可以网上搜索time.strftime()，查看**%Y%m%d%H%M含义**

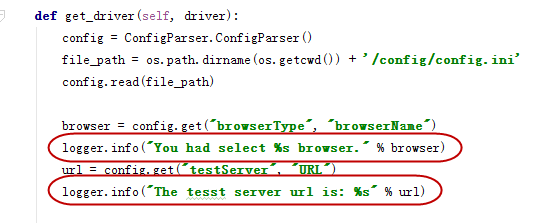
**这里表示：201701010923 --- 年月日时分**

1. 在BrowserEngine.py类去调用Logger.py方法

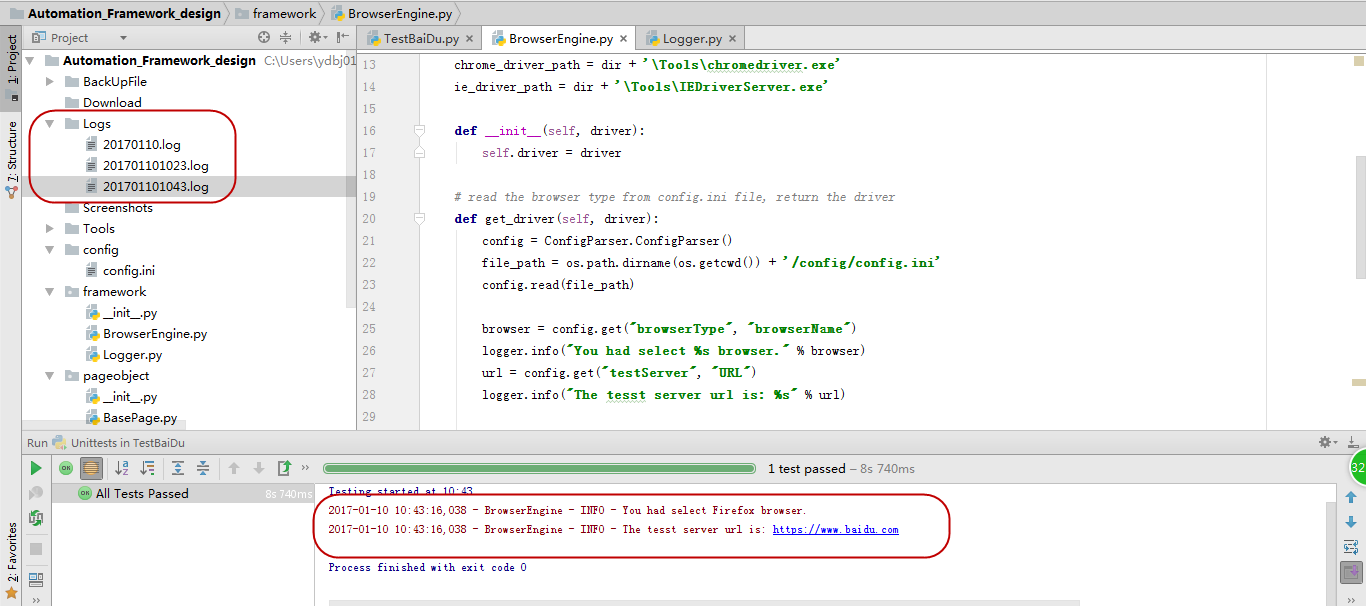
2.1在BrowserEngine模块中添加一个logger的实例，如图：



* 1. 在具体方法中写logger方法，输出message



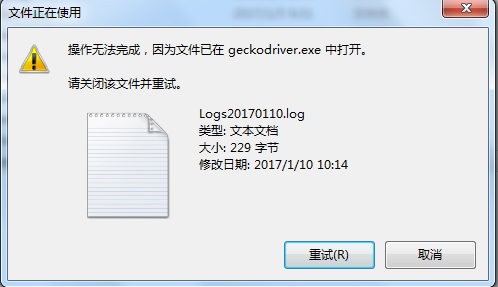
* 1. 不修改脚本，我们测试下TestBaiDu.py 看看输出：



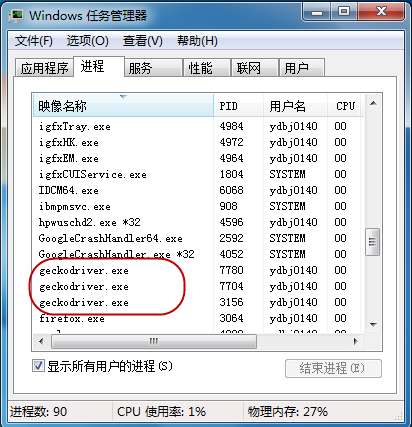
可以看到，控制台和本地存储log的路径都有数据。

小贴士：

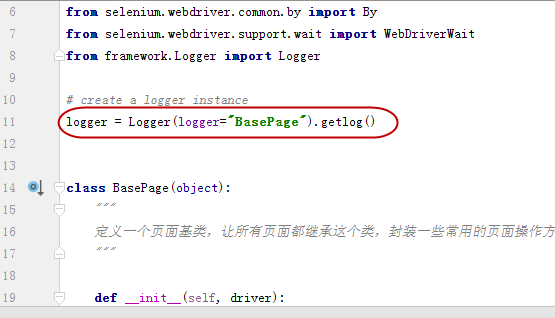
如果运行多次后，会生成很多个xxxx.log，有时候我们为了美观，会及时去/Logs路径下删除一些过期的log文件，当我删除的，发现报错，记录在这里：



Geckodriver.exe这个我们之前放到了python的安装目录下，这个是firefox的原生支持的驱动文件，根据这个提示，我们需要在系统资源管理器结束关于geckodriver.exe的进程，才能删除这些文件。



1. 在BasePage.py 调用Logger()的方法
2. 初始化一个logger实例



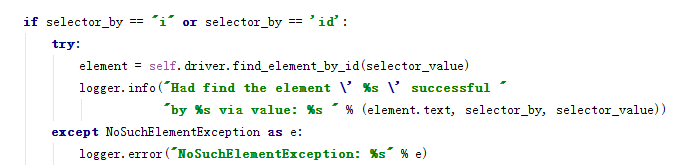
1. 介绍在find\_element（）方法里添加log输出

虽然我们在find\_element()方法里写了8只find\_element的方法去查找元素。实际上UI自动化，大部分元素都采用xpath定位，id,name都比较难定位到一些复杂元素，还一个原因是，firefox上有firepath插件可以获取元素xpath。

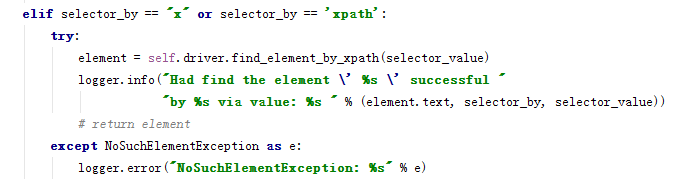
这里我们就在by.id 和by.xpath 两个地方演示如何调用logger，只要每次查找元素都会输出一个INFO 或者EORR的消息到日志文件。

主要添加两个message，INFO，成功找到

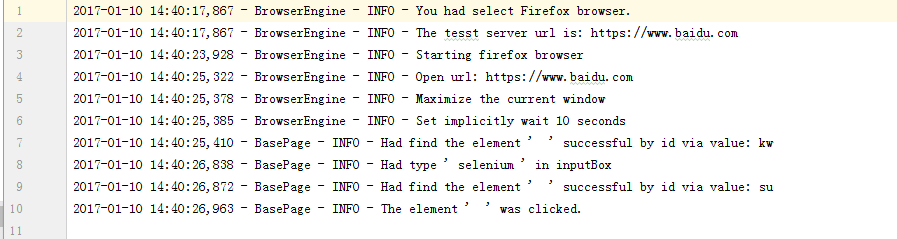
ERROR. 没有找到元素，打印出异常



By.Xpath



1. 测试下by.ID（百度输入，点击搜索按钮）
   1. 没有错误



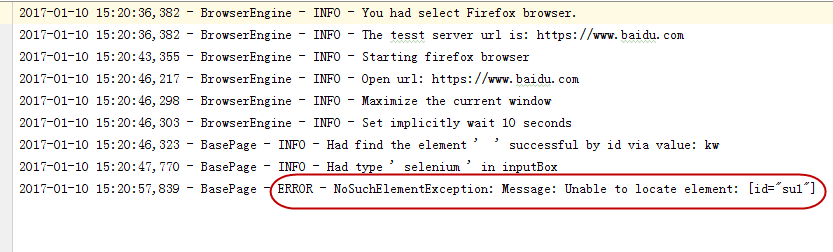
解释下，图中的 ‘’中是空白，因为当前搜索输入框，element.text本身是空白所以显示空白，同样搜索按钮.elelment.text也是空，百度一下，这个字段是存储在value=’百度一下’

这个暂时没有更好方法，我看了下大部分元素element.text是有文字的，所以这个地方我就选择了element.text

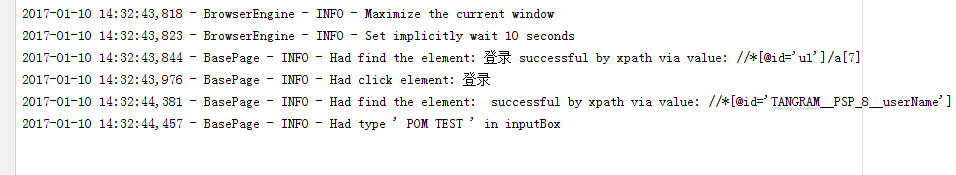
* 1. 有error

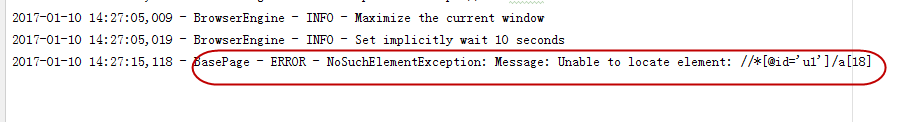
故意写错一个by.id的元素定位，来看看error记录





1. 测试下by.xpath





1. 为其他公用方法添加logger

