

# 빅데이터 수집시스템 개발



## 동적 페이지의 웹 크롤링 구현

## 학습목표

- Selenium을 이용하여 동적 웹 페이지의 내용을 스크래핑할 수 있다.
- Selenium을 이용한 웹 스크래핑의 API 사용 방법과 구현 방법을 설명할 수 있다

## 학습내용

- Selenium의 개요
- Selenium을 사용한 웹 페이지 스크래핑

# Selenium의 개요



## 1 정적 웹 페이지와 동적 웹 페이지

### 1 정적 웹 페이지와 동적 웹 페이지의 구분

정적  
웹 페이지

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



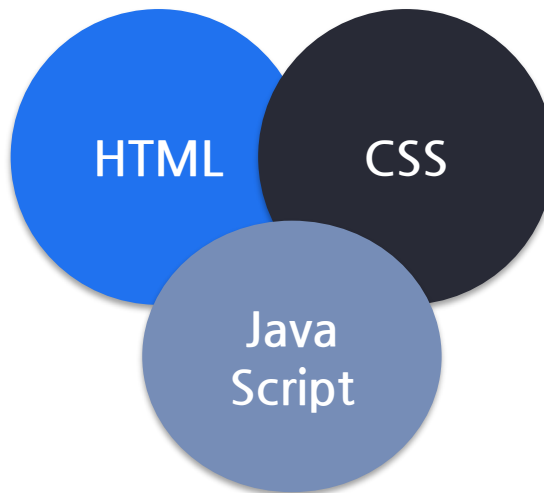
# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 1 정적 웹 페이지와 동적 웹 페이지의 구분

#### 동적 웹 페이지

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

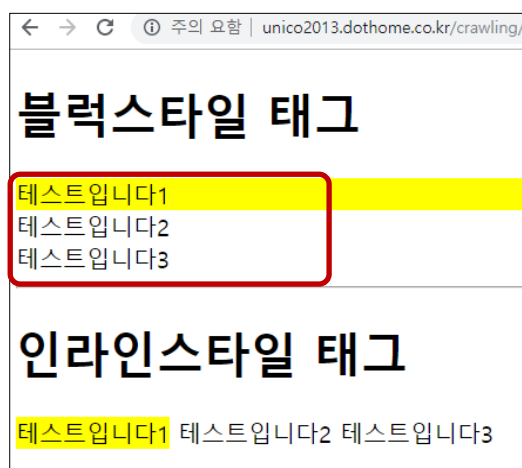


# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 2 정적 웹 페이지 화면

- 화면에 렌더링된 각 태그의 콘텐츠가 페이지의 소스에서도 모두 보여짐

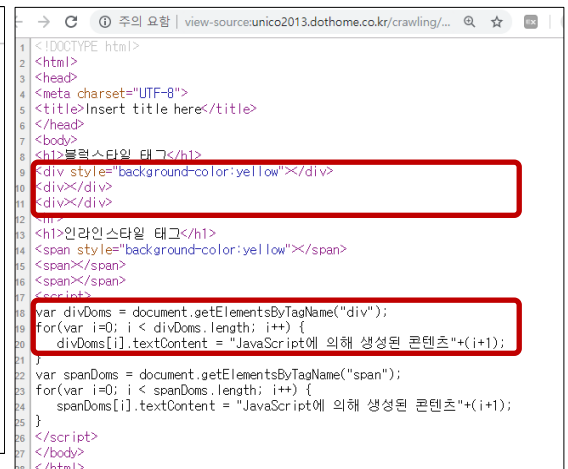


# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 3 동적 웹 페이지 화면

- 화면에 렌더링된 일부 태그의 콘텐츠를 페이지의 소스에서 찾아볼 수 없음
- <div> 태그나 <span> 태그처럼 소스 코드에서 그 내용을 찾아볼 수 없음



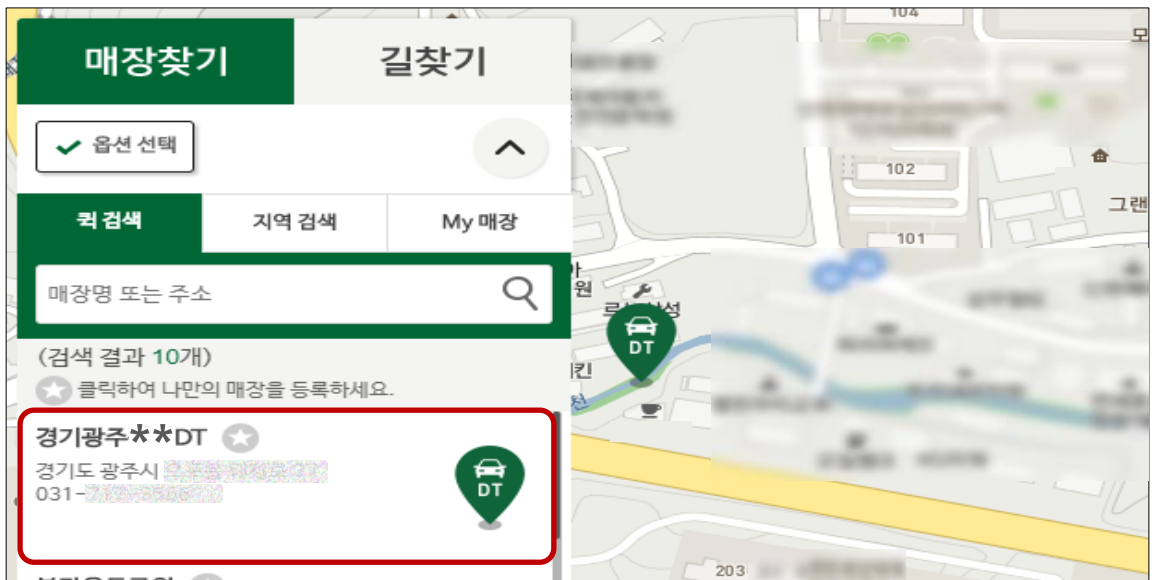
# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 4 정적·동적 콘텐츠 여부 체크

- 카페 웹 사이트의 매장 정보 제공 웹 페이지

#### 1 '경기광주\*\*DT'라는 매장명과 전화번호,주소 등 스크래핑



# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 4 정적·동적 콘텐츠 여부 체크

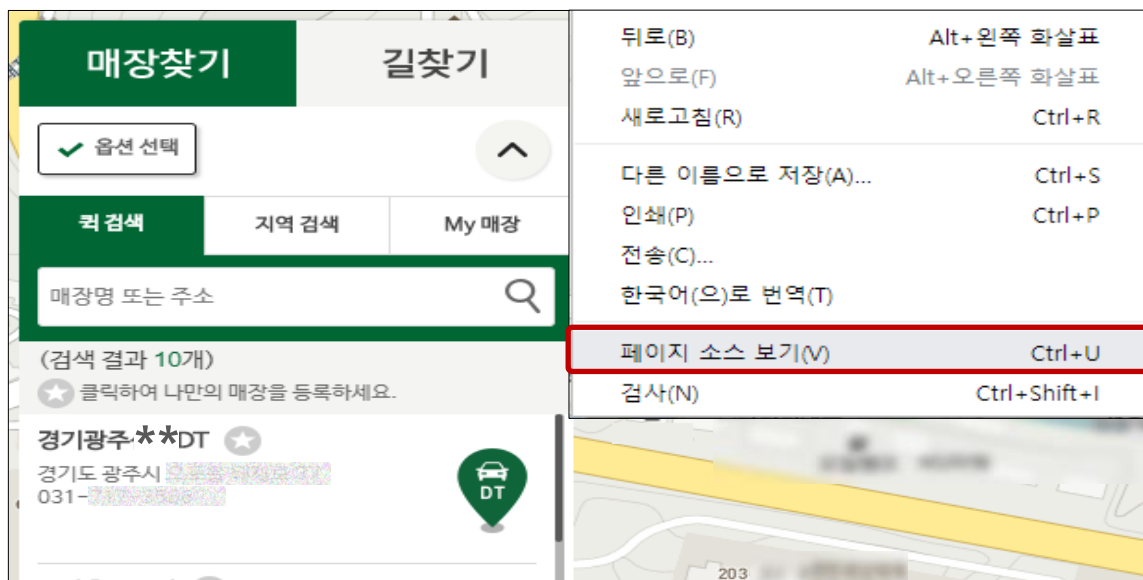
- 카페 웹 사이트의 매장 정보 제공 웹 페이지

## 2 정적 콘텐츠 체크

소스 코드 점검을 위해 페이지에서  
마우스 오른쪽 버튼 클릭

팝업 메뉴 출력

페이지 소스 보기 메뉴 클릭





# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 4 정적·동적 콘텐츠 여부 체크

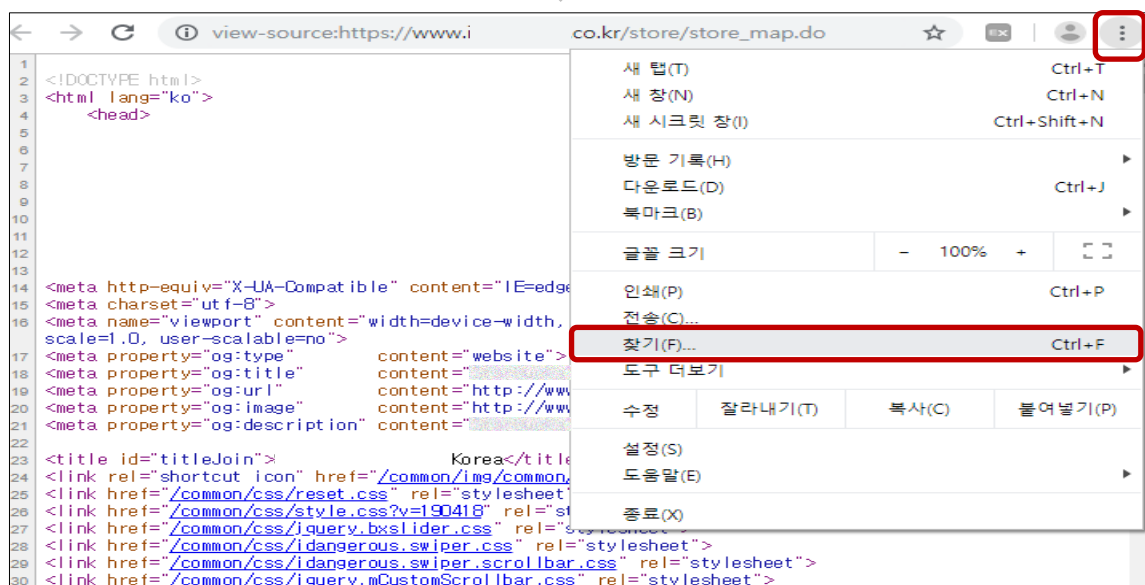
- 카페 웹 사이트의 매장 정보 제공 웹 페이지

### 3 소스창 출력

오른쪽 상단의  
'크롬 맞춤설정 및 제어' 메뉴 클릭

풀다운 메뉴 출력

찾기 메뉴 선택



# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 4 정적·동적 콘텐츠 여부 체크

- 카페 웹 사이트의 매장 정보 제공 웹 페이지

### 4 동적 콘텐츠 체크

광주를 입력하면 0/0출력

이 단어가 소스 코드에 존재하지 않음

JavaScript 코드에 의해 생성되는 동적 콘텐츠임



# Selenium의 개요

## 1 정적 웹 페이지와 동적 웹 페이지

### 4 정적·동적 콘텐츠 여부 체크

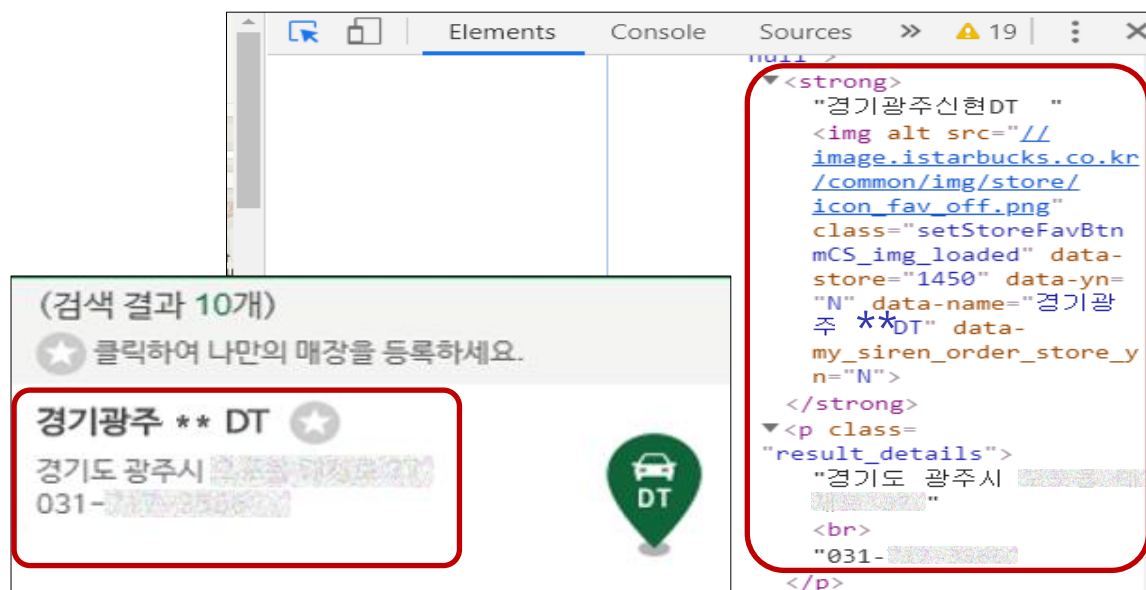
- 카페 웹 사이트의 매장 정보 제공 웹 페이지

### 5 크롬 브라우저의 개발자 도구

Elements 탭 선택

렌더링된 내용의 태그 영역을 찾음

화면과 같은 태그 정보 출력



# Selenium의 개요



## 1 정적 웹 페이지와 동적 웹 페이지

### 4 정적·동적 콘텐츠 여부 체크

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

# Selenium의 개요



## 1 정적 웹 페이지와 동적 웹 페이지

### 5 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 통신 등을 통해 뒤늦게 브라우저에 로드되는 콘텐츠를 처리할 수 있다는 것을 의미

# Selenium의 개요

## 2 Selenium 개발 환경 구축

### 1 Selenium

- Selenium의 홈페이지

<https://www.seleniumhq.org/>

- Selenium에 대한 모든 정보들을 얻을 수 있음

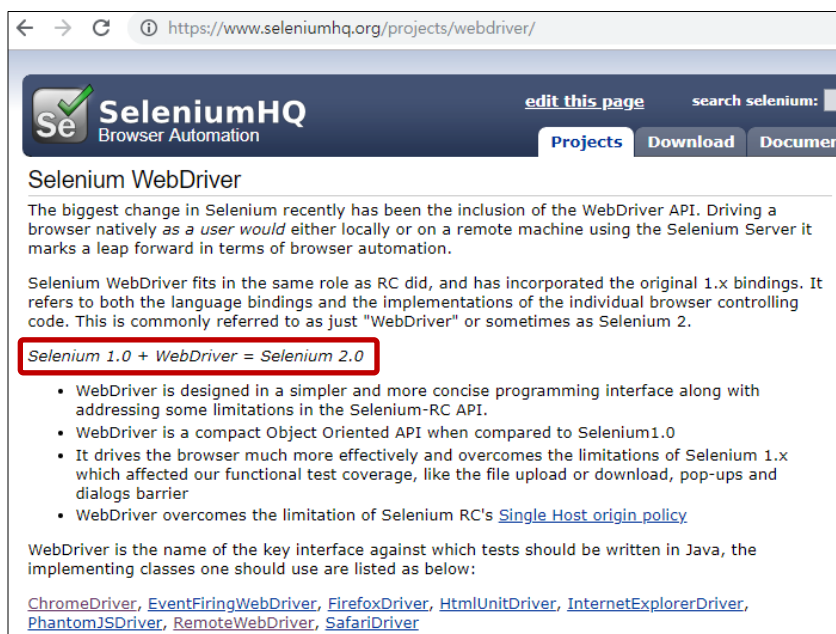


# Selenium의 개요

## 2 Selenium 개발 환경 구축

### 1 Selenium

- Selenium 2.0에서는 WebDriver API 지원



← → ↻ ⓘ https://www.seleniumhq.org/projects/webdriver/

**SeleniumHQ**  
Browser Automation

[edit this page](#) [search selenium:](#)

[Projects](#) [Download](#) [Document](#)

### Selenium WebDriver

The biggest change in Selenium recently has been the inclusion of the WebDriver API. Driving a browser natively *as a user would* either locally or on a remote machine using the Selenium Server it marks a leap forward in terms of browser automation.

Selenium WebDriver fits in the same role as RC did, and has incorporated the original 1.x bindings. It refers to both the language bindings and the implementations of the individual browser controlling code. This is commonly referred to as just "WebDriver" or sometimes as Selenium 2.

**Selenium 1.0 + WebDriver = Selenium 2.0**

- WebDriver is designed in a simpler and more concise programming interface along with addressing some limitations in the Selenium-RC API.
- WebDriver is a compact Object Oriented API when compared to Selenium1.0
- It drives the browser much more effectively and overcomes the limitations of Selenium 1.x which affected our functional test coverage, like the file upload or download, pop-ups and dialogs barrier
- WebDriver overcomes the limitation of Selenium RC's [Single Host origin policy](#)

WebDriver is the name of the key interface against which tests should be written in Java, the implementing classes one should use are listed as below:

[ChromeDriver](#), [EventFiringWebDriver](#), [FirefoxDriver](#), [HtmlUnitDriver](#), [InternetExplorerDriver](#), [PhantomJS WebDriver](#), [RemoteWebDriver](#), [SafariDriver](#)

# Selenium의 개요

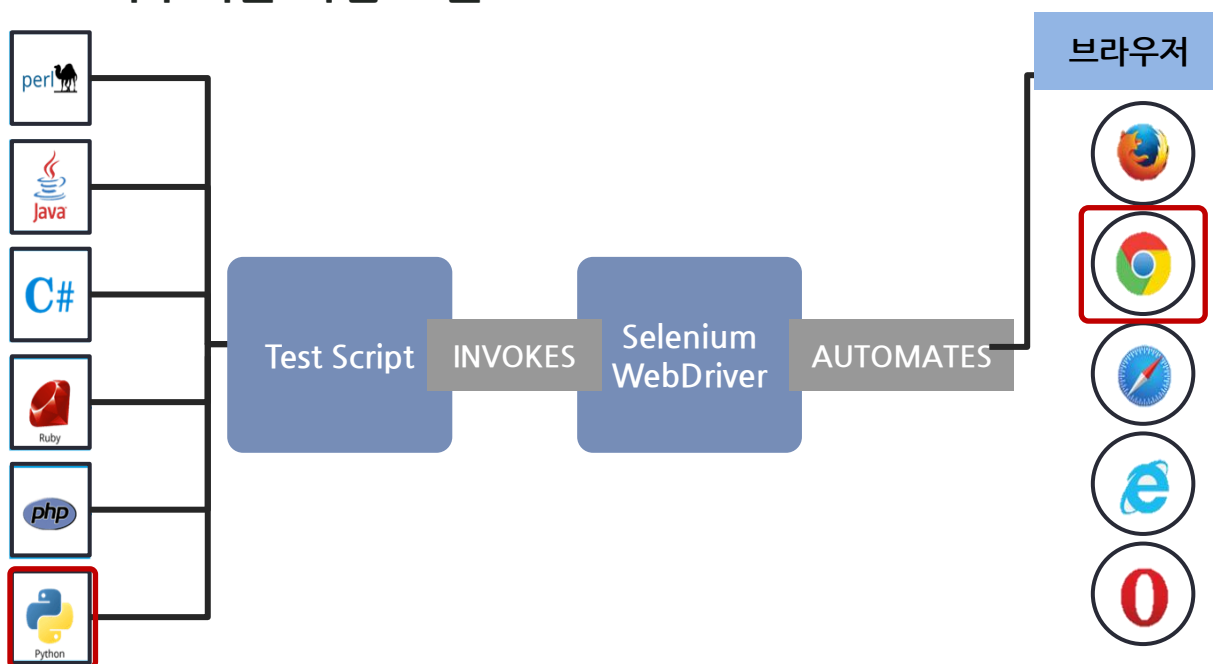
## 2 Selenium 개발 환경 구축

### 1 Selenium

#### WebDriver API

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

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





# Selenium의 개요



## 2 Selenium 개발 환경 구축

### 2 Selenium 설치

- 1 cmd 창에서 pip 명령 또는 conda 명령을 통해 설치 가능

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

- 2 아나콘다를 설치했으므로 conda 명령으로 설치

```
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: |
```

# Selenium의 개요

## 2 Selenium 개발 환경 구축

### 2 Selenium 설치

#### 3 conda 명령으로 selenium 설치(cmd 창 또는 아나콘다 프롬프트 창에서)

```
C:\Users\Samsung>conda install selenium
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Samsung\Anaconda3

added / updated specs:
- selenium

The following NEW packages will be INSTALLED:

selenium          pkgs/main/win-64::selenium-3

Proceed ([y]/n)? y

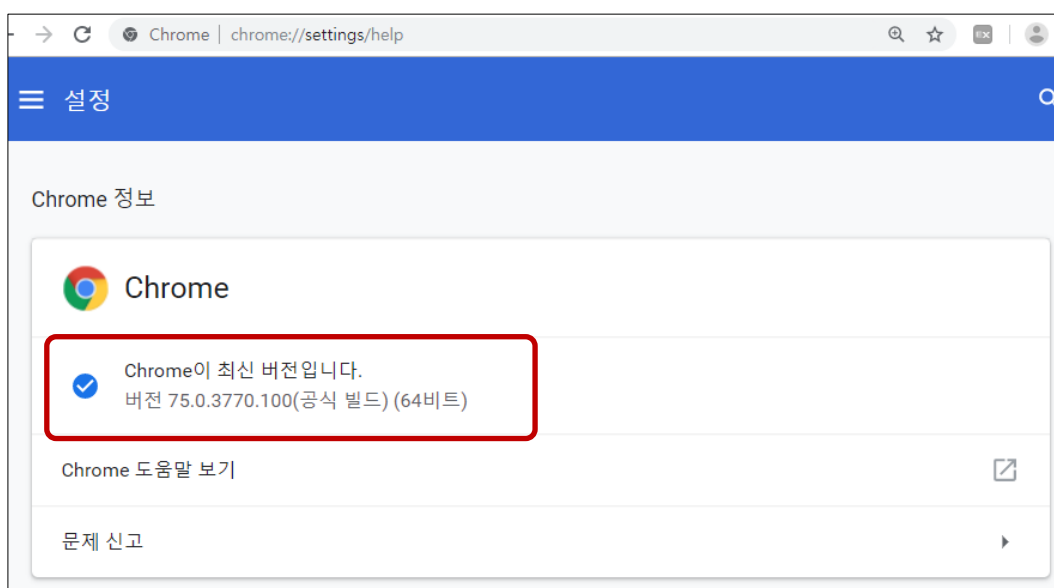
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

# Selenium의 개요

## 2 Selenium 개발 환경 구축

### 3 크롬 드라이버(Chrome Driver) 설치

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



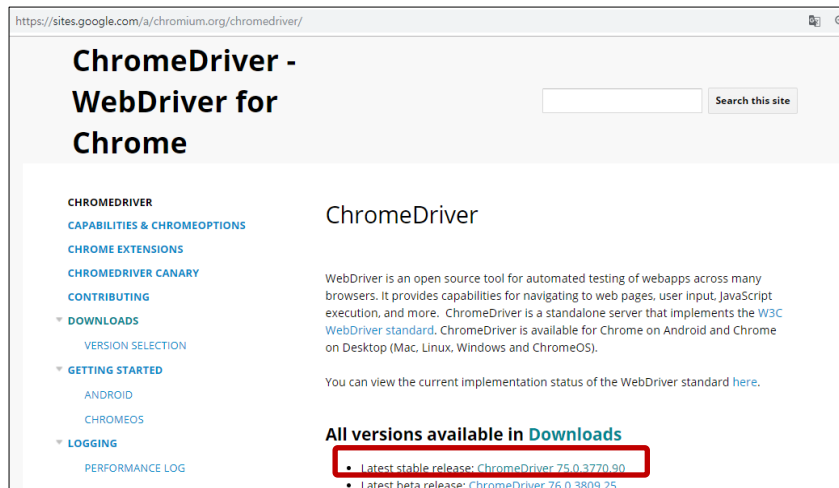
# Selenium의 개요

## 2 Selenium 개발 환경 구축

### 3 크롬 드라이버(Chrome Driver) 설치

- 2 다음 사이트에서 시스템에 설치된 크롬 브라우저와 동일 버전의 링크 클릭

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



# Selenium의 개요

## 2 Selenium 개발 환경 구축






### 3 크롬 드라이버(Chrome Driver) 설치

#### 3 chromedriver\_win32.zip을 다운로드

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

← → ↻ <https://chromedriver.storage.googleapis.com/index.html?path=75.0.3770.90/>

### Index of /75.0.3770.90/


	Name		Last modified	Size	ETag
	<a href="#">Parent Directory</a>			-	
	<a href="#">chromedriver_linux64.zip</a>	20	-06-13 21:21:15	4.90MB	5ceb8cdd79bba758f8975e72b3a28f0
	<a href="#">chromedriver_mac64.zip</a>	20	-06-13 21:21:16	6.77MB	e63bd10100c97a1f7ddaad4a15c586d3
	<a href="#">chromedriver_win32.zip</a>	20	-06-13 21:21:18	4.36MB	9bd10e443b5f98019fa4616fb29a9123
	<a href="#">notes.txt</a>	20	-06-13 21:33:08	0.00MB	200460edd48a8b7baef6cf318d290d6e

# Selenium의 개요



## 2 Selenium 개발 환경 구축

### 4 Selenium을 사용한 크롬 브라우저 제어 예제 테스트



```
#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("태그 객체 : ", type(target))
target.send_keys('파이썬')
target.send_keys(Keys.ENTER)
#driver.quit()
```

# Selenium의 개요

## 2 Selenium 개발 환경 구축

### 4 Selenium을 사용한 크롬 브라우저 제어 예제 테스트

#### ■ exam6\_1.py 실행 결과

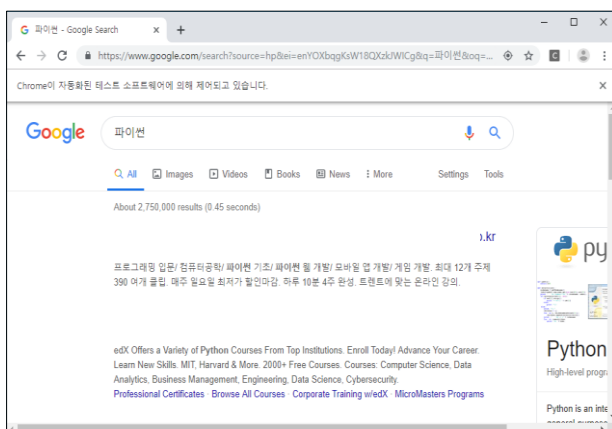
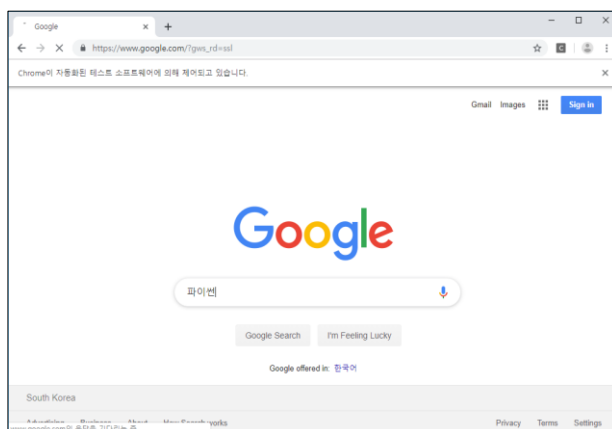
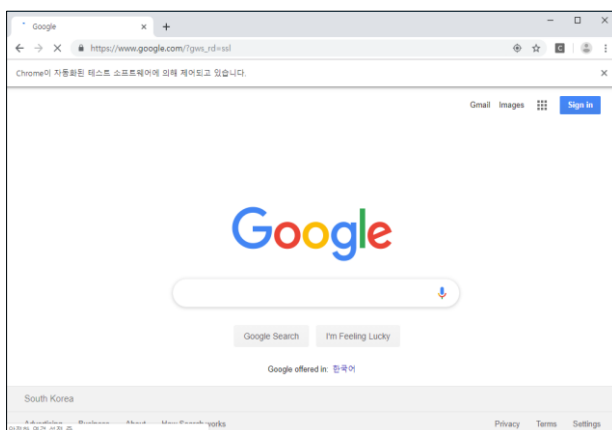
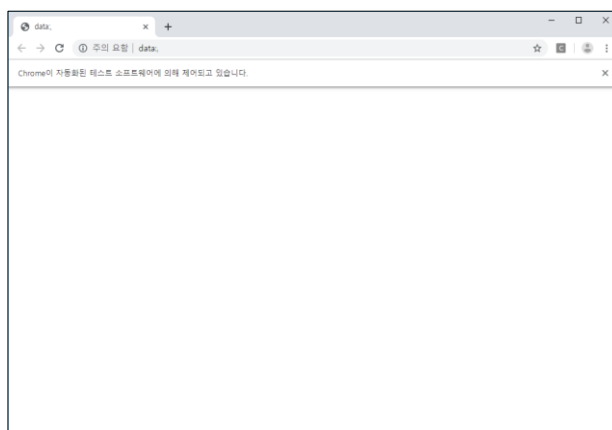
```
PS C:\example\myvscode> & C:/Users/Samsung/Anaconda3/python.exe c:/example/myvscode/unit6/exam6_1.py
```

```
DevTools listening on ws://127.0.0.1:13106/devtools/browser/d8621d5b-91d5-4bc6-8102-8a151b2d2b8c
```

```
KLIB_SelfTest return : KLR_OK
```

```
WebDriver 객체 : <class 'selenium.webdriver.chrome.webdriver.WebDriver'>
```

```
찾아온 태그 객체 : <class 'selenium.webdriver.remote.webelement.WebElement'>
```



# Selenium을 사용한 웹 페이지 스크래핑

## 1 Selenium API 소개

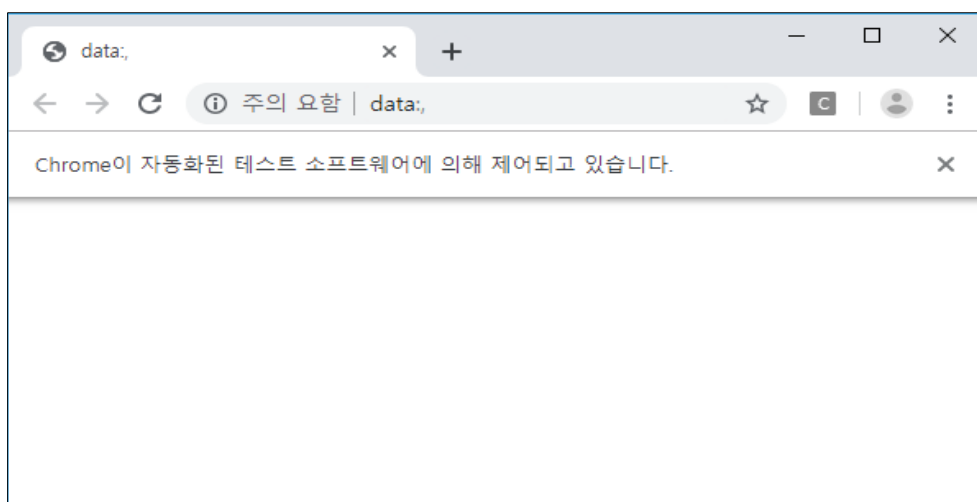
### WebDriver 객체 생성

- 1 다음 코드를 수행시켜서 크롬 드라이버를 기반으로 `selenium.webdriver.chrome.webdriver.WebDriver` 객체 생성

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

- 2 아규먼트로 `chromedriver.exe` 파일이 존재하는 디렉토리와 파일명에 대한 패스 정보 지정

➡ Selenium에 의해 관리되는 크롬 브라우저가 기동됨





# Selenium을 사용한 웹 페이지 스크래핑



## 2 메서드를 사용한 웹 페이지 파싱

### 1 페이지 가져오기

- 1 selenium.webdriver.chrome.webdriver.WebDriver 객체의 get() 메서드 사용

➡ 크롤링하려는 웹 페이지를 제어하는 크롬 브라우저에 로드하고 렌더링

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

### 2 WebDriver가 웹 페이지의 완전한 로드를 보장할 수 없음

- 경우에 따라 페이지 로드 완료 또는 시작 전에 WebDriver가 제어권을 반환할 수 있음
- 견고성을 확보하려면 Explicit & Implicit Waits를 사용하여 요소가 페이지에 존재할 때까지 기다려야 함

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

# Selenium을 사용한 웹 페이지 스크래핑



## 2 메서드를 사용한 웹 페이지 파싱

### 2 요소 찾기

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

# Selenium을 사용한 웹 페이지 스크래핑



## 2 메서드를 사용한 웹 페이지 파싱

### 2 요소 찾기

#### 1 태그의 id 속성 값으로 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')
```

#### 2 태그의 class 속성 값으로 찾기

```
target =  
driver.find_element_by_class_name('quickResultLst  
Con')  
또는  
target = driver.find_element(By.CLASS_NAME,  
"quickResultLstCon")
```

# Selenium을 사용한 웹 페이지 스크래핑



## 2 메서드를 사용한 웹 페이지 파싱

### 2 요소 찾기

#### 3 태그명으로 찾기

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

#### 4 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>
```

# Selenium을 사용한 웹 페이지 스크래핑



## 2 메서드를 사용한 웹 페이지 파싱

### 2 요소 찾기

#### 5 부분 링크 텍스트로 태그 찾기

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

#### 6 CSS 선택자로 태그 찾기

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

# Selenium을 사용한 웹 페이지 스크래핑



## 2 메서드를 사용한 웹 페이지 파싱

### 2 요소 찾기

#### 7 Xpath로 태그 찾기

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

#### 8 조건에 맞는 요소 한 개 찾기 : **WebElement 객체 리턴**

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

#### 9 조건에 맞는 모든 요소 찾기 : **list 객체 리턴**

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

# Selenium을 사용한 웹 페이지 스크래핑



## 2 메서드를 사용한 웹 페이지 파싱

### 3 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')
```

# Selenium을 사용한 웹 페이지 스크래핑



## 3 소스 분석

### 1 소스 분석(1)

```
#파일명 : exam6_2.py
import urllib.request
from bs4 import BeautifulSoup
#서버 접속
server =
urllib.request.urlopen("https://www.i*****ks.co.kr/store/
store_map.do")

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



#### ■ exam6\_2.py 실행 결과

```
PS C:\example\myvscode> & C:/Users/Samsung/Anaconda3/python.exe c:/example/myvscode
/unit6/exam6_2.py
[]
```



# Selenium을 사용한 웹 페이지 스크래핑

## 3 소스 분석

### 2 소스 분석(2)

```
#파일명 : exam6_3.py
from selenium import webdriver

driver = webdriver.Chrome('C:/Temp/chromedriver')
driver.implicitly_wait(3)
driver.get("https://www.i*****s.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()
```

#### ■ exam6\_3.py 실행 결과

```
PS C:\example\myvscode> & C:/Users/Samsung/Anaconda3/python.exe c:/example/myvscode/unit6/exam6_3.py

DevTools listening on ws://127.0.0.1:1271/devtools/browser/40fb9fbc-abf1-491f-892c-55dbea935fea

KLIB_SelfTest return : KLR_OK
<class 'selenium.webdriver.remote.webelement.WebElement'>
<class 'str'>
경기광주 DT
경기도 광주시
031-
리저브 매장 2번
```

## 학습정리

## 1. Selenium의 개요



- 정적 웹 페이지
  - 웹 서버에서 전송된 웹 페이지의 소스에서 화면에 렌더링된 내용을 모두 찾을 수 있는 경우
  - 주로 HTML과 CSS 기술만으로 페이지의 소스가 만들어짐
- 동적 웹 페이지
  - JavaScript 프로그래밍 언어로 브라우저에서 실행시킨 코드에 의해 웹 페이지의 내용이 렌더링 시 자동으로 생성되는 페이지
- Selenium
  - 주로 웹 앱을 테스트하는데 이용하는 브라우저 자동화 툴
  - WebDriver라는 API를 통해 운영체제에 설치된 크롬이나 파이어폭스 등의 브라우저를 기동시킴
  - 웹 페이지를 로드하고 제어

## 학습정리

## 2. Selenium을 사용한 웹 페이지 스크래핑



- webdriver.Chrome(드라이버 파일 패스)를 호출하여 크롬 브라우저에 대한 WebDriver 객체 생성
- Selenium에 의해 WebDriver 객체를 생성하는 동안 크롬 브라우저 기동
- WebDriver 객체의 get() 메서드를 사용하여 크롤링하려는 웹 페이지를 제어되는 크롬 브라우저에 로드
- WebDriver 객체와 WebElement 객체는 find\_element\_by\_xxx()와 find\_elements\_by\_xxx()를 제공하여 태그명, id 속성값, class 속성 값 등을 사용하여 찾고자 하는 태그에 대한 WebElement 객체를 추출하게 지원