실전 웹 크롤링과 자동화 (For 신호용)

2권: 자동화 파이프라인 구축 - GitHub Actions와 Notion의 만남

Chapter 2-2: Python으로 Notion 다루기

Prologue: 코드 한 줄로 Notion을 움직이다

이전 챕터에서 우리는 Notion 페이지에 '채용 공고 아카이브'라는 데이터베이스를 만들고, 우리 프로그램이 접근할 수 있도록 '열쇠(API Key)'와 '주소(DB ID)'를 모두 준비했습니다.

이번 챕터에서는 `notion-client` 라는 편리한 파이썬 라이브러리를 사용하여, 이 열쇠와 주소를 가지고 Notion의 문을 열고 들어가는 방법을 배웁니다. 우리는 1권에서 완성했던 크롤링 코드를 수정하여, 수집된 채용 공고 데이터를 **코드 한 줄로 Notion 데이터베이스에 새로운 행(Row)으로 추가**하게 될 것입니다. 이제 더 이상 CSV 파일을 열어볼 필요가 없습니다. 모든 결과는 Notion에서 실시간으로 확인할 수 있게 됩니다.

---

1. 학습 목표 (Objectives)

- `notion-client` 라이브러리를 설치하고 초기화할 수 있다.
- Notion 데이터베이스의 속성(Property) 형식에 맞게 데이터를 구성할 수 있다.
- `pages.create()` 메서드를 사용하여 Notion 데이터베이스에 새로운 아이템을 추가하는 코드를 작성할 수 있다.

---

2. 핵심 개념 (Core Concepts)

2.1 `notion-client`: Notion을 위한 파이썬 리모컨

`notion-client`는 Notion API를 파이썬에서 매우 쉽고 직관적으로 사용할 수 있도록 만들어진 공식 라이브러리입니다. 복잡한 HTTP 요청을 직접 다룰 필요 없이, `notion.pages.create()` 와 같은 간단한 메서드 호출만으로 Notion을 제어할 수 있게 해줍니다.

2.2 Notion 데이터 구조: 모든 것은 '페이지'와 '블록'

Notion API를 이해하려면 Notion의 데이터 구조를 알아야 합니다.

- **페이지(Page):** Notion의 모든 것은 페이지입니다. 우리가 만든 '채용 공고 아카이브' 데이터베이스 자체도 하나의 페이지이고, 그 안에 들어가는 **각각의 채용 공고(하나의 행)** 역시 **'페이지'**입니다.
- **속성(Properties):** 데이터베이스 페이지 안에 있는 '직무', '회사명', '링크'와 같은 각 열(Column)을 의미합니다.
- **블록(Blocks):** 페이지의 실제 내용을 구성하는 요소들입니다. (예: 텍스트, 제목, 이미지 등)

우리의 목표는 **'채용 공고 아카이브' 데이터베이스 페이지 안에, 새로운 '채용 공고' 페이지를 생성하는 것**입니다.

2.3 API 데이터 형식: Notion만의 규칙

Notion API로 데이터를 보낼 때는, Notion이 정해놓은 고유한 JSON 형식을 따라야 합니다. 예를 들어, '직무'라는 제목(`Title`) 속성에 값을 넣으려면 아래와 같은 복잡한 구조로 데이터를 만들어 보내야 합니다.

```
{
  "properties": {
    "직무": {
      "title": [
        {
          "text": {
            "content": "백엔드 개발자"
          }
        }
      ]
    }
  }
}
```

처음에는 복잡해 보이지만, 각 속성 타입별로 정해진 규칙이 있으므로 익숙해지면 어렵지 않습니다.

---

3. 기초 실습 (Basic Practice)

이제 `main.py`를 대대적으로 수정하여, CSV 저장 로직을 Notion DB 저장 로직으로 교체하겠습니다.

[환경 설정]

```
# 공식 notion-client 라이브러리를 설치합니다.
pip install notion-client

# .env 파일 관리를 위해 python-dotenv를 설치합니다. (이전에 설치했다면 생략)
pip install python-dotenv
```

[.env 파일 설정] 프로젝트 루트 폴더(`main.py`와 같은 위치)에 `.env` 파일을 만들고, Mission 6에서 발급받은 키와 ID를 아래와 같이 저장합니다.

```
# .env
NOTION_API_KEY="secret_..."
NOTION_DATABASE_ID="..."
```

[코드 수정] `main.py` 전체를 아래 내용으로 업데이트합니다.

```python
# main.py

import time
import os
from dotenv import load_dotenv

from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from bs4 import BeautifulSoup

# notion-client 라이브러리를 임포트합니다.
import notion_client

# .env 파일에서 환경 변수를 로드합니다.
load_dotenv()

# --- 1. Notion API 설정 ---
NOTION_API_KEY = os.environ.get("NOTION_API_KEY")
NOTION_DATABASE_ID = os.environ.get("NOTION_DATABASE_ID")

# Notion 클라이언트를 초기화합니다.
notion = notion_client.Client(auth=NOTION_API_KEY)

# --- 2. Selenium 설정 및 실행 ---
service = ChromeService(executable_path=ChromeDriverManager().install())
driver = webdriver.Chrome(service=service)

URL = "https://www.wanted.co.kr/"
driver.get(URL)
time.sleep(2)

search_button = driver.find_element(By.CSS_SELECTOR, "button[data-attribute-
id='gnb']")
search_button.click()
time.sleep(1)

search_input = driver.find_element(By.CSS_SELECTOR,
"input.SearchInput_SearchInput__R6jwT")
search_input.send_keys("백엔드")
search_input.send_keys(Keys.ENTER)
time.sleep(2)

last_height = driver.execute_script("return document.body.scrollHeight")
while True:
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(2)
    new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        break
    last_height = new_height
```

```
print("페이지 스크롤 완료.")

# --- 3. 데이터 추출 및 Notion에 저장 ---
html = driver.page_source
soup = BeautifulSoup(html, "lxml")
job_cards = soup.select("div[role='listitem'] a")

print(f"총 {len(job_cards)}개의 채용 공고를 찾았습니다. Notion에 저장을 시작합니다...")

for card in job_cards:
    title = card.select_one("strong[class*='JobCard_title']").text
    company_name = card.select_one("span[class*='CompanyName']").text
    link = "https://www.wanted.co.kr" + card['href']

    # Notion 데이터베이스에 페이지를 생성합니다.
    try:
        notion.pages.create(
            parent={"database_id": NOTION_DATABASE_ID},
            properties={
                "직무": { # '직무'는 Notion DB의 제목(Title) 속성입니다.
                    "title": [
                        {
                            "text": {
                                "content": title
                            }
                        }
                    ]
                },
                "회사명": { # '회사명'은 Notion DB의 텍스트(Rich Text) 속성입니다.
                    "rich_text": [
                        {
                            "text": {
                                "content": company_name
                            }
                        }
                    ]
                },
                "링크": { # '링크'는 Notion DB의 URL 속성입니다.
                    "url": link
                }
            }
        )
        print(f"'{title}' 공고를 Notion에 성공적으로 저장했습니다.")
    except Exception as e:
        print(f"'{title}' 공고 저장 중 오류 발생: {e}")

    # 서버에 부담을 주지 않기 위해 각 요청 사이에 잠시 대기합니다.
    time.sleep(0.5)

driver.quit()
print("모든 작업이 완료되었습니다.")
```

[Mission 7] Notion에 첫 데이터 기록하기

신호용 님의 일곱 번째 미션입니다.

> 1. `notion-client`와 `python-dotenv` 라이브러리를 설치하십시오.
> 2. 프로젝트 폴더에 `.env` 파일을 만들고 Notion API Key와 Database ID를 저장하십시오.
> 3. `main.py` 파일을 위 코드로 업데이트한 뒤, 터미널에서 `python main.py`를 실행하십시오.
> 4. **예상 결과:**
>
> - 크롤링이 진행된 후, 터미널에 **"~ 공고를 Notion에 성공적으로 저장했습니다."** 라는 메시지가 순차적으로 출력되는지 확인합니다.
> - **Notion의 '채용 공고 아카이브' 데이터베이스 페이지**를 열었을 때, 수집된 채용 공고들이 **'직무', '회사명', '링크'** 열에 맞춰 새로운 행으로 착착 쌓이는 것을 직접 눈으로 확인합니다.

이 미션이 성공하면, 우리는 드디어 데이터를 내 컴퓨터가 아닌, 언제 어디서든 접근할 수 있는 클라우드 데이터베이스에 저장하는 데 성공한 것입니다! 성공 여부를 알려주세요.