Selenium

index

미리 준비해야 할 사항

- selenium, pillow 설치

Selenium 설치

Mac

1. install chrome driver

(1) Download chrome driver

아래의 링크에서 크롬 드라이버를 다운로드 합니다.

- https://sites.google.com/a/chromium.org/chromedriver/home
- https://chromedriver.storage.googleapis.com/index.html?path=2.35/

(2) Change Path

크롬 드라이버를 글로벌하게 사용하기 위해서 경로를 변경해 줍니다. 경로를 수정하지 않으면 코드에서 크롬드라이버 경로를 항상 작성해 줘야 합니다.

$ mv ~/Download/chromedriver /usr/local/bin

2. install selenium package

selenium package를 설치해주기 위해서는 관리자 권한이 필요하기때문에 sudo를 앞에 붙여서 설치합니다.

$ sudo pip3 install selenium

Windows

1. install chrome driver

(1) Download chrome driver

아래의 링크에서 크롬 드라이버를 다운로드 합니다.

- https://sites.google.com/a/chromium.org/chromedriver/home
- https://chromedriver.storage.googleapis.com/index.html?path=2.35/

(2) 환경변수 추가

내컴퓨터 오른쪽 클릭 -> 속성 -> 고급시스템설정 -> 환경변수 -> 시스템변수의 Path 더블클릭 -> chromedriver path를 추가 (예: C:₩Users₩(사용자계정이름)₩chromdriver)

2. install selenium package

아나콘다 셀레니움 패키지 페이지(https://anaconda.org/conda-forge/selenium)에서 패키지 설치 명령어를 복사하여 cmd 창에서 실행하여 셀레니움을 설치합니다.

$ conda install -c conda-forge selenium

Selenium 기본 사용법

셀레니움은 http://www.seleniumhq.org/ 웹페이지에서 자세한 내용과 사용법을 확인하실수 있습니다. 셀레니움은 브라우져를 자동화하는 툴으로 테스팅을 목적으로 웹 어플리케이션을 자동화하는 목적으로 만들어졌지만 테스팅에 국한되서 사용되지는 않습니다. 웹 기반의 작업들을 자동화하여 업무 효율성을 높일수 있으며 다양한 언어와 다양한 브라우져를 지원합니다.
http://www.seleniumhq.org/about/platforms.jsp 웹 페이지에서 지원하는 브라우져와 지원하는 OS와 지원하는 언어를 확인하실수 있습니다. 셀레니움은 Firefox, Internet Explorer, Safari, Opera Chrome 브라우져를 지원합니다. 지원하는 OS로는 Microsoft Windows, Apple OS X, Linux를 지원하며, 지원하는 언어는 C#, Haskell, Java, Javascript, Object-C, Perl, PHP, Python, R, Ruby를 지원합니다.

1. selenium 패키지 호출

아래의 코드로 selenium의 webdriver를 호출 합니다. webdriver를 호출함으로 webdriver를 이용하여 위에 설치한 크롬 드라이버를 코드를 이용하여 자동화 할수 있습니다.

```
from selenium import webdriver
```

2. 브라우져 열기

셀레니움의 webdriver를 이용하여 Chrome 브라우져를 열어 객체를 driver 변수에 저장합니다. 아래의 코드를 실행하면 위에서 설치한 크롬 브라우져가 윈도우상에 열립니다.

```
driver =  webdriver.Chrome()
```

3. 페이지 이동

driver의 get을 이용하여 크롬 브라우져에서 페이지를 이동할수 있습니다. 아래의 코드는 naver웹 페이지로 이동합니다. url은 http를 꼭 포함해야 합니다.

```
driver.get("http://www.naver.com")
```

4. 윈도우 사이즈 조절

driver의 set_window_size를 이용하여 크롬 브라우져의 윈도우 사이즈를 조절할수 있습니다. 함수내의 첫번째 파라미터는 가로 사이즈를 나타내며 두번째 파라미터는 세로 사이즈를 나타냅니다. 사이즈의 단위 는 픽셀입니다.
아래의 코드는 가로 800픽셀, 세로 600픽셀로 윈도우 사이즈를 변경해 줍니다.

```
driver.set_window_size(800, 600)
```

5. 브라우져의 스크롤 위치 이동

브라우져의 스크롤 위치 이동은 selenium의 webdriver에 따로 기능이 없기 때문에 javascript 코드를 실행 하여 브라우져 스크롤 위치를 이동시킬수 있습니다. javascirpt의 window.scrollTo 함수의 첫번째 파라미터는 가로축 위치를 나타내며 아래 코드와 같이 200으로 설정을 하면 왼쪽에서 200픽셀만큼의 브라우져 스크롤 위치를 이동함을 의미합니다. 두번째 파라미터는 세로축의 위치를 나타내며 아래의 코드와 같이 300으로 지정을 하면 위에서 아래로 300픽셀 만큼 스크롤이 내려감을 의미합니다.

```
driver.execute_script("window.scrollTo(200, 300);")
```

6. 윈도우 객체 저장

driver의 current_window_handle을 이용하여 현재 윈도우의 객체를 main_window로 저장합니다. 이와 같이 윈도우의 객체를 저장하면 브라우져에 여러개의 탭이 있을때 다른 탭에서 저장되어 있는 윈도우 객체의 탭으로 돌아올수 있습니다. 객체의 값을 확인하면 window의 고유한 아이디 값을 확인 하실 수 있습니다.

```
main_window = driver.current_window_handle
main_window
```

7. 새로운 윈도우탭 열기

자바스크립트 코드를 실행시킴으로 새로운 탭을 열수 있습니다. 자바스크립트 코드로 새로운 탭을 오픈해도 탭에 대한 포커스는 처음에 열었던 페이지에 있어서 포커스를 변경해 주어야 합니다. 아래의 코드를 실행하면 새로운 탭으로 Google 페이지가 열리고 처음에 열었던 페이지에 Daum 페이지가 열립니다. 새로 열린 탭에 구글 페이지가 열리지 않은 이유는 포커스가 처음에 열었던 페이지에 있기때문입니다. 새로 열린 Google 페이지를 컨트롤 해주기 위해서는 포커스를 변경해 주어야 합니다.

```
driver.execute_script("window.open('https://www.google.com');")
```

```
driver.get('http://daum.net/')
```

8. 열려 있는 윈도우 확인

selenium driver의 window_handles를 이용하여 현재 열려있는 윈도우를 확인 할수 있습니다. 아래의 코드는 크롬 브라우져에서 열려 있는 윈도우에 대한 데이터를 windows 변수를 생성하여 저장하는 코드입니다. windows 변수에 저장되어 있는 데이터를 확인해보면 열려 있는 윈도우 탭의 객체데이터를 확인해 보실 수 있습니다.

```
windows = driver.window_handles
windows
```

9. 포커스 변경

selenium driver의 switch_to_window를 이용하여 탭의 포커스를 변경할수 있습니다. 아래의 코드는 위에서 windows로 크롬 브라우져에 있는 윈도우들중 1 위치에 있는 윈도우로 윈도우 탭의 포커스를 변경하고 윈도우 탭의 페이지를 페이스북 페이지로 이동하는 코드 입니다. 아래 코드를 실행하면 1위치의 윈도우 탭으로 포커스가 변경되고 1위치의 윈도우 탭이 페이스북 페이지로 이동됩니다.

```
driver.switch_to_window(windows[1])
driver.get(‘http://facebook.com‘)
```

아래의 코드는 앞에서 저장해 놓은 main_window 변수를 이용하여 메인페이지로 다시 포커스 변경합니다. 아래의 코드를 실행하면 첫번째 윈도우 탭에 열려있는 Daum 페이지가 있는 탭으로 이동됩니다.

```
driver.switch_to_window(main_window)
```

10. 페이지 새로고침

페이지 새로 고침은 자바스크립트 코드를 실행 시킴으로 페이지 새로 고침을 할수 있습니다. 아래의 코드처럼 excute_script를 이용하여 자바스크립트 코드를 실행시키는 함수를 작성하고 파라미터로 페이지를 새로고침하는 자바스크립트 코드인 location.relaod()를 넣어주면 페이지를 새로고침 합니다.

```
driver.execute_script("location.reload();")
```

11. alert 다루기

아래와 같이 alert이 나오는 자바스크립트 코드를 실행시키면 크롬 브라우져의 웹 페이지에서 alert 메시지가 나옵니다.

```
driver.execute_script("alert('selenium test');")
```

selenium의 switch_to_alert 함수를 이용하여 alert 변수에 alert에 해당되는 객체를 저장할수 있습니다. 또한 alert 객체의 text 값을 확인해보면 alert의 메시지를 문자열로 확인해 보실 수 있습니다.

```
alert = driver.switch_to_alert()
alert.text
```

alert 객체의 accept 함수를 실행키면 확인 버튼이 눌리면서 alert이 사라집니다.

```
alert.accept()
```

12. confirm 다루기

confirm은 alert과 비슷하지만 확인 버튼만 있는 alert과는 달리 취소버튼도 있습니다. selenium에서는 confirm의 확인버튼과 취소버튼을 누를수 있는 기능을 제공합니다.
아래의 코드는 confirm? 이라는 메시지가 있는 확인, 취소 두개의 버튼이 있는 창을 띄우는 자바스크립트 코드를 실행합니다. 아래의 코드를 실행하면 크롬 브라우져 현재의 웹페이지에서 confirm 창이 뜨게 됩니다.

```
driver.execute_script("confirm('confirm?');")
```

위의 alert 다루기에서와 같이 switch_to_alert을 이용하여 alert 객체를 받아오고 alert.text를 실행하시면 confirm창의 메시를 확인해 보실 수 있습니다.

```
alert = driver.switch_to_alert()
alert.text
```

alert의 accept 함수를 실행하시면 confirm의 확인 버튼이 실행되게 되며, dismiss를 실행하게 되면 취소 버튼이 실행되게 됩니다.

```
alert.accept()
alert.dismiss()
```

13. 키 입력하기

selenium driver의 send_keys를 이용하여 웹페이지의 input 태그에 문자열을 입력할수 있습니다. 아래의 코드는 Daum 페이지의 검색창에 “패스트캠퍼스” 문자열을 입력하는 코드입니다. 아래 코드의 find_element_by_css_selector는 html 코드의 태그 위치를 접근하는 함수로 파라미터로 css seletor를 넣어 주시면 파라미터로 넣은 selector의 태그 위치로 접근합니다. 그러므로 아래의 코드는 id값이 q인 html 태그의 위치에 “패스트캠퍼스”라는 문자열을 입력하게 됩니다.

```
driver.find_element_by_css_selector( "#q" ).send_keys( "패스트캠퍼스" )
```

14. 버튼 클릭하기

selenium의 driver의 click 함수를 이용하여 버튼을 클릭할수 있습니다. 아래의 코드는 검색버튼의 css selector 인 .ico_pctop.btn_search 태그에 접근하여 click 버튼을 누르는 동작을 하는 코드입니다. 아래의 코드를 실행하면 위에서 입력된 “패스트캠퍼스”키워드로 검색된 결과가 웹페이지에 나타납니다.

```
driver.find_element_by_css_selector(".ico_pctop.btn_search").click()
```

15. 윈도우 창 닫기

selenium dirver의 close를 이용하여 윈도우 창을 닫을수 있습니다. 아래와 같이 코드를 실행하면 현재 포커스 되어 있는 윈도우창이 닫히게 됩니다.

```
driver.close()
```

위의 코드는 하나의 윈도우 창을 닫습니다. 모든창을 모두 종료하고 싶으면 아래의 코드와 같이 window_handles를 이용하여 모든 윈도우 객체를 반복문으로 하나씩 가져와서 포커스를 가져온 윈도우로 이동한후에 close를 이용하여 윈도우 창을 닫게 되면 모든 창이 한번에 닫히게 됩니다.

```
for window in driver.window_handles:
    driver.switch_to_window(window)
    driver.close()
```

Selenium 파일 업로드

Selenium을 이용하여 브라우져에서 프로그램으로 파일을 업로드하거나 다운로드 할수 있습니다. google vision api 테스트 페이지에 이미지 파일을 업로드하는 예제에 대해서 알아보도록 하겠습니다.

File Upload

Selenium과 크롬 웹드라이버를 이용하여 google vision api 웹 페이지에 들어가서 이미지 파일을 업로드하고 결과 값을 가져오는 코드입니다.

1. 크롬 웹드라이버 실행과 google vision api 페이지로 이동

아래의 코드를 실행하여 크롬 웹드라이버를 실행하고 실행된 크롬 브라우져에 google vision api 페이지로 이동합니다.

```
from selenium import webdriver
driver =  webdriver.Chrome()
driver.get('https://cloud.google.com/vision/')
```

2. iframe 영역으로 포커스 변경

google vision api에서 파일 업로드하는 영역은 iframe으로 되어 있습니다. iframe은 웹페이지 안에 다른 URL을 가지는 웹페이지를 보여주는 기능이 있는 html 태그입니다. 그래서 selenium을 통해서 파일을 업로드 하려면 먼저 iframe으로 포커스를 변경해주어야 합니다. 아래의 코드는 id값이 vision_demo_section인 html 태그 아래 iframe 태그를 css selector를 이용하여 접근하고 그 selector 객체를 iframe 변서에 저장한후에 selenium의 driver의 switch_to_frame 함수를 이용하여 iframe 객체를 파라미터로 넣어 iframe으로 포커스를 변경하는 코드 입니다.

```
iframe = driver.find_element_by_css_selector("#vision_demo_section iframe")
driver.switch_to_frame(iframe)
```

iframe 으로 포커스가 아닌 원래 페이지로 포커스로 이동을 하려면 selenium driver의 switch_to_default_content 함수를 사용하시면 iframe에서 나와 원래 페이지로 포커스가 이동됩니다.

```
driver.switch_to_default_content()
```

3. 파일 업로드

iframe으로 포커스가 되어 있는 상태에서 아래와 같이 코드를 입력하시면, file_path 경로에 있는 이미지 파일이 업로드 됩니다. file_path는 업로드 하고자 하는 이미지에 대한 경로를 넣어 주시면 됩니다.

```
file_path = "/Users/radajin/Documents/fastcampus/marketer/day1/selenium/screenshot_element.png"
driver.find_element_by_css_selector(“#input").send_keys(file_path)
```

4. 결과 확인

아래 코드는 이미지를 분석한 결과가 나오고 나온 결과에서 Web 탭을 선택한후 가장 위에 있는 결과를 text로 출력합니다. text 데이터를 출력한 후에는 크롬 브라우져가 종료 됩니다.

```
driver.find_element_by_css_selector("#webDetection").click()
result = driver.find_elements_by_css_selector('.vs-web .name’)[0].text
print(result)
driver.close()
```

Selenium 웹 데이터 스크래핑

selenium을 이용하여 웹 데이터를 스크래핑 할수 있습니다. 아래 예제에서는 gmaket의 베스트 셀러 데이터중 상위 10개의 데이터를 가져오는 스크래핑 코드를 작성하겠습니다.

gmarket 베스트 셀러 상품 10개

1. 크롬 웹드라이버 실행과 gmarket 페이지로 이동

```
from selenium import webdriver

driver =  webdriver.Chrome()
driver.get('http://corners.gmarket.co.kr/Bestsellers')
```

2. 데이터를 가져와서 출력하기

```
bestlist_selector = driver.find_elements_by_css_selector("#gBestWrap .best-list")[1]
items = bestlist_selector.find_elements_by_css_selector("li")[:10]

for item in items:
    idx = item.find_element_by_css_selector("p").text # 순번
   content = item.find_element_by_css_selector(".itemname").text # 아이템 텍스트
    o_price = int(item.find_element_by_css_selector(".item_price .o-price").text[:-1].replace(",","")) # 오리지날 가격
    s_price = item.find_element_by_css_selector(".item_price .s-price").text # 판매 가격
    s_price = int(s_price.split(" ")[0][:-1].replace(",",""))
    print("{}\n제목 - {}\n정상가 - {}\n할인가 - {}\n".format(idx, content, o_price, s_price))
```

Naver 메일 데이터 스크래핑

1. 크롬 웹드라이버 실행과 naver 페이지로 이동

셀레니움 웹 드라이버와 패스워드를 가져올 pickle 모듈을 import 한후 main_url에는 로그인을 할 네이버 메인 페이지 URL을 저장하고, mail_url에는 네이버 메일 사이트 URL을 저장합니다. 크롬 웹드라이버를 실행 시키고 네이버 메인 페이지로 이동합니다.

```
from selenium import webdriver
import pickle

main_url = "http://naver.com"
mail_url = "http://mail.naver.com"

driver =  webdriver.Chrome()
driver.get(main_url)
```

2. 아이디 패스워드 입력 및 로그인

아래의 코드는 아이디 입력칸에 아이디를 입력하고 패스워드 입력칸에 패스워드를 입력하고 로그인 버튼을 클릭하는 코드입니다. 패스워드는 pw 변수에 저장하는데 노출을 막기위해서 pickle 파일로 저장한 pw를 불러왔습니다. pw 변수에 본인의 naver 메일의 pw을 넣으시면 됩니다.

```
pw = pickle.load(open("pw.pickle", "rb")) # pw 변수에 본인의 패스워드 입력
driver.find_element_by_css_selector( "#id" ).send_keys( "1221pdj" )
driver.find_element_by_css_selector( "#pw" ).send_keys( pw )
driver.find_element_by_css_selector("span.btn_login input[type=submit]").click()
```

3. 메일 페이지로 이동

로그인이 완료되면 메일 관련 데이터를 스크래핑 하기 위해 메일 페이지로 이동합니다

```
driver.get(mail_url)
```

4. 전체 메일 갯수 가져오기

네이버 메일 페이지에 있는 headUnreadNum 아이디 값을 가지는 전체 메일 갯수 데이터를 가져와서 출력해 줍니다.

```
total_mail = driver.find_element_by_css_selector("#headUnreadNum").text
total_mail = int(total_mail)
total_mail
```

5. 메일 제목, 보낸사람 이름 가져오기

css selector를 이용하여 제목과 이름 값을 가져와 출력해 줍니다.

```
titles = driver.find_elements_by_css_selector(".mail_title")
names = driver.find_elements_by_css_selector(".name")
for idx, title in enumerate(titles):
    print(idx+1, names[idx].text, titles[idx].text)
```

6. 로그아웃

로그 아웃 버튼의 href 속성 값을 가져와 로그아웃 페이지로 이동하여 로그아웃 해줍니다.

```
logout_url = driver.find_element_by_css_selector(".gnb_btn_login").get_attribute("href")
driver.get(logout_url)
```

7. 브라우져 닫기

selenium driver의 close함수를 이용하여 브라우져를 닫아줍니다.

```
driver.close()
```

Selenium 이미지 스크린샷

Selenium으로 브라우져 내의 화면을 이미지로 저장할수 있습니다. 이미지를 저장하고 썸네일을 만들거나 특정 위치의 이미지를 자를때 pillow 패키지를 사용합니다. pillow 패키지의 레퍼런스는 http://pillow.readthedocs.io/en/3.1.x/reference/Image.html 웹 사이트를 확인하시면 되며 설치는 pip를 이용하여 아래와 같이 설치 해주세면 됩니다.

Mac

$ pip3 install pillow

Windows

windows에서 최신버전을 설치하면 4.1.x 버전이 설치됩니다. 4.1.x 버전에서는 주피터 노트북에서 에러가 발생되므로 4.0.0 버전으로 설치해 주어야 합니다.
windows에서 plilow가 설치되어 있다면 언인스톨 한후에 4.0.0 버전으로 재설치 해줍니다.

install
$ conda install -c ananconda pillow==4.0.0

unitstall
$ conda uninstall pillow

1. time, selenium webdriver, pillow 모듈을 import

```
import time
from selenium import webdriver
from PIL import Image as pil
```

2. 브라우져를 열고 youtube 페이지로 이동하여 브라우져 화면을 이미지로 저장

크롬 브라우져를 열고 유투브 페이지의 200초 상태로 이동한후에 1초 후에 save_screenshot 함수를 호출하여 브라우져에 나오는 화면을 screenshot.png로 저장합니다. 유투브 url의 마지막에 &t=200s 이렇게 파라미터를 추가하면 유투브 영상의 특정 시간대부터 재생을 시킬수 있습니다. 페이지 이동후 1초의 time sleep을 주는 이유는 페이지 로딩이 완전히 되지 않았는데 브라우져 화면의 이미지를 저장하는것을 방지 하기 위함입니다.

```
driver =  webdriver.Chrome()
driver.get("https://www.youtube.com/watch?v=tXOJIBPwSyo&t=200s")
time.sleep(1)
driver.save_screenshot('screenshot.png')
```

3. 저장된 이미지파일의 사이즈와 이미지를 pillow를 이용하여 확인

pillow의 open함수를 이용하여 저장한 이미지를 주피터 노트북 상에서 확인할수 있으며 저장된 이미지의 사이즈도 확인이 가능합니다.

```
pil_im = pil.open('screenshot.png')
print(pil_im.size)
pil_im
```

4. pillow를 이용하여 thumbnail 이미지 만들기

pillow의 thumbnail 함수를 이용하여 이미지를 축소하여 썸네일 이미지로 저장할수 있습니다. 아래 코드는 512*512픽셀로 썸네일을 만들고 screenshot_thumnail.png로 이미지를 저장하고 open 함수를 이용하여 주피터 노트북에서 이미지를 확인 하는 코드 입니다.

```
pil_im.thumbnail((512, 512))
pil_im.save("screenshot_thumnail.png")
pil.open('screenshot_thumnail.png')
```

5. 특정영역의 이미지 자르기

(1) 특정 selector의 element의 위치와 사이즈 데이터를 확인합니다.

```
element = driver.find_element_by_css_selector('#movie_player')
location = element.location
size = element.size
location, size
```

(2) 저장했던 원본이미지를 열어 top, bottom, right, left의 위치를 계산합니다.

* 모니터의 해상도에 따라서 위치를 계산하는 방법에 차이가 있을수 있습니다. (레티나 디스플레이의 경우 2를 곱해주고 일반 hd 화면에서는 2를 안 곱해주면 됩니다.)

```
pil_im = pil.open('screenshot.png')
left = location['x'] * 2
top = location['y'] * 2
right = left + size['width'] * 2
bottom = top + size['height'] * 2
area = (left, top, right, bottom)
area
```

(3) pillow를 이용하여 계산된 위치로 이미지를 잘라줍니다.

```
pil_im = pil_im.crop(area)
```

(4) 자른 이미지를 저장하고 저장된 이미지를 보여줍니다.

```
pil_im.save("screenshot_element.png")
pil.open('screenshot_element.png')
```