Pyppeteer简介

异步的selenium。在 Pyppetter的背后是有一个类似 Chrome 浏览器的 Chromium 浏览器在执行一些动作进行网页渲染，首先说下 Chrome 浏览器和 Chromium 浏览器的渊源。

> Chromium 是谷歌为了研发 Chrome 而启动的项目，是完全开源的。二者基于相同的源代码构建，Chrome 所有的新功能都会先在 Chromium 上实现，待验证稳定后才会移植，因此 Chromium 的版本更新频率更高，也会包含很多新的功能，但作为一款独立的浏览器，Chromium 的用户群体要小众得多。两款浏览器“同根同源”，它们有着同样的 Logo，但配色不同，Chrome 由蓝红绿黄四种颜色组成，而 Chromium 由不同深度的蓝色构成。

Pyppeteer 就是依赖于 Chromium 这个浏览器来运行的。那么有了 Pyppeteer 之后，我们就可以免去那些繁琐的环境配置等问题。如果第一次运行的时候，Chromium 浏览器没有安装，那么程序会帮我们自动安装和配置，就免去了繁琐的环境配置等工作。另外 Pyppeteer 是基于 Python 的新特性 async 实现的，所以它的一些执行也支持异步操作，效率相对于 Selenium 来说也提高了。

环境安装

- 由于 Pyppeteer 采用了 Python 的 async 机制，所以其运行要求的 Python 版本为 3.5 及以上
- pip install pyppeteer

快速上手

- 爬取http://quotes.toscrape.com/js/ 全部页面数据

```
import asyncio
from pyppeteer import launch
from lxml import etree


async def main():
    browser = await launch(headless=False)
    page = await browser.newPage()
    await
page.goto('http://quotes.toscrape.com/js/')
    page_text = await page.content()
    tree = etree.HTML(page_text)
    div_list =
tree.xpath('//div[@class="quote"]')
    print(len(div_list))
    await browser.close()

c = main()
```

```
asyncio.get_event_loop().run_until_complete(c
)
```

- 代码解释：
  - 解释：launch 方法会新建一个 Browser 对象，然后赋值给 browser，然后调用 newPage 方法相当于浏览器中新建了一个选项卡，同时新建了一个 Page 对象。然后 Page 对象调用了 goto 方法就相当于在浏览器中输入了这个 URL，浏览器跳转到了对应的页面进行加载，加载完成之后再调用 content 方法，返回当前浏览器页面的源代码。在这个过程中，我们没有配置 Chrome 浏览器，没有配置浏览器驱动，免去了一些繁琐的步骤，同样达到了 Selenium 的效果，还实现了异步抓取，爽歪歪！

详细用法

- 开启浏览器
  - 调用 launch() 方法即可，相关参数介绍：
    - ignoreHTTPSErrors (bool): 是否要忽略 HTTPS 的错误，默认是 False。
    - headless (bool): 是否启用 Headless 模式，即无界面模式，默认是开启无界面模式的。如果设置为 False则是有界面模式。
    - executablePath (str): 可执行文件的路径，如果指定

之后就不需要使用默认的 Chromium 了，可以指定为已有的 Chrome 或 Chromium。
    - devtools (bool): 是否为每一个页面自动开启调试工具(浏览器开发者工具)，默认是 False。如果这个参数设置为 True，那么 headless 默认参数就会无效，会被强制设置为 False。
    - args (List[str]): 在执行过程中可以传入的额外参数。
  - 关闭提示条：”Chrome 正受到自动测试软件的控制”，这个提示条有点烦，那咋关闭呢？这时候就需要用到 args 参数了，禁用操作如下：

```
browser = await launch(headless=False,
args=['--disable-infobars'])
```

- 处理页面显示问题：访问淘宝首页

```
import asyncio
from pyppeteer import launch

async def main():
    browser = await launch(headless=False)
    page = await browser.newPage()
    await
page.goto('https://www.taobao.com')
    await asyncio.sleep(3)
    await browser.close()
asyncio.get_event_loop().run_until_complete
(main())
```

- 发现页面显示出现了问题，需要手动调用setViewport方法设置显示页面的长宽像素。设置如下：

```
import asyncio
from pyppeteer import launch

width, height = 1366, 768
async def main():
    browser = await
launch(headless=False)
    page = await browser.newPage()

    await page.setViewport({'width':
width, 'height': height})
```

```
    await
page.goto('https://www.taobao.com')
    await asyncio.sleep(3)
    await browser.close()
asyncio.get_event_loop().run_until_comple
te(main())
```

- 执行js程序：拖动滚轮。调用evaluate方法。

```
import asyncio
from pyppeteer import launch

width, height = 1366, 768
async def main():
    browser = await launch(headless=False)
    page = await browser.newPage()
    await page.setViewport({'width': width,
'height': height})
    await
page.goto('https://www.taobao.com')
    await asyncio.sleep(2)
    #执行js代码
    await
page.evaluate('window.scrollTo(0,document.b
ody.scrollHeight)')
    await asyncio.sleep(2)
```

```
    await browser.close()
asyncio.get_event_loop().run_until_complete
(main())
```

- 规避检测

```
import asyncio
from pyppeteer import launch

width, height = 1366, 768

async def main():
    #规避检测
    browser = await launch(headless=False,
args=['--disable-infobars'])
    page = await browser.newPage()
    await page.setViewport({'width': width,
'height': height})
    await
page.goto('https://login.taobao.com/member/
login.jhtml?
redirectURL=https://www.taobao.com/')
    #规避检测
    await page.evaluate(
        '''() =>{
Object.defineProperties(navigator,{
webdriver:{ get: () => false } }) }''')
```

```
    await asyncio.sleep(3)
    await browser.close()

asyncio.get_event_loop().run_until_complete
(main())
```

- 节点交互

```
import asyncio
from pyppeteer import launch


async def main():
    # headless参数设为False，则变成有头模式
    browser = await launch(
        headless=False
    )
    page = await browser.newPage()
    # 设置页面视图大小
    await page.setViewport(viewport=
{'width': 1280, 'height': 800})

    await
page.goto('https://www.baidu.com/')
    # 节点交互
    await page.type('#kw', '周杰伦',
{'delay': 1000})
```

```
    await asyncio.sleep(3)
    #点击搜索按钮
    await page.click('#su')
    await asyncio.sleep(3)
    # 使用选择器选中标签进行点击
    alist = await
page.querySelectorAll('.s_tab_inner > a')
    a = alist[3]
    await a.click()
    await asyncio.sleep(3)
    await browser.close()
asyncio.get_event_loop().run_until_complete
(main())
```

爬虫练习

- 异步爬取头条和网易新闻首页的新闻标题
  - https://www.toutiao.com/
  - https://news.163.com/domestic/

```
import asyncio
from pyppeteer import launch
from lxml import etree
```

```
async def main():
    # headless参数设为False，则变成有头模式
    browser = await launch(
        headless=False
    )
    page1 = await browser.newPage()

    # 设置页面视图大小
    await page1.setViewport(viewport={'width':
1280, 'height': 800})

    await page1.goto('https://www.toutiao.com/')
    await asyncio.sleep(2)
    # 打印页面文本
    page_text = await page1.content()

    page2 = await browser.newPage()
    await page2.setViewport(viewport={'width':
1280, 'height': 800})
    await
page2.goto('https://news.163.com/domestic/')
    await
page2.evaluate('window.scrollTo(0,document.body.
scrollHeight)')
    page_text1 = await page2.content()

    await browser.close()
```

```
    return {'wangyi': page_text1, 'toutiao':
page_text}


def parse(task):
    content_dic = task.result()
    wangyi = content_dic['wangyi']
    toutiao = content_dic['toutiao']
    tree = etree.HTML(toutiao)
    a_list = tree.xpath('//div[@class="title-
box"]/a')
    for a in a_list:
        title = a.xpath('./text()')[0]
        print('toutiao:', title)
    tree = etree.HTML(wangyi)
    div_list =
tree.xpath('//div[@class="data_row news_article
clearfix "]')
    for div in div_list:
        title =
div.xpath('.//div[@class="news_title"]/h3/a/text
()')[0]
        print('wangyi:', title)

tasks = []
task1 = asyncio.ensure_future(main())
```

```
task1.add_done_callback(parse)
tasks.append(task1)
asyncio.get_event_loop().run_until_complete(asyncio.wait(tasks))
```

滑动验证

```
import random
from pyppeteer import launch
import asyncio
import cv2
from urllib import request


async def get_track():
    background = cv2.imread("background.png", 0)
    gap = cv2.imread("gap.png", 0)

    res = cv2.matchTemplate(background, gap, cv2.TM_CCOEFF_NORMED)
    value = cv2.minMaxLoc(res)[2][0]
    print(value)
    return value * 278 / 360
```

```
async def main():
    browser = await launch({
        # headless指定浏览器是否以无头模式运行，默认是True。
        "headless": False,
        #设置窗口大小
        "args": ['--window-size=1366,768'],
    })
    # 打开新的标签页
    page = await browser.newPage()
    # 设置页面大小一致
    await page.setViewport({"width": 1366, "height": 768})
    # 访问主页
    await page.goto("https://passport.jd.com/new/login.aspx?")

    # 单击事件
    await page.click('div.login-tab-r')
    # 模拟输入用户名和密码,输入每个字符的间隔时间delay ms
    await page.type("#loginname", '324534534@qq.com', {
        "delay": random.randint(30, 60)
    })
    await page.type("#nloginpwd", '345653332', {
```

```
            "delay": random.randint(30, 60)
        })

        # page.waitFor 通用等待方式，如果是数字，则表示等待具体时间（毫秒）：等待2秒
        await page.waitFor(2000)
        await page.click("div.login-btn")
        await page.waitFor(2000)
        # page.jeval（selector, pageFunction）#定位元素，并调用js函数去执行
        #=>表示js的箭头函数：el = function(el){return el.src}
        img_src = await page.Jeval(".JDJRV-bigimg > img", "el=>el.src")
        temp_src = await page.Jeval(".JDJRV-smallimg > img", "el=>el.src")

        request.urlretrieve(img_src, "background.png")
        request.urlretrieve(temp_src, "gap.png")

        # 获取gap的距离
        distance = await get_track()
        """
            # Pyppeteer 三种解析方式
            Page.querySelector()  # 选择器
            Page.querySelectorAll()
```

```
        Page.xpath()  # xpath  表达式
        # 简写方式为:
        Page.J(), Page.JJ(), and Page.Jx()
        """
    #定位到滑动按钮标签
    el = await page.J("div.JDJRV-slide-btn")
    # 获取元素的边界框，包含x,y坐标
    box = await el.boundingBox()
    #box={'x': 86, 'y': 34, 'width': 55.0,
'height': 55.0}
    #将鼠标悬停/一定到指定标签位置
    await page.hover("div.JDJRV-slide-btn")
    #按下鼠标
    await page.mouse.down()
    #模拟人的行为进行滑动
    # steps 是指分成几步来完成，steps越大，滑动速度越
慢
    #move(x,y)表示将鼠标移动到xy坐标位置
    await page.mouse.move(box["x"] + distance +
random.uniform(10, 30),
                          box["y"],
                          {"steps": 100})
    await page.waitFor(1000)
    await page.mouse.move(box["x"] + distance +
random.uniform(10, 30),
                          box["y"], {"steps":
100})
```

```
    await page.mouse.up()
    await page.waitFor(2000)


loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```