동적 스크래핑(크롤링)

다음의 경우에는 웹 페이지의 내용이 동적으로 생성되는 경우로 지금까지의 방법으로 스크래핑 할 수 없다.

- 사용자의 선택과 같은 이벤트에 의해서 자바스크립트의 수행 결과로 컨텐츠를 생성한다.
- 페이지의 랜더링을 끝낸 후에 Ajax 기술을 이용하여 서버로 부터 컨텐트의 일부를 전송받아 동적으로 구성한다.

이러한 경우에는 Selenium 을 사용하면 제어되는 브라우저에 페이지를 랜더링 해놓고 랜더링된 결과에서 컨텐츠를 읽어올 수 있다. 뿐만 아니라 컨텐츠내에서 클릭이벤트를 발생할 수도 있으며 로그인과 같은 데이터를 입력하는 것도 가능하다.

[Selenium 서버 기동 과정]

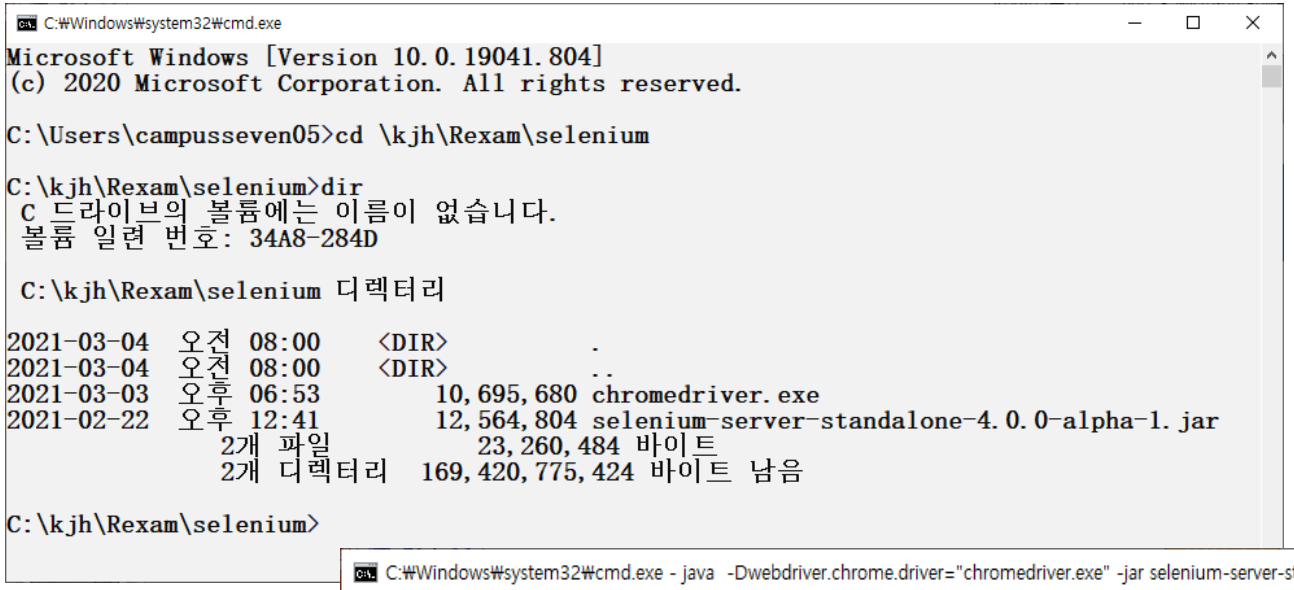
(1) chromedriver.exe, selenium-server-standalone-4.0.0-alpha-1.jar 를 네이버 MYBOX 에서 다운로드 하여 c:₩xxx₩Rexam₩selenium 폴더에 저장한다.

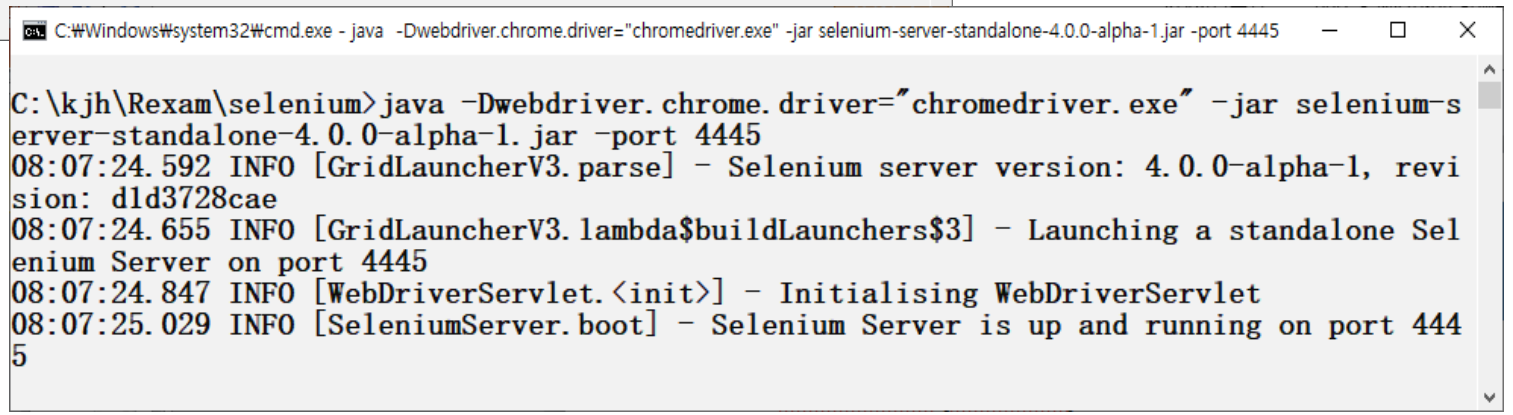
(2) cmd 창을 기동시키고 c:₩xxx₩Rexam₩selenium 로 옮겨가서 다음 명령을 수행시켜서 Selenium 서버를 기동시킨다.

네이버 클라우드에서 chromedriver.exe와 xxxx.jar 파일을 다운로드하여 C:/xxx/Rexam/selenium 폴더에 저장한다. cmd 창을 기동하고 C:/xxx/Rexam/selenium 폴더로 옮겨 가서 다음 명령을 실행한다. 만일 java 명령을 찾지 못한다는 오류가 발생한다면 환경 변수 설정에 문제가 있는 것이니 강사에게 꼭 알려서 해결해야 한다.

```
java -Dwebdriver.chrome.driver="chromedriver.exe" -jar selenium-server-standalone-4.0.0-alpha-1.jar -port 4445
```

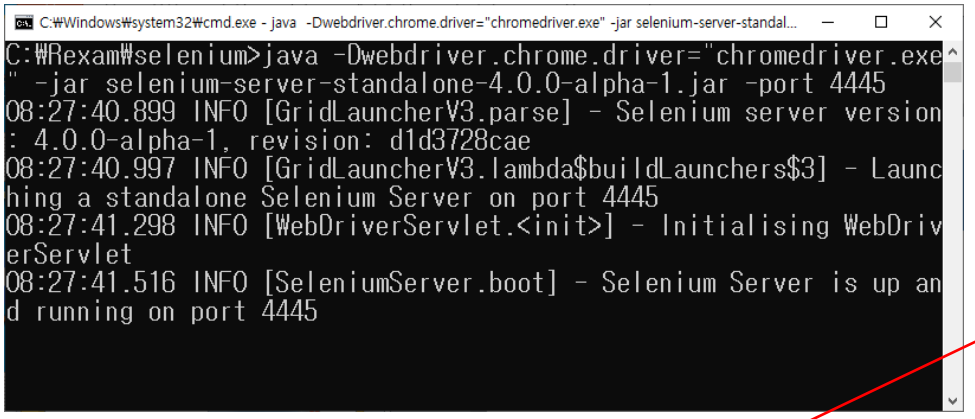
동적 스크래핑(크롤링)

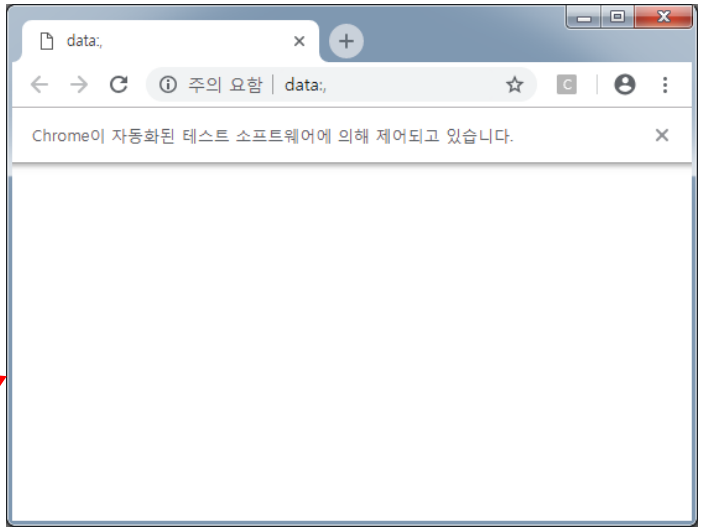

C:₩Windows₩system32₩cmd.exe
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Users\campusseven05>cd \kjh\Rexam\selenium
C:\kjh\Rexam\selenium>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 34A8-284D
C:\kjh\Rexam\selenium 디렉터리
2021-03-04 오전 08:00 <DIR> .
2021-03-04 오전 08:00 <DIR> ..
2021-03-03 오후 06:53 10,695,680 chromedriver.exe
2021-02-22 오후 12:41 12,564,804 selenium-server-standalone-4.0.0-alpha-1.jar
2개 파일 23,260,484 바이트
2개 디렉터리 169,420,775,424 바이트 남음
C:\kjh\Rexam\selenium>


C:₩Windows₩system32₩cmd.exe - java -Dwebdriver.chrome.driver="chromedriver.exe" -jar selenium-server-standalone-4.0.0-alpha-1.jar -port 4445
C:\kjh\Rexam\selenium>java -Dwebdriver.chrome.driver="chromedriver.exe" -jar selenium-server-standalone-4.0.0-alpha-1.jar -port 4445
08:07:24.592 INFO [GridLauncherV3.parse] - Selenium server version: 4.0.0-alpha-1, revision: d1d3728cae
08:07:24.655 INFO [GridLauncherV3.lambda$buildLaunchers$3] - Launching a standalone Selenium Server on port 4445
08:07:24.847 INFO [WebDriverServlet.<init>] - Initialising WebDriverServlet
08:07:25.029 INFO [SeleniumServer.boot] - Selenium Server is up and running on port 4445

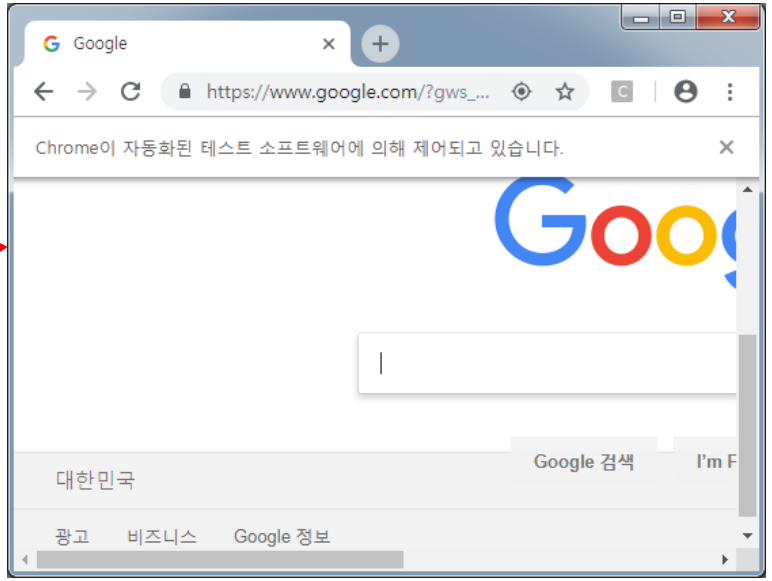
데이터 수집

동적 스크래핑(크롤링)



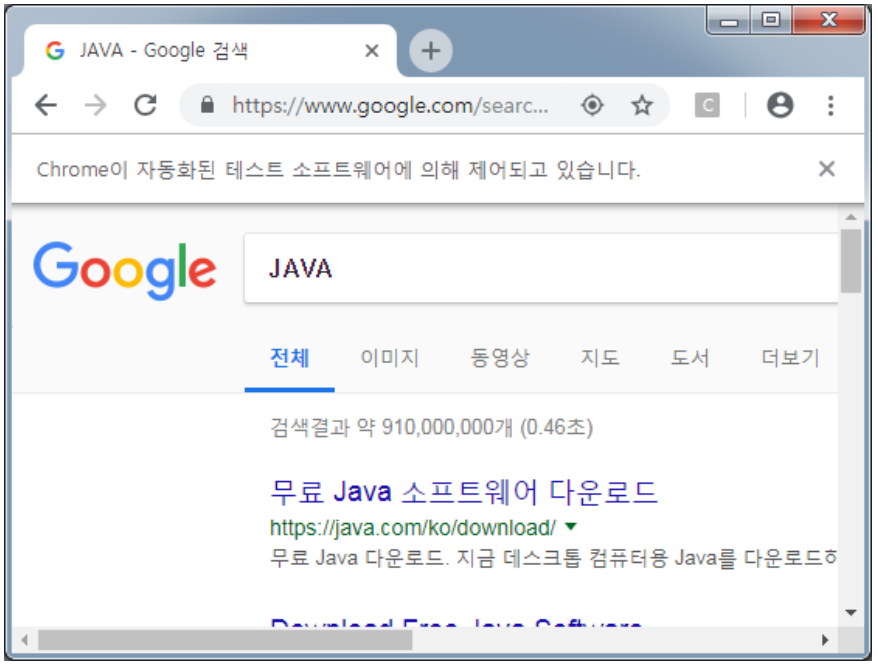



```
install.packages("RSelenium")
library(RSelenium)
remDr <- remoteDriver(remoteServerAddr = "localhost" ,
                      port = 4445, browserName = "chrome")
remDr$open()
remDr$navigate("http://www.google.com/")
```



동적 스크래핑(크롤링)

```
webElem <- remDr$findElement(using = "css selector", "[name = 'q']")
webElem$sendKeysToElement(list("JAVA", key = "enter"))
```



동적 스크래핑(크롤링)

[API 소개]

remDr <- remoteDriver(remoteServerAddr="localhost", port=4445,browserName="chrome")	R 코드로 Selenium 서버에 접속하고 remoteDriver 객체 리턴
remDr$open()	브라우저 오픈(크롬)
remDr$navigate(url)	url 에 해당하는 웹페이지 랜더링
one <- remDr$findElement(using='css selector','css선택자')	태그 한 개 찾기(webElement 객체) 태그가 없으면 NoSuchElement 오류 발생
one$getElementTagName()	찾아진 태그의 태그 명 추출(webElement 객체가 제공)
one$getElementText()	찾아진 태그의 태그 내용 추출(webElement 객체가 제공)
one$getElementAttribute("속성명")	찾아진 태그의 속성 명에 대한 값 추출 (webElement 객체가 제공)
one$clickElement()	찾아진 태그에서 클릭이벤트 발생시키기 (webElement 객체가 제공)

태그를 한 개만 찾는 함수로서 webElement 객체 리턴

동적 스크래핑(크롤링)

[API 소개]

doms <- remDr$findElements(using ="css selector", "컨텐트를추출하려는태그의 CSS선택자")	태그들을 찾기 존재하지 않으면 비어있는 리스트 리턴
sapply(doms,function(x){x$getElementText()})	찾아진 태그들의 컨텐트들의 추출하여 리스트로 리턴
sapply(doms, function(x){x$clickElement()})	찾아진 태그들에 각각 클릭 이벤트 발생
remDr$executeScript("arguments[0].click();",nextPageLink)	가끔 clickElement() 가 일을 안 할 때가 있음...ㅜ 이 때 사용하면 좋음
webElem <- remDr$findElement("css selector", "body") remDr$executeScript("scrollTo(0, document.body.scrollHeight)", args = list(webElem))	페이지를 아래로 내리는(스크롤) 효과

태그를 0개 이상 찾는 함수로서 webElement 객체들의 리스트 리턴

동적 스크래핑(크롤링)

```
remDr <- remoteDriver(remoteServerAddr = "localhost" , port = 4445, browserName = "chrome")
remDr$open()

remDr$navigate("http://www.google.com/")

webElem <- remDr$findElement(using = "css selector", "[name = 'q']")
webElem$sendKeysToElement(list("PYTHON", key = "enter"))


remDr$navigate("http://www.naver.com/")
str(remDr)

webElem <- remDr$findElement(using = "css selector", "#query")
webElem$sendKeysToElement(list("PYTHON", key = "enter"))
str(webElem)
```

Selenium 서버에 접속

Selenium 서버에 의해 제어되는 브라우저 기동

지정된 URL 페이지 를 요청하고 랜더링(자바스크립트코드 수행)

태그에 대한 DOM 객체 찾기

<input> 태그에 텍스트 입력하고 엔터키 입력을 자동화 하기

동적 스크래핑(크롤링)

```
url <- "http://unico2013.dothome.co.kr/crawling/tagstyle.html"
remDr$navigate(url)

# 단수형으로 노드 추출 - webElement 객체 리턴
one <- remDr$findElement(using='css selector','div') # 노드 한 개 리턴(webElement 객체)
one$getElementTagName()
one$getElementText()
one$getElementAttribute("style")

# 단수형으로 없는 노드 추출 – 오류 발생
one <- NULL
one <- remDr$findElement(using='css selector','p')
# 만일 오류 발생을 무시하고 싶어서 사용하지만 소용없음, 외부 라이브러리 사용 때문이라 추정(^^)
# 없을 수도 있으면 복수형(findElements()) 사용할것
one <- NULL
try(one<-remDr$findElement(using='css selector','p'))
```

동적 스크래핑(크롤링)

#복수형으로 노드 추출 – list 객체 리턴

```
more <- remDr$findElements(using='css selector','div')
sapply(more, function(x) x$getElementTagName())
sapply(more, function(x) x$getElementText())
```

#복수형으로 없는 추출 – 비어있는 list 객체 리턴

```
more <- remDr$findElements(using='css selector','p')
if(length(more) == 0)
  cat("<p> 태그는 없슈₩n")
```

동적 스크래핑(크롤링)

```
#이벤트 처리
url <- "http://unico2013.dothome.co.kr/crawling/exercise_bs.html"
remDr$navigate(url)

one<-remDr$findElement(using='css selector','a:nth-of-type(4)')
one$getElementTagName()
one$getElementText()
one$clickElement()


url <- "http://unico2013.dothome.co.kr/crawling/exercise_bs.html"
remDr$navigate(url)

one<-remDr$findElement(using='css selector','a:nth-of-type(3)')
one$getElementTagName()
one$getElementText()
remDr$executeScript("arguments[0].click();",list(one));
```