실전 웹 크롤링과 자동화 (For 신호용)

1권: 실전 크롤링 입문 - 원티드(Wanted) 정복하기

> 집필: 시니어 개발자 멘토, Gemini
> 최종 목표: 카카오 신입 공채 합격

Chapter 1-2: `Selenium` 첫걸음

Prologue: 왜 `Selenium`인가?

이전 챕터에서 우리는 `robots.txt`를 통해 크롤링의 규칙을 배웠습니다. 그렇다면 이제 바로 정보를 가져오면 될까요? 안타깝게도, 현대의 웹사이트는 그렇게 간단하지 않습니다.

과거의 웹사이트는 마치 책 페이지처럼, 서버로부터 받은 HTML 코드를 그대로 보여주기만 했습니다. 이런 '정적(Static) 페이지'는 `requests` 라이브러리만으로도 쉽게 내용을 가져올 수 있습니다.

하지만 원티드와 같은 최신 웹사이트들은 다릅니다. 처음에는 껍데기뿐인 HTML을 받아온 뒤, **JavaScript가 백그라운드에서 API 서버와 통신**하여 데이터를 가져와 화면을 동적으로 채워 넣습니다. 이것이 바로 '동적(Dynamic) 페이지'입니다.

`Selenium`은 바로 이 문제를 해결하기 위한 **'웹 브라우저 자동화 도구'**입니다. `Selenium`은 실제 사람이 하듯, 코드로서 웹 브라우저(Chrome 등)를 직접 실행시키고, JavaScript가 모든 데이터를 불러와 렌더링이 끝날 때까지 기다려줍니다. 그 덕분에 우리는 최종적으로 완성된 HTML 페이지의 정보를 온전히 얻을 수 있습니다.

---

1. 학습 목표 (Objectives)

- `Selenium`과 `WebDriver`의 관계를 설명할 수 있다.
- `webdriver-manager`를 사용하여 WebDriver를 자동으로 설정할 수 있다.
- `Selenium`으로 Chrome 브라우저를 열고, 특정 URL로 이동하는 코드를 작성할 수 있다.

---

2. 핵심 개념 (Core Concepts)

2.1 `Selenium`과 `WebDriver`

- **`Selenium`**: 웹 브라우저를 자동화하기 위한 다양한 명령어(API)를 제공하는 **파이썬 라이브러리**입니다. (예: "URL로 이동해!", "버튼을 클릭해!")
- **`WebDriver`**: `Selenium`의 명령을 실제로 수행하는 **'브라우저 조종 드라이버'**입니다. 각 브라우저(Chrome, Firefox 등)는 자신을 조종하기 위한 전용 드라이버를 가지고 있습니다. 즉, `Selenium`이 `WebDriver`에게 명령을 내리고, `WebDriver`가 브라우저를 움직이는 구조입니다.

과거에는 내 컴퓨터에 설치된 Chrome 브라우저의 버전에 맞는 `chromedriver.exe` 파일을 직접 다운로드하여 경로를 지정해줘야 하는 불편함이 있었습니다.

2.2 `webdriver-manager`: 귀찮은 설정은 이제 그만!

`webdriver-manager`는 이 불편한 과정을 자동화해주는 매우 유용한 라이브러리입니다.

> **[Developer's Tip]** `webdriver-manager`는 코드가 실행되는 시점에, **현재 내 컴퓨터에 설치된 Chrome 버전을 자동으로 확인**하고, 그 버전에 **정확히 맞는 WebDriver를 다운로드하여 설정**까지 알아서 해줍니다. 우리는 더 이상 브라우저가 업데이트될 때마다 드라이버를 새로 받을 필요가 없습니다.

---

3. 기초 실습 (Basic Practice)

이제 우리의 `job-crawler` 프로젝트에 첫 파이썬 파일을 만들고, `Selenium`으로 브라우저를 열어보는 실습을 시작하겠습니다.

1. **파일 생성:** `job-crawler` 폴더 안에 `main.py` 파일을 생성합니다.
2. **코드 작성:** 아래 코드를 `main.py`에 그대로 입력하고, 각 라인이 어떤 의미인지 주석을 통해 이해해 보세요. (이 부분을 손코딩하시면 됩니다.)

```
# main.py

import time

# Selenium의 핵심 모듈들을 임포트합니다.
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from webdriver_manager.chrome import ChromeDriverManager

# 1. WebDriver 설정
# ChromeDriverManager().install()은 현재 내 PC에 설치된 Chrome 버전에 맞는
# chromedriver를 자동으로 다운로드하여 그 경로를 반환합니다.
# 이 드라이버를 사용하여 ChromeService 객체를 생성합니다.
service = ChromeService(executable_path=ChromeDriverManager().install())

# 2. Chrome 브라우저 열기
# webdriver.Chrome에 위에서 만든 service를 인자로 전달하여 Chrome 브라우저를 실행합니다.
driver = webdriver.Chrome(service=service)

# 3. 특정 URL로 이동하기
# driver.get() 메서드를 사용하여 원하는 웹사이트로 이동합니다.
# 우리의 첫 번째 목표인 원티드의 '백엔드' 채용 공고 목록 페이지입니다.
URL = "https://www.wanted.co.kr/search?query=백엔드&tags=all"
driver.get(URL)

# 4. 페이지 로딩 대기
# 페이지의 모든 콘텐츠(특히 JavaScript로 동적으로 생성되는 부분)가
# 로드될 때까지 잠시 기다려줍니다. 여기서는 5초를 기다립니다.
time.sleep(5)
```

```
# 5. 브라우저 닫기
# 모든 작업이 끝나면 driver.quit()를 호출하여 브라우저를 종료합니다.
# 이걸 호출하지 않으면, 실행된 브라우저 창이 계속 남아있게 됩니다.
driver.quit()

print("Selenium 실행이 완료되었습니다.")
```

---

4. 프로젝트 적용 및 과제 (Assignment)

[Mission 2] 첫 번째 크롤러 실행

신호용 님의 두 번째 미션입니다.

> 1. `job-crawler` 프로젝트의 가상환경을 활성화하고, `main.py` 파일을 생성한 뒤 위 코드를 작성하십시오. (만약 `pip install selenium webdriver-manager`를 아직 안 하셨다면 먼저 실행해주세요.)
> 2. 터미널에서 `python main.py` 명령어를 실행하십시오.
> 3. **예상 결과:** 잠시 후 Chrome 브라우저 창이 **자동으로 열리고**, 원티드 백엔드 채용 공고 페이지로 **이동한 뒤**, 5초 후에 **자동으로 닫히는지** 확인하십시오. 터미널에는 "Selenium 실행이 완료되었습니다." 라는 메시지가 출력되어야 합니다.

이 미션이 성공했다면, 신호용 님은 웹 크롤링을 위한 가장 중요하고 첫 번째 관문을 통과한 것입니다. 우리는 이제 코드로서 웹 브라우저를 조종할 수 있게 되었습니다!