

3. 동적 크롤링

1. Selenium 개요
2. Selenium을 사용한 웹 페이지 스크래핑
3. 카페 및 서점 동적 웹 페이지 스크래핑 실습

□ 정적 웹 페이지 VS 동적 웹 페이지

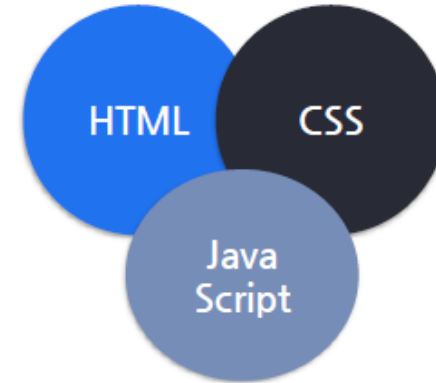
정적 웹 페이지

- 웹 서버에서 전송된 웹 페이지의 소스에서 화면에 렌더링된 내용을 모두 찾을 수 있는 경우
- HTML만으로 작성되거나 HTML과 CSS 기술 등으로 구현된 경우



동적 웹 페이지

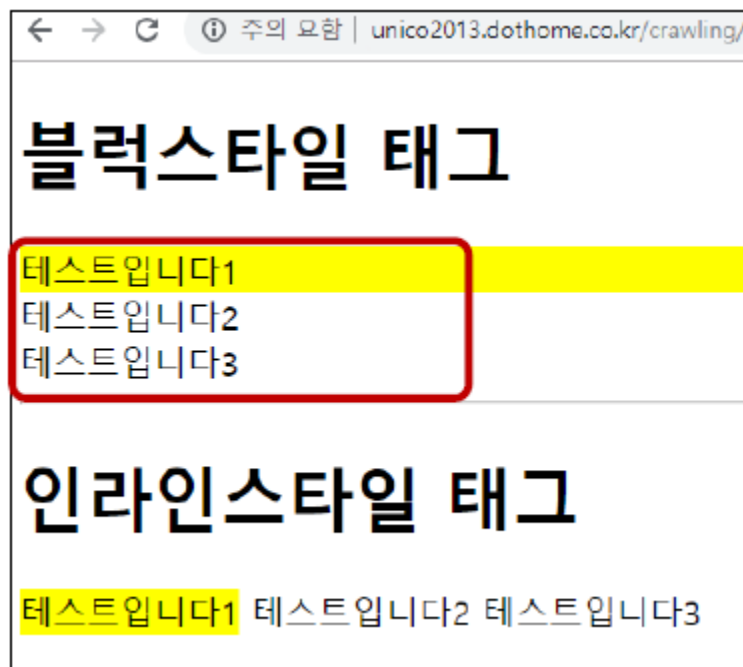
- 웹 서버에서 전송된 웹 페이지의 소스에서 화면에 렌더링된 내용을 일부 찾을 수 없는 경우
- HTML과 CSS 기술 외에 JavaScript 프로그래밍 언어로 브라우저에서 실행시킨 코드에 의해 웹 페이지의 내용을 렌더링 시 자동 생성



□ 정적 웹 페이지 VS 동적 웹 페이지

▣ 정적 웹 페이지 화면

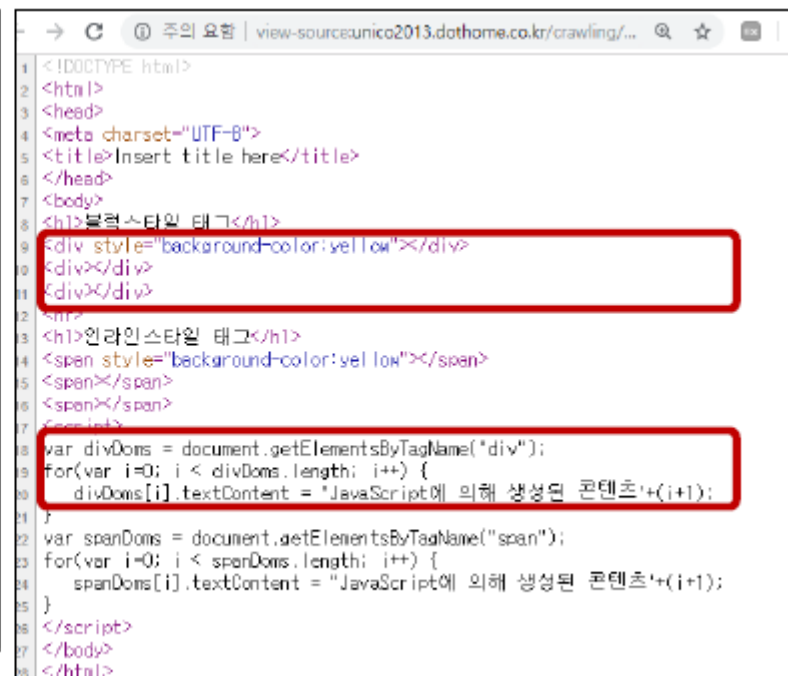
- 화면에 렌더링된 각 태그의 콘텐츠가 페이지의 소스에서도 모두 보여짐



□ 정적 웹 페이지 VS 동적 웹 페이지

▣ 동적 웹 페이지 화면

- 화면에 렌더링된 일부 태그의 콘텐츠를 페이지의 소스에서 찾아볼 수 없음
- <div> 태그나 태그처럼 소스코드에서 그내용을 찾아볼 수 없음



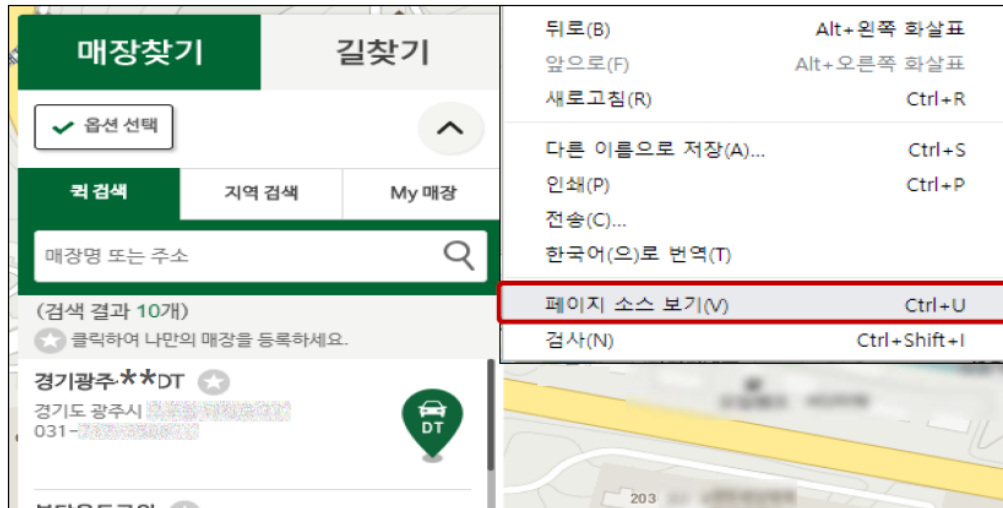
정적 웹 페이지와 동적 웹 페이지

정적·동적 콘텐츠 여부 체크

소스 코드 점검을 위해 페이지에서
마우스 오른쪽 버튼 클릭

팝업 메뉴 출력

페이지 소스 보기 메뉴 클릭



오른쪽 상단의
'크롬 맞춤설정 및 제어' 메뉴 클릭

풀다운 메뉴 출력

찾기 메뉴 선택



정적 웹 페이지와 동적 웹 페이지

정적·동적 콘텐츠 여부 체크

광주를 입력하면 0/0출력

이 단어가 소스 코드에 존재하지 않음

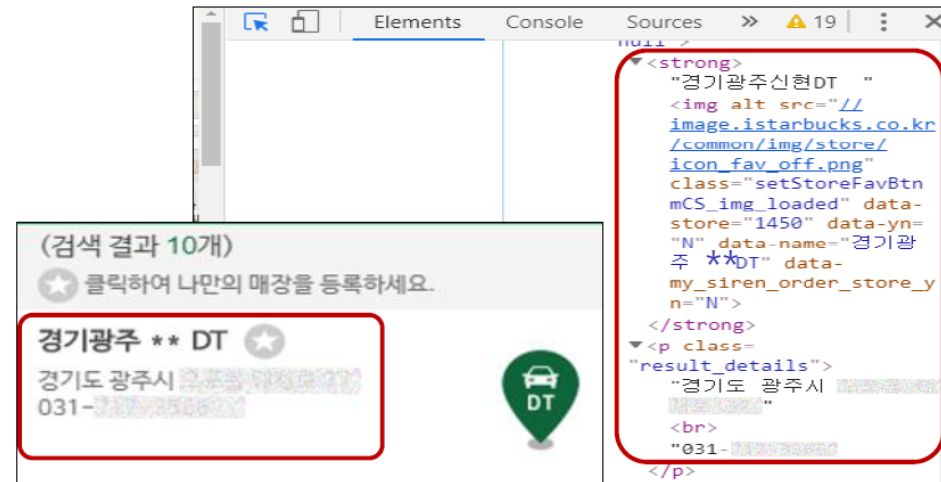
JavaScript 코드에 의해 생성되는 동적 콘텐츠임



Elements 탭 선택

렌더링된 내용의 태그 영역을 찾음

화면과 같은 태그 정보 출력



□ 정적 웹 페이지와 동적 웹 페이지

▣ 정적·동적 콘텐츠 여부 체크

- 서버로부터 전송된 소스에는 없으나 렌더링된 내용에는 있는 것이 **동적 콘텐츠**
- 이런 콘텐츠를 포함하고 있는 페이지는 **동적 웹 페이지**임

□ 정적 웹 페이지와 동적 웹 페이지

- ▣ 동적 웹 페이지에 의해 렌더링된 동적 콘텐츠의 스크래핑

Selenium이라는 웹 브라우저를
자동화하는 도구 모음 사용

Selenium

- 다양한 플랫폼과 언어 지원
- 이용하는 브라우저 자동화 도구 모음

- WebDriver라는 API를 통해 운영체제에 설치된 크롬이나 파이어폭스 등의 브라우저를 기동시키고 웹 페이지를 로드하고 제어
- 브라우저를 직접 동작시킨다는 것은 JavaScript에 의해 생성되는 콘텐츠와 Ajax 통신등을 통해 뒤늦게 브라우저에 로드되는 콘텐츠를 처리할 수 있다는것을 의미

2. Selenium 개요

□ Selenium이란?

- ▣ 주로 웹앱을 테스트하는데 이용하는 프레임워크
- ▣ Webdriver라는 API를 통해 운영체제에 설치된 Chrome 등 브라우저를 제어함
- ▣ 브라우저를 직접 동작시켜 JavaScript를 이용해 비동기적으로, 혹은 뒤늦게 불러와지는 콘텐츠들을 가져올 수 있다.
- ▣ 공식 홈페이지(<http://www.seleniumhq.org/>)
- ▣ Selenium with Python : <http://selenium-python.readthedocs.io/index.html>

□ Selenium 개발환경 구축

▣ WebDriver API

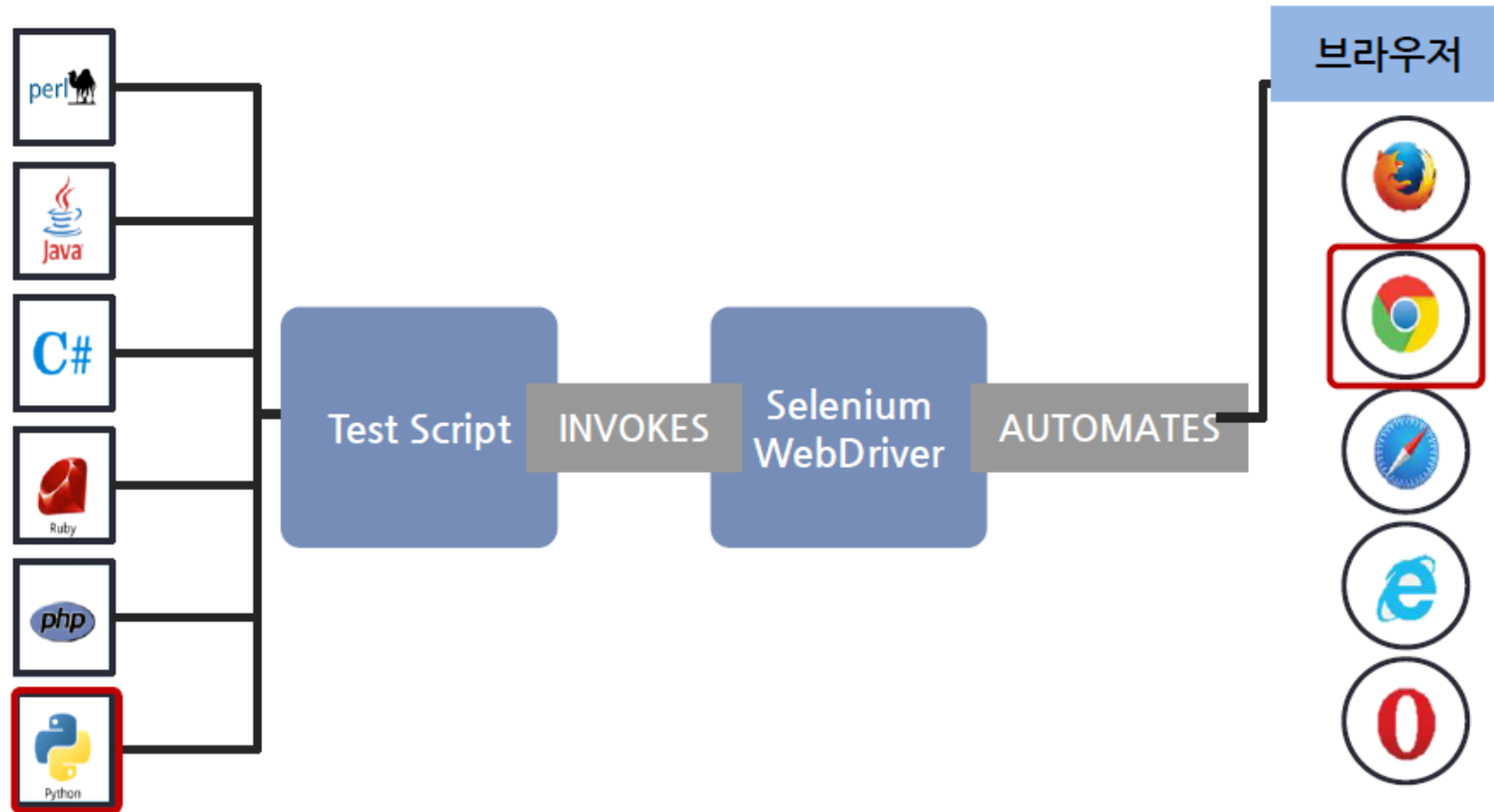
- 간결한 프로그래밍 인터페이스를 제공 하도록 설계
- 동적 웹 페이지를 보다 잘 지원할 수 있도록 개발

▣ WebDriver의 목표

- 최신 고급 웹 응용 프로그램 테스트 문제에 대한 향상된 지원과 잘 디자인된 객체지향 API 제공
- 자동화를 위한 각 브라우저의 기본 지원을 사용하여 브라우저를 직접 호출

□ Selenium 개발환경 구축

▣ WebDriver



□ Selenium 개발환경 구축

▣ Selenium 설치

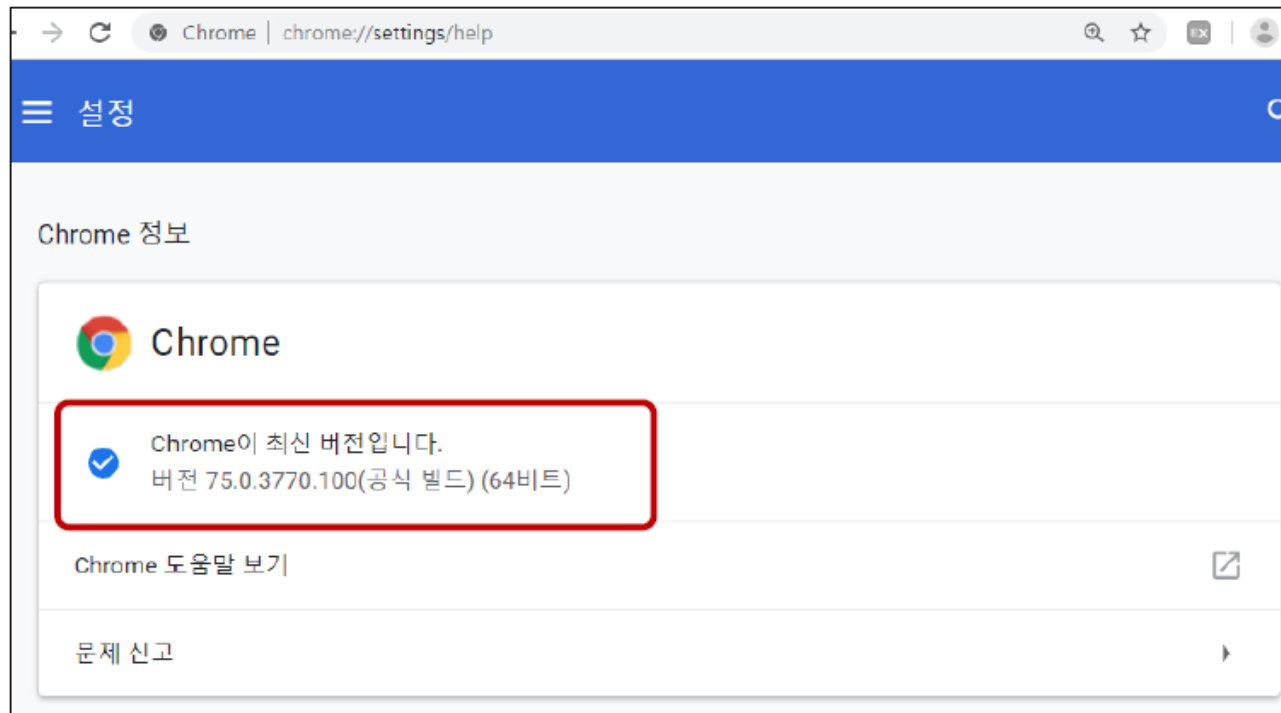
```
pip install selenium  
conda install selenium
```

```
C:\WINDOWS\system32\cmd.exe - conda install selenium  
Microsoft Windows [Version 10.0.17134.829]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\Samsung>conda install selenium  
Collecting package metadata: |
```

□ Selenium 개발환경 구축

▣ 크롬드라이버(Chrome Driver) 설치

- Selenium의 Web Driver에 의해 제어되는 크롬 드라이버 설치를 위해 시스템에 설치된 크롬 브라우저의 버전 체크

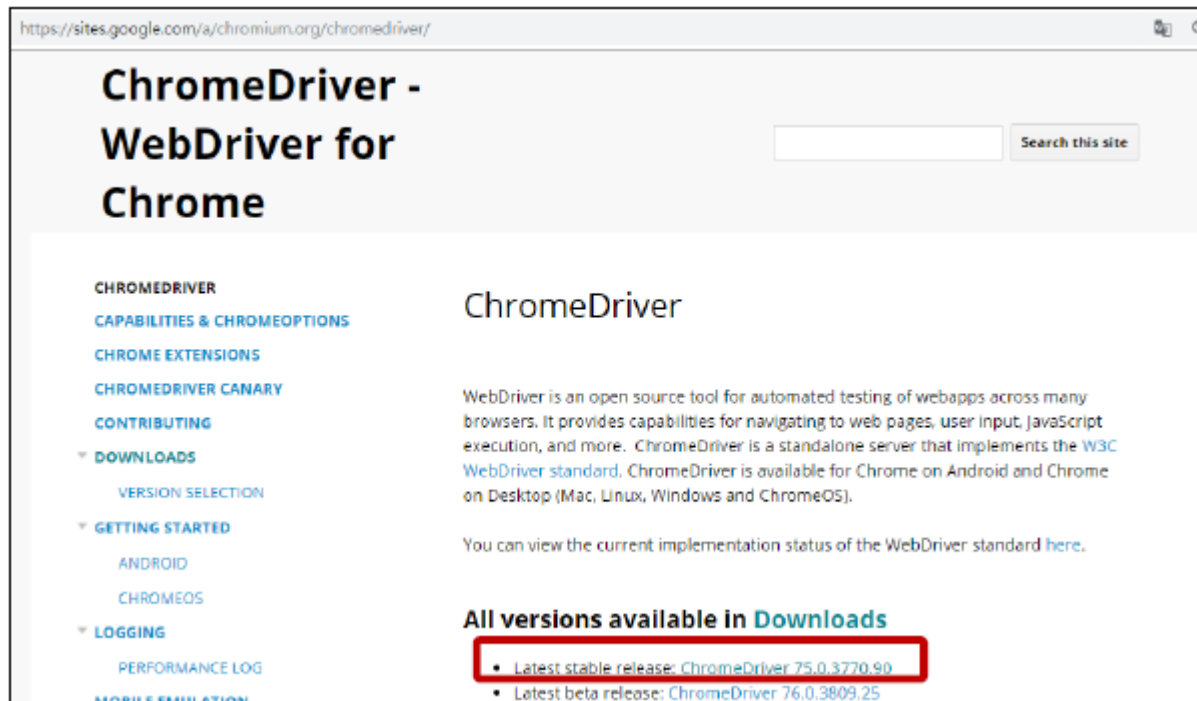


□ Selenium 개발환경 구축

▣ 크롬드라이버(Chrome Driver) 설치

- 다음 사이트에서 시스템에 설치된 크롬 브라우저와 동일 버전의 링크 클릭

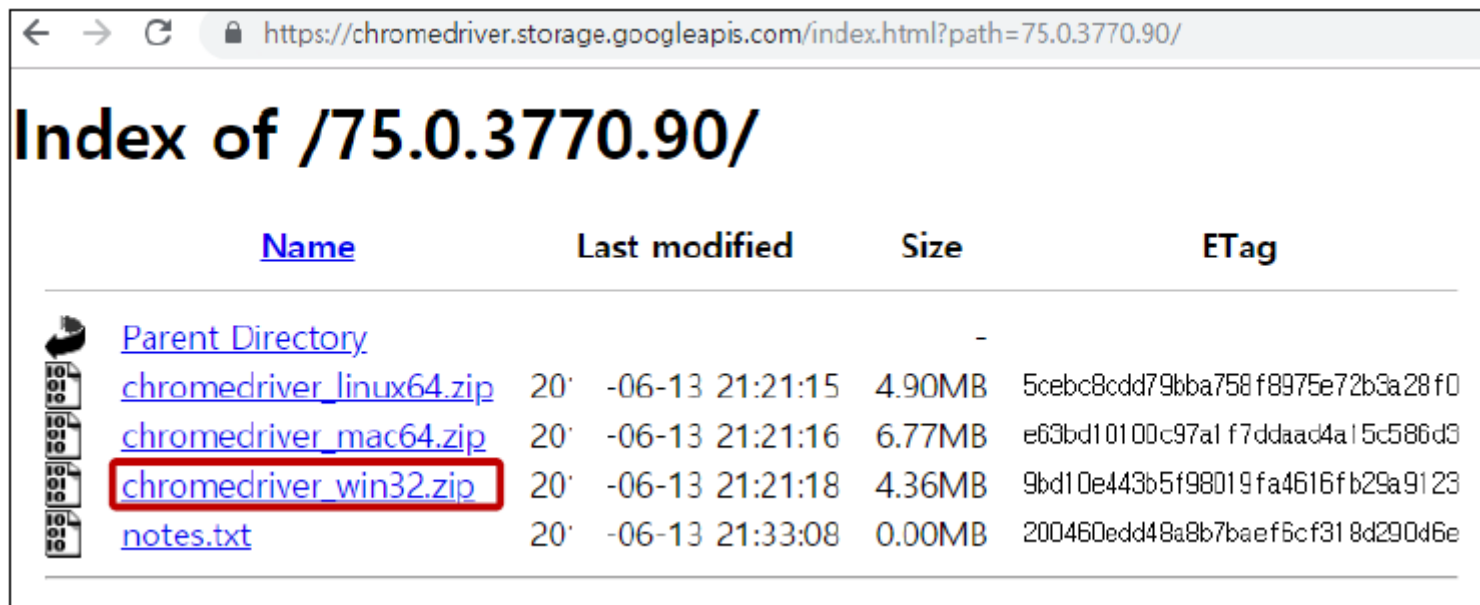
<https://sites.google.com/a/chromium.org/chromedriver/>








□ Selenium 개발환경 구축

▣ 크롬드라이버(Chrome Driver) 설치

- chromedriver_win32.zip을 다운로드
- 압축을 풀고 생성되는 chromedriver.exe를 적당한 디렉토리에 복사
- 이 과정에서는 c:/Temp 폴더에 복사



Index of /75.0.3770.90/				
Name	Last modified	Size	ETag	
 Parent Directory		-		
 chromedriver_linux64.zip	20' -06-13 21:21:15	4.90MB	5ceb8cdd79bba758f8975e72b3a28f0	
 chromedriver_mac64.zip	20' -06-13 21:21:16	6.77MB	e63bd10100c97a1f7ddaad4a15c586d3	
 chromedriver_win32.zip	20' -06-13 21:21:18	4.36MB	9bd10e443b5f98019fa4616fb29a9123	
 notes.txt	20' -06-13 21:33:08	0.00MB	200460edd48a8b7baef6cf318d290d6e	

3. Selenium 사용

□ Selenium 개발환경 구축

▣ Selenium을 사용한 크롬 브라우저 제어 예제 테스트0+

```
파일명 : exam6_1.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome('C:/Temp/chromedriver')
print("WebDriver 객체 : ", type(driver))

driver.get('http://www.google.com/ncr')
target=driver.find_element_by_css_selector("[name = 'q']")
print("찾아온 태그 객체 : ", type(target))
target.send_keys('파이썬')
target.send_keys(Keys.ENTER)
#driver.quit()
```

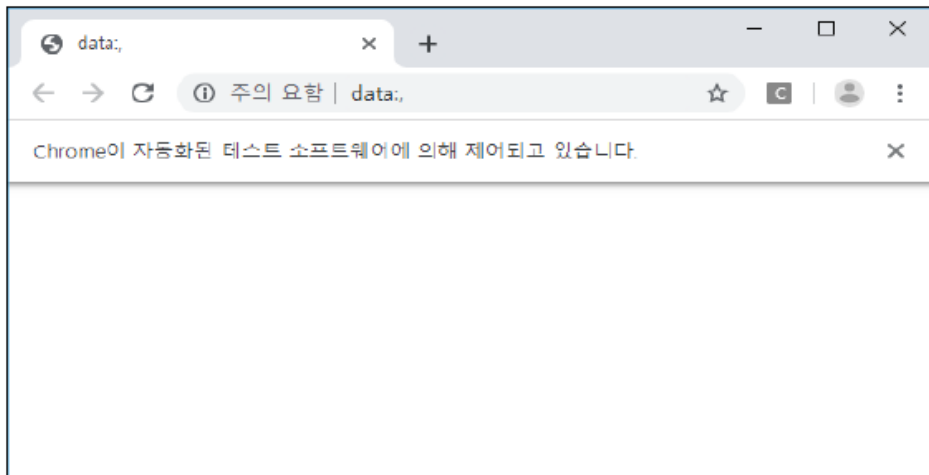

□ Selenium API 소개

▣ WebDriver 객체 생성

- 다음 코드를 수행 시켜서 크롬드라이버를 기반으로 selenium.webdriver.chrome.webdriver.WebDriver 객체 생성

```
driver = webdriver.Chrome('C:/Temp/chromedriver')
```

- 아규먼트로 chromedriver.exe 파일이 존재하는 디렉토리와 파일명에 대한 패스정보 지정 하면 Selenium에 의해 관리되는 크롬브라우저가 기동됨



□ 메서드를 사용한 웹페이지 파싱

▣ 페이지 가져오기

- `selenium.webdriver.chrome.webdriver.WebDriver` 객체의 `get()` 메서드사용
 - 크롤링하려는 웹페이지를 제어하는 크롬브라우저에 로드하고 렌더링

```
driver.get('http://www.google.com/ncr')
```

- WebDriver가 웹 페이지의 완전한 로드를 보장할 수 없음
 - 경우에 따라 페이지 로드 완료 또는 시작전에 *WebDriver*가 제어권을 반환할 수 있음
 - 견고성을 확보하려면 *Explicit & Implicit Waits*를 사용하여 요소가 페이지에 존재할 때까지 기다려야 함

```
driver.implicitly_wait(3)  
driver.get('http://www.google.com/ncr')
```

□ 메서드를 사용한 웹 페이지 파싱

▣ 요소 찾기

- WebDriver의 요소 찾기는 WebDriver 객체 및 WebElement 객체에서 제공되는 메서드들을 사용

■ 태그의 id 속성값으로 찾기

```
byId = driver.find_element_by_id('btype')
```

또는

```
from selenium.webdriver.common.by import By
```

```
byId = driver.find_element(by=By.ID, value='btype')
```

□ 메서드를 사용한 웹 페이지 파싱

▣ 요소 찾기

- 태그의 class 속성값으로 찾기

```
target =  
driver.find_element_by_class_name('quickResultLst  
Con')
```

또는

```
target = driver.find_element(By.CLASS_NAME,  
"quickResultLstCon")
```

□ 메서드를 사용한 웹 페이지 파싱

▣ 요소 찾기

■ 태그명으로찾기

```
byTagName =  
driver.find_element_by_tag_name('h1')  
또는  
byTagName = driver.find_element(By.TAG_NAME,  
'h1')
```

□ 메서드를 사용한 웹 페이지 파싱

▣ 요소 찾기

■ 링크 텍스트로 태그 찾기

```
byLinkText =  
driver.find_element_by_link_text('파이썬 학습  
사이트')  
  
또는  
  
byLinkText = driver.find_element(By.LINK_TEXT,  
'파이썬 학습 사이트')
```

```
<a href="https://www.python.org/">파이썬 학습 사이트</a>
```

□ 메서드를 사용한 웹 페이지 파싱

▣ 요소 찾기

■ 부분 링크 텍스트로 태그 찾기

```
byLinkText =  
driver.find_elements_by_partial_link_text('사이트')  
또는  
byLinkText =  
driver.find_element(By.PARTIAL_LINK_TEXT,  
'사이트')
```

□ 메서드를 사용한 웹 페이지 파싱

▣ 요소 찾기

■ CSS 선택자로 태그 찾기

```
byCss1 =  
driver.find_element_by_css_selector('section>h2')  
또는  
byCss1 = driver.find_element(By.CSS_SELECTOR,  
'section>h2')
```


□ 메서드를 사용한 웹 페이지 파싱

▣ 요소 찾기

■ Xpath로 태그 찾기

```
byXpath1 =  
driver.find_element_by_xpath('//*[@id="f_subtitle"]  
)  
또는  
byXpath1 = driver.find_element(By.XPATH,  
'//*[@id="f_subtitle"]')
```

■ 조건에 맞는 요소한개 찾기: *WebElement* 객체리턴

```
driver.find_element_by_xxx("xxx에 알맞는 식")
```

■ 조건에 맞는 모든 요소 찾기: *list* 객체리턴

```
driver.find_elements_by_xxx("xxx에 알맞는 식")
```

□ 메서드를 사용한 웹 페이지 파싱

▣ 요소의 정보 추출

```
element = driver.find_element_by_id("element_id")  
# 태그명  
element.tag_name  
# 텍스트 형식의 콘텐츠  
element.text  
# 속성값  
element.get_attribute('속성명')
```

□ 실습(정적 스크래핑)

```
import urllib.request
from bs4 import BeautifulSoup
#서버 접속
server = urllib.request.urlopen("https://www.istarbucks.co.kr/store/store_map.do")

response =server.read().decode()
bs = BeautifulSoup(response, "html.parser")
li = bs.find_all('li', class_="quickResultLstCon")
print(li)
```

□ 실습(동적 스크리핑)

```
from selenium import webdriver

driver = webdriver.Chrome('C:/Temp/chromedriver')
driver.implicitly_wait(3)
driver.get("https://www.istarbucks.co.kr/store/store_map.do")
target=driver.find_element_by_class_name("quickResultLstCon")

print(type(target))
print(type(target.text))
print(target.text)
driver.quit()
```

□ 실습

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

driver = webdriver.Chrome('C:/Temp/chromedriver')
print("webdriver 객체 : ", type(driver))

driver.get('http://www.naver.com/')
target=driver.find_element_by_css_selector("[name = 'query']")
print("태그 객체 : ", type(target))
target.send_keys('파이썬')
target.send_keys(Keys.ENTER)
driver.quit()
```

□ 실습

```
#파일명 :  
from selenium import webdriver  
  
driver = webdriver.Chrome('C:/Temp/chromedriver')  
print("webdriver 객체 : ", type(driver))  
  
driver.get('http://www.naver.com/')  
target=driver.find_element_by_id("query")  
print("태그 객체 : ", type(target))  
target.send_keys('파이썬')  
target.submit()  
driver.quit()
```

□ 실습

#파일명 :

```
from selenium import webdriver
```

```
driver = webdriver.Chrome('C:/Temp/chromedriver')
```

```
print("webdriver 객체 : ", type(driver))
```

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

```
target=driver.find_element_by_class_name("input_text")
```

```
print("태그 객체 : ", type(target))
```

```
target.send_keys('파이썬')
```

```
target.submit()
```

```
driver.quit()
```

3. Selenium 사용

□ Selenium으로 사이트 브라우징

```
from selenium import webdriver
```

```
## Chrome의 경우 | 아까 받은 chromedriver의 위치를 지정해준다.
```

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

□ 로딩 대기

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

```
driver.implicitly_wait(3)
```

```
## url에 접근한다.
```

```
driver.get('https://google.com')
```


3. Selenium 사용

- URL에 접근하는 api

```
get('http://url.com')
```

- 페이지의 단일 element에 접근하는 api

```
find_element_by_name('HTML_name')
```

```
find_element_by_id('HTML_id')
```

```
find_element_by_xpath('/html/body/some/xpath')
```

- 페이지의 여러 elements에 접근하는 api

```
find_element_by_css_selector('#css > div.selector')
```

```
find_element_by_class_name('some_class_name')
```

```
find_element_by_tag_name('h1')
```

```
#파일명 : exercise_solution1.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
```

```
driver = webdriver.Chrome('C:/Temp/chromedriver')
print("webdriver 객체 : ", type(driver))
```

```
driver.get('http://www.naver.com/')
target=driver.find_element_by_css_selector("[name = 'query']")
print("태그 객체 : ", type(target))
target.send_keys('파이썬')
target.send_keys(Keys.ENTER)
driver.quit()
```

```
#파일명 : exercise_solution1.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome('C:/Temp/chromedriver')
print("webdriver 객체 : ", type(driver))

driver.get('http://www.naver.com/')
target=driver.find_element_by_css_selector("[name = 'query']")
print("태그 객체 : ", type(target))
target.send_keys('파이썬')
target.send_keys(Keys.ENTER)
driver.quit()
```

□ XPATH를 이용하여 크롤링하기

- ▣ 마크업에서 요소를 정의하기 위해 path 경로를 사용하는 방법
- ▣ find_element_by_xpath(), find_elements_by_xpath() 메서드로 검색 가능

/ : 절대경로를 나타냄

// : 문서내에서 검색

//@href : href 속성이 있는 모든 태그 선택

//a[@href='http://google.com'] : a 태그의 href 속성에 <http://google.com> 속성값을 가진 모든 태그 선택

(//a)[3] : 문서의 세 번째 링크 선택

(//table)[last()] : 문서의 마지막 테이블 선택

(//a)[position() < 3] : 문서의 처음 두 링크 선택

//table/tr/* 모든 테이블에서 모든 자식 tr 태그 선택

//div[@*] 속성이 하나라도 있는 div 태그 선택

□ 네이버 로그인하기

```
from selenium import webdriver  
driver = webdriver.Chrome('chromedriver')  
driver.implicitly_wait(3)  
driver.get('https://nid.naver.com/nidlogin.login')  
driver.find_element_by_name('id').send_keys('naver_id')  
driver.find_element_by_name('pw').send_keys('mypassword1234')  
## 로그인 버튼을 눌러주자.  
driver.find_element_by_xpath('//*[@id="log.login"] /fieldset/input').click()
```

□ 네이버 페이지의 주문 내역 페이지 가져오기

▣ 로그인 된 상태에서 진행

```
from bs4 import BeautifulSoup

driver.get('https://order.pay.naver.com/home')
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
notices = soup.select('div.goods_item > div > a > p')

for n in notices:
    print(n.text.strip())
```

3. 카페 및 서점 동적 웹 페이지 스크래핑 실습

- Selenium을 활용한 웹 크롤링과 스크래핑을 고려 해야하는 경우
 - ▣ 화면에 렌더링된 웹 페이지의 내용을 서버로부터 전송된 소스코드에서 찾을 수 없는 경우
 - ▣ 페이지 내의 링크를 클릭할 때 이동되는 페이지의 URL이 주소 필드에 출력되지 않는 경우
 - ▣ 웹 페이지를 크롤링하기전에 로그인 과정을 거쳐서 인증처리를 해야하는 경우
 - ▣ 추출하려는 웹페이지의 내용이 스크롤과 같은 이벤트가 발생해야 화면에 렌더링 되는 경우
 - ▣ 버튼을 클릭해야 웹페이지의 콘텐츠가 출력되는 경우