



```
option[value='teyaoxgrs']").click()
    driver.find_element_by_css_selector("select#tabJcwyxt_jieci >
option[value='d11jie']").click()
    search2 = driver.find_element_by_class_name('input_a2')
    search2.click()
    time.sleep(5)

    #get the response
    body = driver.page_source
    return HtmlResponse(driver.current_url, body=body, encoding='utf-8',
request=request)
```

```
class ProfileSpider(CrawlSpider):
    name = 'pccprofile2'
    rules = [Rule(SgmlLinkExtractor(allow=(),restrict_xpaths=
('//div[@class='table']")), callback='parse_item')]

    def parse_item(self, response):
        hxs = HtmlXPathSelector(response)
        items = []
        item = Ppcprofile2Item()
        item ["name"] = hxs.select("//h1/text()").extract()
        item ["title"] =
hxs.select("//div[@id='contentbody']//tr//td//text()").extract()
        items.append(item)

    #click next page
    while True:
        next = response.findElement(By.linkText("下一页"))
        try:
            next.click()
        except:
            break

    return(items)
```

python

selenium

scrapy

Share

Improve this question

Follow

edited Jul 2, 2015 at 17:30

asked Jul 2, 2015 at 1:33

 Onyi Lam

147

1

2

10


[This question](#) and [this one](#) helped me a lot – [Henadzi Rabkin](#) Dec 22, 2019 at 22:30

## 2 Answers

Sorted by:

Highest score (default)


⬆




Use [Downloader Middleware](#) to catch selenium-required pages *before* you process them regularly with Scrapy:


26


The downloader middleware is a framework of hooks into Scrapy's request/response processing. It's a light, low-level system for globally altering Scrapy's requests and responses.





Here's a very basic example using PhantomJS:






```
from scrapy.http import HtmlResponse
from selenium import webdriver

class JSMiddleware(object):
    def process_request(self, request, spider):
        driver = webdriver.PhantomJS()
        driver.get(request.url)

        body = driver.page_source
        return HtmlResponse(driver.current_url, body=body, encoding='utf-8',
request=request)
```

Once you return that `HtmlResponse` (or a `TextResponse` if that's what you really want), Scrapy will cease processing downloaders and drop into the spider's `parse` method:

If it returns a `Response` object, Scrapy won't bother calling any other `process_request()` or `process_exception()` methods, or the appropriate download function; it'll return that response. The `process_response()` methods of installed middleware is always called on every response.



In this case, you can continue to use your spider's `parse` method as you normally would with HTML, except that the JS on the page has already been executed.

**Tip: Since the Downloader Middleware's `process_request` method accepts the spider as an argument, you can add a conditional in the spider to check whether you need to process JS at all, and that will let you handle both JS and non-JS**

pages with the exact same spider class.

Share Improve this answer Follow

edited Jul 2, 2015 at 17:22

answered Jul 2, 2015 at 14:02

 **JoeLinux**  
4,307 1 31 31

- Hi Joe, thanks for the suggestion. I did what you suggested by including a JSMiddleware class before the CrawlSpider class. And within the middleware class, I pasted the code of which I select from the dropdown and click. There was no result and no error returned either. Please see the edits above. – Onyi Lam Jul 2, 2015 at 17:07
- Your edit doesn't include a return statement. Make sure you're returning a response from the middleware. – JoeLinux Jul 2, 2015 at 17:10
- I had the return statement. i realized just now I got an error from scrapy.http import HttpResponse ImportError: cannot import name HttpResponse i will see if this is the issue... – Onyi Lam Jul 2, 2015 at 17:21
- AH, that's my fault! It should be HtmlResponse , not HttpResponse . I'll make that edit. – JoeLinux Jul 2, 2015 at 17:22
- still no luck. no error and result. This is the console output: ` 2015-07-02 10:33:28-0700 [scrapy] INFO: Enabled downloader middlewares: HttpAuthMiddleware, DownloadTimeoutMiddleware, UserAgentMiddleware, RetryMiddleware, DefaultHeadersMiddleware, MetaRefreshMiddleware, HttpCompressionMiddleware, RedirectMiddleware, CookiesMiddleware, ChunkedTransferMiddleware, DownloaderStats 2015-07-02 10:33:28-0700 [scrapy] INFO: Enabled spider middlewares: HttpErrorMiddleware, OffsiteMiddleware, RefererMiddleware, UrlLengthMiddleware, DepthMiddleware` – Onyi Lam Jul 2, 2015 at 17:38

▲

9

❄

▼

🔖

🕒

Here is a middleware for Scrapy and Selenium

```
from scrapy.http import HtmlResponse
from scrapy.utils.python import to_bytes
from selenium import webdriver
from scrapy import signals

class SeleniumMiddleware(object):

    @classmethod
    def from_crawler(cls, crawler):
        middleware = cls()
        crawler.signals.connect(middleware.spider_opened,
signals.spider_opened)
        crawler.signals.connect(middleware.spider_closed,
signals.spider_closed)
        return middleware

    def process_request(self, request, spider):
        request.meta['driver'] = self.driver # to access driver from response
        self.driver.get(request.url)
        body = to_bytes(self.driver.page_source) # body must be of type bytes
        return HtmlResponse(self.driver.current_url, body=body, encoding='utf-8', request=request)

    def spider_opened(self, spider):
        self.driver = webdriver.Firefox()

    def spider_closed(self, spider):
        self.driver.close()
```

Also need to add in settings.py

```
DOWNLOADER_MIDDLEWARES = {
    'yourproject.middlewares.selenium.SeleniumMiddleware': 200
}
```

Decide weather its 200 or something else based on docs.

Update firefox headless mode with scrapy and selenium

If you want to run firefox in headless mode then install xvfb

```
sudo apt-get install -y xvfb
```

and PyVirtualDisplay

```
sudo pip install pyvirtualdisplay
```

and use this middleware

```
from shutil import which

from pyvirtualdisplay import Display
from scrapy import signals
from scrapy.http import HtmlResponse
from scrapy.utils.project import get_project_settings
from selenium import webdriver
```

```
from selenium.webdriver.firefox.firefox_binary import FirefoxBinary

settings = get_project_settings()

HEADLESS = True

class SeleniumMiddleware(object):

    @classmethod
    def from_crawler(cls, crawler):
        middleware = cls()
        crawler.signals.connect(middleware.spider_opened,
signals.spider_opened)
        crawler.signals.connect(middleware.spider_closed,
signals.spider_closed)
        return middleware

    def process_request(self, request, spider):
        self.driver.get(request.url)
        request.meta['driver'] = self.driver
        body = str.encode(self.driver.page_source)
        return HtmlResponse(self.driver.current_url, body=body, encoding='utf-
8', request=request)

    def spider_opened(self, spider):
        if HEADLESS:
            self.display = Display(visible=0, size=(1280, 1024))
            self.display.start()
            binary = FirefoxBinary(settings.get('FIREFOX_EXE') or which('firefox'))
            self.driver = webdriver.Firefox(firefox_binary=binary)

    def spider_closed(self, spider):
        self.driver.close()
        if HEADLESS:
            self.display.stop()
```

where settings.py contains

```
FIREFOX_EXE = '/path/to/firefox.exe'
```

The problem is that some versions of firefox don't work with selenium. To solve this problem you can download firefox version 47.0.1 (this version works flawlessly) from [here](#) then extract it and put it inside your selenium project. Afterwards modify firefox path as

```
FIREFOX_EXE = '/path/to/your/scrapyproject/firefox/firefox.exe'
```

Share Improve this answer Follow

edited Oct 29, 2016 at 12:55

answered Aug 18, 2016 at 6:03

 [Levon](#)

12k 4 49 44

what does the from\_crawler function do? what does the cls parameter represent and what is it that the statements in the function do? – oldboy Jul 14, 2018 at 17:57

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

python selenium scrapy

See similar questions with these tags.