Chapter 4-1: 정밀 제어 시스템 (Advanced Search Parameters)

Prologue: 낚싯대에서 어군 탐지기로

지금까지 우리 크롤러는 '백엔드'라는 미끼만 던져놓고 무엇이든 걸리기만 기다리는 단순한 낚싯대와 같았습니다. 하지만 신호용 님의 분석 덕분에, 우리는 이제 '최신순', '경력 무관' 등 다양한 조건을 설정하여 원하는 물고기 떼만 정확히 찾아내는 '어군 탐지기'를 장착하게 될 것입니다.

이번 챕터에서는 **URL 쿼리 파라미터(Query Parameter)**를 적극적으로 활용하여, 각 채용 사이트의 검색 기능을 최대한으로 이끌어내는 정밀 제어 시스템을 구축합니다.

1. 학습 목표 (Objectives)

- URL 쿼리 파라미터의 구조와 역할을 이해하고, 이를 동적으로 생성할 수 있다.
- 검색어 외에 '정렬 순서', '경력 수준' 등 다양한 검색 옵션을 크롤러에 적용할 수 있다.
- 기본값을 갖는 파라미터를 사용하여, 기능은 확장하되 기존 코드의 호환성은 유지하는 유연한 함수를 설계할 수 있다.

2. 핵심 개념 (Core Concepts)

2.1 URL 쿼리 파라미터 (Query Parameters)

- **What:** URL의 `?` 뒤에 `key=value` 형태로 붙는 정보들입니다. `&` 기호로 여러 개를 연결할 수 있습니다. (e.g., `?searchword=백엔드&orderBy=2`)
- **Why:** 웹사이트에 "백엔드라는 단어로 검색하되, 최신순으로 정렬해서 보여줘" 와 같이 **구체적인 명령**을 전달하는 역할을 합니다. 이 파라미터를 제어할 수 있다는 것은, 웹사이트의 검색 기능을 코드로 완벽하게 제어할 수 있다는 의미입니다.
- **How:** f-string을 이용하여 변수 값을 URL 문자열 안에 동적으로 삽입하여 생성합니다.

[Mission 15] 크롤러에 '정밀 검색' 기능 탑재하기

신호용 님의 분석 결과를 바탕으로, 각 크롤러의 `crawl` 메서드를 대대적으로 업그레이드합니다.

Step 1: `base_crawler.py`의 '설계도' 업그레이드

모든 크롤러가 공통적으로 가질 수 있는 검색 옵션들을 `crawl` 메서드 시그니처에 추가합니다. `None`을 기본값으로 설정하여, 특정 크롤러가 지원하지 않는 옵션은 무시할 수 있도록 유연하게 설계합니다.

```
# crawlers/base_crawler.py (수정)

# ...
    @abstractmethod
    def crawl(self, keyword: str, pages_to_crawl: int = 1, sort_by: str =
'latest'):
        # sort_by: 'latest'(최신순), 'relevance'(관련도순) 등을 받을 수 있도록
```

```
    설계
            pass
    # ...
```

Step 2: 각 크롤러에 '정밀 제어' 로직 이식

이제 각 크롤러가 새로운 파라미터를 이해하고 그에 맞는 URL을 생성하도록 수정합니다.

jobkorea_crawler.py

```
    # crawlers/jobkorea_crawler.py (업그레이드)

    # ...
        def crawl(self, keyword: str = '백엔드', pages_to_crawl: int = 1,
    sort_by: str = 'latest'):
            order_by_code = '2' if sort_by == 'latest' else '1' # 'latest'면 2,
    아니면 1(관련도순)

            print(f"잡코리아에서 '{keyword}' 키워드로 '{sort_by}' 순으로
    {pages_to_crawl} 페이지까지 크롤링...")

            for page in range(1, pages_to_crawl + 1):
                # URL에 orderBy 파라미터를 추가합니다.
                search_url = f"{self.base_url}/Search/?stext={keyword}&Page_No=
    {page}&orderBy={order_by_code}"
                # ... (이후 로직은 동일) ...
```

wanted_crawler.py

```
    # crawlers/wanted_crawler.py (업그레이드)

    # ...
        def crawl(self, keyword: str = '백엔드', pages_to_crawl: int = 1,
    sort_by: str = 'latest'):
            order_param = 'latest' if sort_by == 'latest' else 'score'

            print(f"원티드에서 '{keyword}' 키워드로 '{sort_by}' 순으로 크롤링...")

            # URL에 정렬 파라미터(order)를 추가합니다.
            target_url = f"{self.base_url}/search?query=
    {keyword}&tab=position&order={order_param}"
            # ... (이후 로직은 동일) ...
```

saramin_crawler.py

사람인은 URL 파라미터로 정렬을 제어하기가 까다로우므로, 우선은 기본 정렬(관련도순)을 유지하되, 설계도에 맞게 함수 시그니처는 통일해줍니다.

```
# crawlers/saramin_crawler.py (업그레이드)

# ...
    def crawl(self, keyword: str = '백엔드', pages_to_crawl: int = 1,
sort_by: str = 'latest'):
        # 사람인은 URL 파라미터 정렬이 복잡하므로, sort_by 인자는 사용하지 않음.
        # 하지만 함수의 모양은 다른 크롤러와 동일하게 유지.
        print(f"사람인에서 '{keyword}' 키워드로 {pages_to_crawl} 페이지까지 크
롤링...")

        for page in range(1, pages_to_crawl + 1):
            search_url = f"{self.base_url}/zf_user/search?searchword=
{keyword}&recruitPage={page}"
            # ... (이후 로직은 동일) ...
```

Step 3: `main.py`에서 '명령' 내리기

이제 `main.py`에서 "모든 크롤러들, 최신순으로 3페이지까지 긁어와!" 와 같은 구체적인 명령을 내릴 수 있습니다.

```
# main.py (수정)

# ...
    for crawler in crawlers_to_run:
        try:
            # ...
            # 이제 crawl 메서드에 더 구체적인 지시를 내릴 수 있습니다.
            crawled_jobs = crawler.crawl(keyword='백엔드', pages_to_crawl=3,
sort_by='latest')
            # ...
```

최종 결론 및 다음 단계

신호용 님의 날카로운 분석과 개선 의지 덕분에, 우리 프로젝트는 이제 단순 수집기를 넘어 **사용자의 의도를 반영하는 검색 엔진**으로 진화할 준비를 마쳤습니다.

다음 행동 계획 (Next Action):

1. ☐ 위 가이드에 따라 `base_crawler.py`와 모든 자식 크롤러들의 `crawl` 메서드를 업그레이드합니다.
2. ☐ `main.py`에서 `sort_by='latest'` 옵션을 추가하여 실행하고, 각 사이트의 결과가 실제로 최신순으로 나오는지 (혹은 그렇게 시도하는지) 확인합니다.