Ask Company

크롤링 차근차근 시작하기 (2/E) - 중급편

Selenium: Element 찾아서 읽어내기

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

WebElement

HTML Element를 표현하는 개체
selenium.webdriver.remote.webelement.WebElement

Element(s) 찾기

단순히 HTML 내 컨텐츠를 읽는 목적으로는 driver.page_source를 통해 전체 HTML 내역을 읽어내고, HTML Parser(BeautifulSoup4 등)를 통해 처리하시는 것이 편리하실 수 있습니다.

driver/element

.find_element_by

_css_selector

_id, _tag_name, _class_name, _name → css selector로 변환하여 전달

_xpath

_link_text : 링크 레이블명으로 찾기

_partial_link_text

1개 Element 찾기 (매칭되는 WebElement가 다수 있을 경우, 처음 1개 반환)
- 반환값 : WebElement로 반환
- 없을 경우 : selenium.common.exceptions.NoSuchElementException 예외 발생

.find_elements_by

_css_selector

~~_id~~, _tag_name, _class_name, _name → css selector로 변환하여 전달

_xpath

_link_text

_partial_link_text

다수 Element 찾기
- 반환값 : WebElement 리스트로 반환
- 없을 경우 : 빈 리스트를 반환

WebElement의 주요 속성 (1)

.clear() : text 입력 요소일 경우에, 입력된 값을 삭제

.send_keys(*value) : 타이핑 시뮬레이션

file 업로드 필드의 경우, 이를 통해 파일경로 입력 가능 (상대경로/절대경로)

모든 Element에서 가능하고, keyboard shortcut 입력으로도 가능

특수키 : Keys.ENTER, Keys.ARROW_DOWN 등

```
from selenium.webdriver.common.keys import Keys
```

.click() : 클릭

.submit() : Form Field일 경우, submit

.is_display() : visible 여부

.is_enabled() : enable 여부

.is_selected() : 선택 여부 (checkbox/radio)

WebElement의 주요 속성 (2)

.text

display(즉 visible) 상태에서만 text 획득 else 빈 문자열

.get_attribute(*name*)

property를 먼저 시도 → attribute를 시도 → 없을 경우 None을 반환

attribute : HTML를 통해 지정된 initial state

property : 현재 state

.screenshot(*filename*)

driver에서는 .save_screenshot(*filename*)

.screenshot_as_base64

.screenshot_as_png

예시

```
<input type="text" name="passwd" id="passwd-id" />
```

```
# 조건에 따라, element는 찾는 다양한 방법
element = driver.find_element_by_id("passwd-id")
element = driver.find_element_by_css_selector("#passwd-id")
element = driver.find_element_by_css_selector("input#passwd-id")

element = driver.find_element_by_name("passwd")
element = driver.find_element_by_css_selector("[name=passwd]")
element = driver.find_element_by_css_selector("input[name=passwd]")

element = driver.find_element_by_xpath("//input[@id='passwd-id']")

# 지정 Element에 키 입력 넣기
element.send_keys("My Password")
```

예시) 네이버 검색

첫번째 추천 검색어로 검색하기

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

with webdriver.Chrome(...) as driver:
    driver.implicitly_wait(10)

    driver.get('https://www.naver.com')
    element = driver.find_element_by_id('query')
    element.send_keys('파이썬')
    time.sleep(2)
    element.send_keys(Keys.ARROW_DOWN, Keys.ENTER)

    element_list = driver.find_elements_by_class_name('sh_blog_title')
    for element in element_list:
        post_url = element.get_attribute('href')
        post_title = element.get_attribute('title')
        print(post_title)
        print(post_url)
```

주의) invisible WebElement의 .text 속성

WebElement의 .text 속성

visible할 때에만 획득 가능 (== .is_displayed())

대신 .get_attribute('innerHTML') 사용 가능

ex) 네이버 실시간 검색어

예시) 네이버 실시간 검색어

```
from bs4 import BeautifulSoup
from selenium import webdriver

with webdriver.Chrome(...) as driver:
    driver.get("https://www.naver.com")

    css_selector = '.PM_CL_realtimeKeyword_rolling_base .ah_k'
    element_list = driver.find_elements_by_css_selector(css_selector)

    keyword_list = [element.text for element in element_list]
    print(keyword_list) ['국립중앙의료원', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '']

    keyword_list = [element.get_attribute('innerHTML') for element in element_list]
    print(keyword_list) ['국립중앙의료원', '타워', '곡성', '곡성 해석', '나탈리포트만', '레옹', '위메프 33데이', '양잿물', '영화 브이아이피', '최현호', '원피스 875화 애니',
                         '트랩 7화', '빙의', '버드맨', '영화 곡성', '강다니엘', '브이아이피', '김연학', 'Im엔터테인먼트', '하나뿐인 내편 결말']
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    tag_list = soup.select(css_selector)
    keyword_list = [tag.text for tag in tag_list]
    print(keyword_list) ['국립중앙의료원', '타워', '곡성', '곡성 해석', '나탈리포트만', '레옹', '위메프 33데이', '양잿물', '영화 브이아이피', '최현호', '원피스 875화 애니',
                         '트랩 7화', '빙의', '버드맨', '영화 곡성', '강다니엘', '브이아이피', '김연학', 'Im엔터테인먼트', '하나뿐인 내편 결말', '국립중앙의료원']
```

Ajax 컨텐츠 크롤링하기

무작정 기다리지 말아요.

Waits

각 Elements들은 서로 다른 타이밍에 로딩됩니다.
→ GUI 애플리케이션은 비동기 세상

동작 원리

1. 지정 시간(timeout) 동안에
2. 지정 주기(interval, 디폴트 : 0.5초)로
3. 지정 조건의 Element(s) 존재 여부를 체크 (Polling)
    1. 혹은 커스텀 조건의 함수 지정 가능

2가지 waits

implicitly wait : driver에 전역 timeout 설정

explicitly wait : 매번 timeout 지정

Implicitly Wait 기본 코드

```
from selenium import webdriver

with webdriver.Chrome(...) as driver:
    driver.implicitly_wait(10)  # 현 driver에 전역 설정

    # ...

    element_list = driver.find_elements_by_css_selector('.course a')

    # ...
```

Explicitly Wait 기본 코드

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

with webdriver.Chrome(...) as driver:
    # ...

    locator = (By.CSS_SELECTOR, '.course a')
    element_list = WebDriverWait(driver, 10).until(
        EC.presence_of_all_elements_located(locator)
    )

    # ...
```

주의: implicit/explicit waits를 같이 쓰지 않기

공식문서에서 같지 쓰지 않기를 권고

WHY?

- 예기치못한 wait 시간이 소요될 수 있음.
  - implicit wait (10초), explicit wait (15초) 지정의 경우,
  - 실제로는 20초 이후에 timeout이 발생할 수 있다고 합니다.

https://www.seleniumhq.org/docs/04_webdriver_advanced.jsp#explicit-and-implicit-waits

단순 처리

```
from selenium import webdriver

with webdriver.Chrome(...) as driver:
    url = "https://askdjango.github.io/lv2/"

    driver.get(url)

    # 페이지 로딩이 끝나자마자, Elements 찾기
    element_list = driver.find_elements_by_css_selector('.course a')
    course_name_list = [e.text for e in element_list]
    print(course_name_list)
```

빈 리스트가 반환됩니다.

implicitly wait 적용하여,

Ajax 처리 완료시까지 timeout=10초 적용

```
from selenium import webdriver

with webdriver.Chrome(...) as driver:
    url = "https://askdjango.github.io/lv2/"

    driver.implicitly_wait(10)

    driver.get(url)

    element_list = driver.find_elements_by_css_selector('.course a')

    course_name_list = [e.text for e in element_list]
    print(course_name_list)
```

이후 모든 find_element(s) 함수에 대해, timeout 10초 적용

빈 리스트가 반환됩니다.

explicitly wait 적용하여,

Ajax 처리 완료시까지 timeout=10초 적용

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait              # 2.4.0 버전부터 지원
from selenium.webdriver.support import expected_conditions as EC     # 2.26.0 버전부터 지원

with webdriver.Chrome(...) as driver:
    url = "https://askdjango.github.io/lv2/"

    driver.get(url)

    locator = (By.CSS_SELECTOR, '.course a')
    element_list = WebDriverWait(driver, 10).until(
        EC.presence_of_all_elements_located(locator)
    )

    course_name_list = [e.text for e in element_list]
    print(course_name_list)
```

until_not도 지원

인자로 driver를 받는
호출가능한 객체 (Callable Object)

다양한 Wait Conditions

selenium.webdriver.support.expected_conditions

title_is
title_contains
presence_of_element_located
presence_of_all_elements_located
visibility_of_element_located
visibility_of
text_to_be_present_in_element
text_to_be_present_in_element_value
frame_to_be_available_and_switch_to_it
invisibility_of_element_located
element_to_be_clickable
staleness_of
element_to_be_selected
element_located_to_be_selected
element_selection_state_to_be
element_located_selection_state_to_be
alert_is_present

```python
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# 중략

locator = (By.CSS_SELECTOR, '.course a')
element_list = WebDriverWait(driver, 10).until(
    EC.presence_of_all_elements_located(locator)
)

course_name_list = [e.text for e in element_list]
```

https://github.com/SeleniumHQ/selenium/blob/master/py/selenium/webdriver/support/expected_conditions.py

커스텀 Wait Conditions 예시 (1)

```
def cond(driver):
    return driver.find_element_by_id('some_id')

element = WebDriverWait(driver, 10).until(cond)
print(element)
```

커스텀 Wait Conditions 예시 (2)

Lambda를 통한 함수 구현

```
cond = lambda driver: driver.find_element_by_id('some_id')
element = WebDriverWait(driver, 10).until(cond)
print(element)
```

커스텀 Wait Conditions 예시 (3)

함수 인스턴스에서의 호출가능한 객체

```
class Cond:
    def __init__(self):
        pass

    def __call__(self, driver):
        return driver.find_element_by_id('some_id')

cond = Cond()
element = WebDriverWait(driver, 10).until(cond)
print(element)
```

커스텀 Wait Conditions 예시 (4)

```
def element_has_css_class(locator, clss_class):
    def check_cond(driver):
        element = driver.find_element(*locator)
        if clss_class in element.get_attribute('class'):
            return element
        else:
            return False
    return check_cond

cond = element_has_css_class((By.CSS_SELECTOR, "div.intro_main > h3"), "myCSSClass")
element = WebDriverWait(driver, 10).until(cond, "Not Found myCSSClass in intro_main.")
print(element)
```

Demo: 네이버 뉴스 댓글 읽기

```
from selenium import webdriver

driver_path = 'drivers/chromedriver' # FIXME: 각자 환경에 맞게 수정
page_url = 'https://news.naver.com/main/read.nhn?mode=LSD&mid=shm&sid1=101&oid=022&aid=0003353811'

with webdriver.Chrome(executable_path=driver_path) as driver:
    driver.get(page_url)

    tag_list = driver.find_elements_by_css_selector(".u_cbox_list .u_cbox_contents")
    for tag in tag_list:
        print(tag.text)
```

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

- Ask Company