COMENTO

직무부트캠프

[부천대] 실제 현업 WEB 개발자와 함께 SW 포트폴리오 제작까지!

위승빈

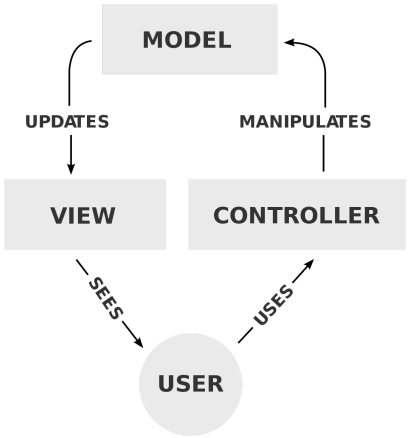
1. 웹 개발 디자인 패턴 학습

• MVC 패턴 조사

MVC란 소프트웨어 디자인 패턴 중 하나로, Model, View, Controller의 약자입니다. MVC 패턴을 이용하면 사용자 인터페이스로부터 비즈니스 로직을 분리하여 서로 영향없이 개발하기 수월해집니다.

Model은 어플리케이션의 정보를 나타내며, View는 사용자 인터페이스, Controller는 데이터와 비즈니스 로직 사이의 상호동작을 관리합니다.

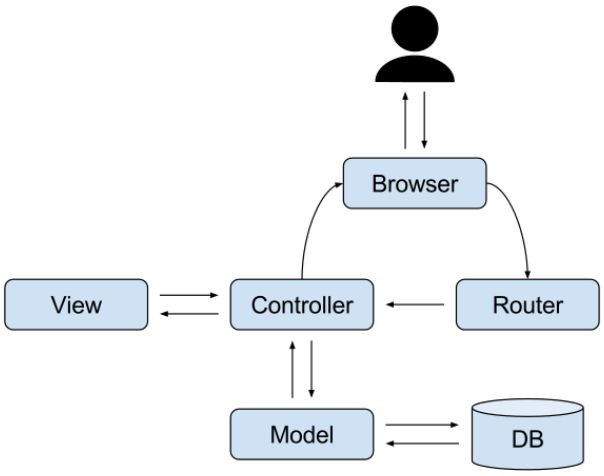
아래 사진은 MVC 패턴의 관계를 묘사하는 다이어그램입니다.



위와 같은 개념을 WEB에 적용하게 되면 다음과 같습니다.

1. User 웹사이트에 접속합니다 (USES)
2. Controller User 요청한 웹페이지를 호출하기 위해 Model을 호출합니다. (MANIPULATES)
3. Model은 DB나 파일과 같은 데이터 소스를 제어한 후 그 결과를 리턴합니다.
4. Controller는 Model이 리턴한 결과를 View에 반영합니다. (UPDATES)
5. 데이터가 반영된 View는 User에게 보여집니다. (SEES)

위 개념을 적용하여, 웹 애플리케이션에서 일반적인 다이어그램으로 나타내면 아래 사진과 같습니다.



Model, View, Controller를 더욱 자세히 알아보겠습니다.

Model은 쉽게 말해 데이터를 나타낸다고 생각하면 됩니다. 데이터베이스, 초기화값, 변수 등을 뜻합니다. 또한 이러한 데이터, 정보들의 가공을 책임지는 컴포넌트를 말합니다. Model은 다음과 같은 규칙을 가지고 있습니다.

- 사용자가 편집하길 원하는 모든 데이터를 가지고 있어야 합니다.
    - 즉 화면에서 표현되는 모든 데이터 정보를 가지고 있어야 합니다.
- View나 Controller에 대해 어떤 정보도 알지 않도록 해야 합니다.
    - 데이터 변경 시에 Model에서 View를 참조하는 내부 속성값을 가지면 안 됩니다.
- 변경이 일어나면 변경 통지에 대한 처리방법을 구현해야 합니다.
    - 즉 데이터 정보 변경 시에 이벤트를 발생시켜 누군가에게 전달해야 하며, 누군가 Model을 변경하도록 요청한다면 이를 수신할 수 있는 처리 방법을 구현해야 합니다.

View는 쉽게 말해 UI를 나타냅니다. 데이터 및 객체의 입력, 그를 보여주는 출력을 담당합니다. View는 아래와 같은 규칙을 가지고 있습니다.

- Model이 가지고 있는 정보를 따로 저장해서는 안 됩니다.
    - Model이 가지고 있는 정보를 전달받으면 그 정보를 유지하기 위해 임의의 View 내부에 저장하면 안 됩니다. 즉 화면에 표시만 하고 그 화면을 그릴 때 필요한 정보들은 저장하면 안 됩니다.

- Model이나 Controller와 같이 다른 구성 요소들을 몰라야 합니다.
    - 즉 View는 데이터를 받으면 화면에 표시해주는 역할만 가지게 됩니다.
- 변경이 일어나면 변경 통지에 대한 처리방법을 구현해야 합니다.
    - View에서는 화면에서 사용자가 화면에 표시된 내용을 변경하게 되면 이를 Model에게 전달하여 Model을 변경해야 합니다. 이 작업을 위해 변경 통지를 구현해야 합니다.

Controller는 데이터와 UI 요소들을 잇는 다리역할을 한다고 보면 됩니다. 다시 말해 사용자가 발생시킨 이벤트들을 처리하는 부분입니다. Controller는 아래와 같은 규칙을 가지고 있습니다.

- Model이나 VIew에 대해서 알고 있어야 한다.
    - Controller는 Model과 View를 중재하기 위해 Model과 그와 관련된 View에 대해서 알고 있어야 합니다.
- Model이나 View의 변경을 모니터링 해야 합니다.
    - Model이나 View의 변경 통지를 받으면 이를 해석해서 각각의 구성 요소에게 통지를 해야 합니다.

위와 같은 특성을 이용하면 아래와 같은 장점을 지니게 됩니다.

- Model, View, Controller 3가지로 구성되는 하나의 애플리케이션을 만들면 각각 맡은 바에 집중을 할 수 있게 되어 효율적으로 개발할 수 있게됩니다.
- 디자이너와 개발자의 협업에 용이하게 됩니다.
- 서로 분리되어 각자 역할에 집중할 수 있어, 유지보수성, 애플리케이션 확장성, 유연성이 증가하게 됩니다.
- 중복코딩의 문제점이 사라지게 됩니다.

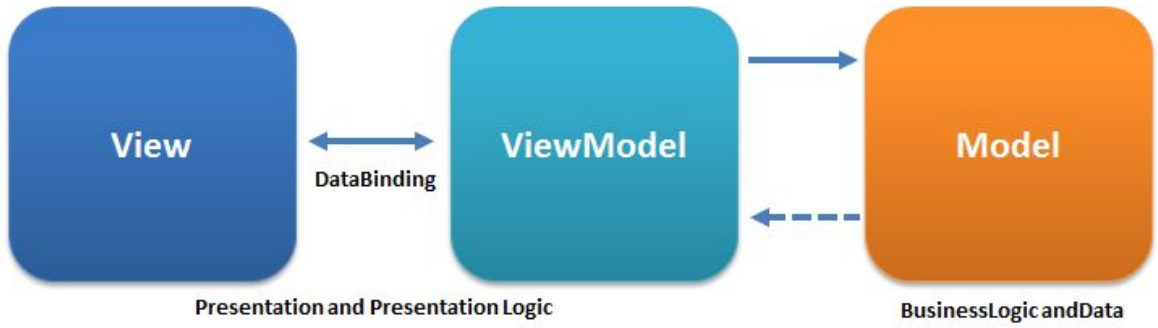
하지만 아래와 같은 단점이 있을 수 있습니다.

- 기본기능 설계를 위해 클래스가 많이 필요하기 때문에 복잡할 수 있습니다.
- 설계시간이 오래 걸리고 숙련된 개발자가 필요합니다.
- Model과 View의 의존성이 높아, 애플리케이션이 커질수록 복잡해지고 유지보수가 어려워 질 수 있습니다.

MVVM 패턴 조사

MVVM이란 소프트웨어 아키텍처 패턴 중 하나로, Model, View, ViewModel의 약자입니다. 마크업 언어 또는 GUI 코드로 구현하는 사용자 인터페이스의 개발을 비즈니스 로직 또는 백엔드 로직으로부터 분리시켜 View가 어느 특정한 모델 플랫폼에 종속되지 않도록 해줍니다.

MVVM의 ViewModel은 값 변환기라고 할 수 있는데, 이는 ViewModel이 Model에 있는 데이터 객체를 노출하는 책임을 지기 때문에, 객체를 관리하고 표현하기가 쉬워진다는 것을 의미합니다. 이런 점에서 ViewModel은 View 보다 더 모델이며, 모든 View들의 디스플레이 로직을 제외한 대부분을 처리합니다. 아래는 MVVM의 이러한 흐름을 나타낸 다이어그램 입니다.



위의 내용을 자세히 풀어보자면 다음과 같습니다.

1. 클라이언트의 Action들은 View를 통해 들어옵니다.
2. View에 액션이 들어오면 Command 패턴으로 ViewModel에 Action을 전달합니다.
3. ViewModel은 Model에게 데이터를 요청합니다.
4. Model은 ViewModel에게 요청받은 데이터를 응답합니다.
5. ViewModel은 응답받은 데이터를 가공하여 저장합니다.
6. View는 ViewModel과 데이터 바인딩하여 화면을 나타냅니다.

Model, View, ViewModel을 더욱 자세히 알아보겠습니다.

Model은 MVC 패턴과 같이 데이터를 나타낸다고 생각하면 됩니다. 비즈니스 로직과 유효성 검사, 데이터를 포함하는 애플리케이션의 도메인 모델로 생각할 수 있습니다.

View는 UI와 관련된 것들을 다룹니다. 사용자가 화면을 통해 보는 것들에 대한 구조, 레이아웃, 형태를 정의하며 View는 UI 로직을 제외하여 비즈니스 로직을 포함하지 말아야 합니다.

ViewModel은 View가 사용할 메서드와 필드를 구현하고 View에게 상태 변화를 알립니다. ViewModel에서 제공하는 메서드와 필드가 UI에서 제공할 기능을 정의하지만 View에서 해당

기능을 어떻게 보여줄 것인지를 결정합니다. 또한 일반적으로 ViewModel과 View는 일대다 관계를 형성합니다.

위와 같은 특성을 이용하면 아래와 같은 장점을 지니게 됩니다.

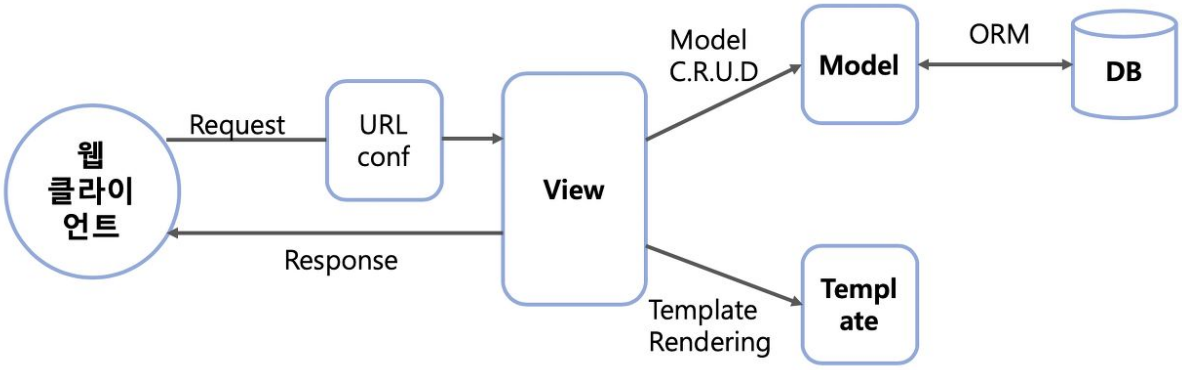
- ViewModel이 Model과 View 사이의 어댑터 역할로서 변경이 생겼을 때 변경을 최소화 할 수 있습니다.
- Model과 View 사이, ViewModel과 View 사이의 의존성이 없습니다.
    - ViewModel과 Model을 플랫폼 독립적으로 개발할 수 있습니다.
    - 유닛 테스트하기에 용이합니다.
- 개발 기간 동안 개발자와 디자이너가 동시에 작업이 가능해지게 됩니다.
- 중복코드를 모듈화 할 수 있습니다.

하지만 아래와 같은 단점이 있을 수 있습니다.

- 소형 애플리케이션에서 사용하게 되면 오버헤드가 커지게 됩니다.
- 애플리케이션이 너무 거대해지게 되면 애플리케이션의 메모리 소모가 데이터 바인딩 때문에 커지게 됩니다.
- ViewModel은 설계하기 어려운 문제가 있습니다.

- Django - models, views, urls, templates

Django에서는 View를 Template, Controller를 View라고 표현합니다. 즉 Django에서는 Model, View, Template의 약자인 MVT 패턴을 따르는데, MVT 패턴은 MVC 패턴과 동일한 작동을 한다고 보시면 됩니다. 다시말해 Model은 데이터, View는 애플리케이션 로직 처리를 하며, Template는 사용자 인터페이스를 나타냅니다. 아래 사진은 Django의 MVT 패턴을 나타낸 다이어그램 입니다.



위의 내용을 자세히 살펴보자면 다음과 같습니다.

1. 클라이언트로부터 요청을 받으면 URLconf를 이용하여 URL을 분석합니다.
2. URL 분석 결과를 통해 해당 URL에 대한 처리를 담당할 View를 결정합니다.
3. View는 자신의 로직을 실행하면서 데이터 베이스 처리가 필요하다면 Model을 통해 처리하고 그 결과를 반환합니다.
4. View는 자신의 로직 처리가 끝나면 Template를 사용하여 클라이언트에게 전송할 HTML 파일을 생성합니다.
5. View는 최종 결과로 HTML 파일을 클라이언트에게 보내 응답하게 됩니다.

Model, View, Template에 대해 자세히 알아보겠습니다.

Model은 데이터를 나타내며, 데이터에 관한 정의가 class 기반으로 이루어집니다. 또한 ORM을 통해 모델링을 하게 되면 어떤 종류의 데이터 베이스가 와도 Django와 호환되는 데이터 베이스라면 ORM이 자동으로 SQL문을 선언해 주기 때문에 구현에 대한 코드의 재사용도 가능합니다.

View는 애플리케이션에 맞는 로직을 처리합니다. 즉, 클라이언트로 부터 요청을 받아 데이터 베이스 접속 등 해당하는 로직을 처리하고 데이터를 HTML 템플릿 처리 후 반환하는 부분입니다. 함수나 class 방식으로 구현할 수 있습니다.

Template는 사용자 인터페이스로, Django가 View에서 로직 수행 후 HTML에 데이터를 붙여서 반환하게 되는데 여기서 HTML에 해당하는 부분이 바로 Template입니다. 기존의 HTML 문법 뿐만 아니라 Django Template 문법이 적용될 수 있으며 Template는 해당하는 파일들을 적당한 디렉토리에 위치 시키는 게 중요합니다. 그 경로는 settings.py에 정해줄 수 있습니다.

또한 Django의 URL에 대해서 알아보겠습니다.

URLConf란 URL의 패턴이 정의되는 부분이며, 클라이언트로부터 요청을 받게되면 가장 먼저 URLConf를 통해 URL 패턴을 분석하게 됩니다. Django에서는 복잡한 URL까지 쉽게 처리할 수 있을 정도로 URLConf가 잘 정의되어 있습니다.

URL이 분석되는 순서는 다음과 같습니다.

1. settings.py 파일의 ROOT_URLCONF 항목을 읽어 최상위 URLConf(urls.py)의 위치를 확인합니다.
2. URLConf를 로딩하여 urlpatterns 변수에 지정 되어있는 url 리스트를 검사합니다
3. 위에서 부터 순서대로 URL 리스트의 내용을 검사하면서 URL 패턴이 매치되면 검사를 종료합니다.
4. 매치된 URL의 View를 호출합니다. 여기서 View는 함수 또는 class 메소드가 됩니다. 호출 시에 HttpRequest 객체와 매칭할 때 추출된 단어들을 View에 인자로 넘겨줍니다.
5. URL 리스트를 끝까지 검사했는데 매칭이 안 된다면 error를 호출합니다.

URL 패턴에 정규표현식을 사용하면 URL을 좀 더 세밀하게 다룰 수 있습니다. 정규표현식에서 사용되는 문자의 의미는 아래 표와 같습니다.

표현	정의
.	모든 문자 하나
^	문자열의 시작
$	문자열의 끝
[ ]	[ ] 괄호에 있는 문자 하나
[^]	[ ] 괄호에 있는 문자 이외의 문자 하나
*	0번 이상의 반복
+	1번 이상의 반복
?	0번 또는 1번 반복
{n}	n번 반복

{m, n}	최소 m번에서 최대 n번까지 반복
A|B	A 또는 B
[a-z]	a에서 z까지 임의의 문자 하나
\w	영문, 숫자 또는 밑줄(_) 한개 ([0-9a-zA-Z] 와 동일)
\d	숫자 한개 ([0-9] 와 동일)

MVT 패턴의 장단점은 MVC 패턴의 장단점과 동일하다고 보시면 됩니다.

RESTful API, Django REST FRAMEWORK

REST란 Representation State Transfer의 약자로, 월드 와이드 웹과 같은 분산 하이퍼미디어 시스템을 위한 소프트웨어 아키텍처의 한 형식입니다. 다시말해 자원을 이름으로 구분하여 해당 자원의 상태를 주고 받는 모든 것을 의미합니다.

REST API란 웹에 존재하는 모든 자원에 고유한 URI를 부여하여 활용 가능한 API를 제공하는 것으로, 백엔드 개발자와 프론트엔드, iOS, Android 개발자와의 협업을 위해 데이터 정보 교환이 가능합니다.

RESTful이란 일반적으로 REST라는 아키텍처를 구현하는 웹 서비스를 나타내기 위해 사용되는 용어입니다. 즉, REST API를 제공하는 웹 서비스를 RESTful 하다고 할 수 있습니다. 다시말해 REST 원리를 따르는 시스템은 RESTful이란 용어로 지칭됩니다. RESTful은 REST를 REST답게 쓰기 위한 방법으로, 공식적으로 발표된 것은 아닙니다. 이로인해 명확한 정의는 없다고 하지만 RESTful의 목적은 이해하기 쉽고 사용하기 쉬운 REST API를 만드는 것입니다.

RESTful API 설계를 위해서는 URI는 정보의 자원을 표현해야 하며 리소스명은 동사보다는 명사를 사용하는 게 좋습니다. 자원에 대한 행위는 HTTP Method로 표현하여 GET, POST, PUT, DELETE 등으로 나타냅니다.

Django REST Framework는 약자를 사용하여 DRF라고 칭하기도 합니다. 그럼 DRF의 핵심 기능을 살펴보겠습니다.

- Request 객체
    - DRF는 HttpRequest를 Request 객체로 확장하여 더 유연한 요청 파싱을 제공합니다.
    - 핵심 기능은 request.POST와 비슷하지만 웹 API에 더 유용한 request.data 속성입니다.
- Response 객체
    - Response 객체는 TemplateResponse 객체의 일종입니다.
    - 렌더링 되지 않은 컨텐츠를 가져오고 협상을 통해 클라이언트에게 반환할 컨텐츠 유형을 결정합니다.
- 상태 코드
    - DRF는 status 모듈의 HTTP_400_BAD_REQUEST와 같이 각 상태 코드에 대해 보다 명시적인 식별자를 제공합니다.
- Wrapping API Views
    - DRF는 API View를 작성하는 데 사용할 두 wrapper를 제공합니다.
        - FBV(Function Based View)에서 사용되는 @api_view 데코레이터

- CBV(Class Based View)에서 사용되는 APIView 클래스

다음은 DRF의 사용 방법 입니다.

1. Rest Framework와 Swagger 설치
2. INSTALLED_APP에 REST API 관련 기능 추가
3. API 동작을 위한 Serializers 생성
4. 인증 권한 부여 방법 설정
5. settings.py에 기본 인증 방식, 토큰 방식, 서치 기능 추가
6. generic을 이용하여 View 작성
7. API View의 경로 설정
8. accounts 앱에 serializers.py 생성
9. serializer을 활용한 View 작성
10. View에 대한 경로 설정

- Beautifulsoup, Selenium 파이썬 웹 크롤링 조사 및 구현

Beautifulsoup은 HTML 및 XML 파일에서 원하는 데이터를 손쉽게 파싱할 수 있는 Python 라이브러리입니다.

아래는 BeautifulSoup을 이용한 크롤링 예제로, 다음의 미디어 뉴스 토픽을 가져왔습니다.

```
import requests
from bs4 import BeautifulSoup as bs

url = "https://www.daum.net/"
html_text = requests.get(url).text

soup_obj = bs(html_text, "html.parser")  # BeautifulSoup 객체 생성
# 첫번째 인자는 str 형식, html이나 xml로 작성된 문자열이어야 한다.
# 보통 requests나 urllib로 가져온 웹페이지 정보에 .text를 붙여서 넣어준다.
# 두번째 인자는 해석기(parser)를 넣어줘야 한다.

keywords = soup_obj.find_all('span', class_='txt_ranking')  # 데이터에서 태그와 클래스를 찾는 함수
keywords = [each_line.get_text().strip() for each_line in keywords]  # .get_text()를 이용하야 문자열만 추출, .strip()을 이용하여 양옆 공백제거

print(keywords)
```

Selenium은 웹 애플리케이션 테스트를 위한 포터블 프레임워크입니다. Selenium은 테스트 스크립트 언어를 학습할 필요 없이 기능 테스트를 만들기 위한 플레이백 도구를 제공합니다. Selenium Server와 Selenium Client가 있을 때, 로컬 컴퓨터의 웹 브라우저를 컨트롤하기 위해서는 Selenium Client를 사용합니다. Selenium Client는 Web Driver라는 공통 인터페이스와 각 브라우저 타입별로 하나씩 있는 Browser Driver로 구성되어 있습니다.

아래는 Selenium을 이용한 예제로, 웹드라이버를 이용해 구글 검색을 해봤습니다.

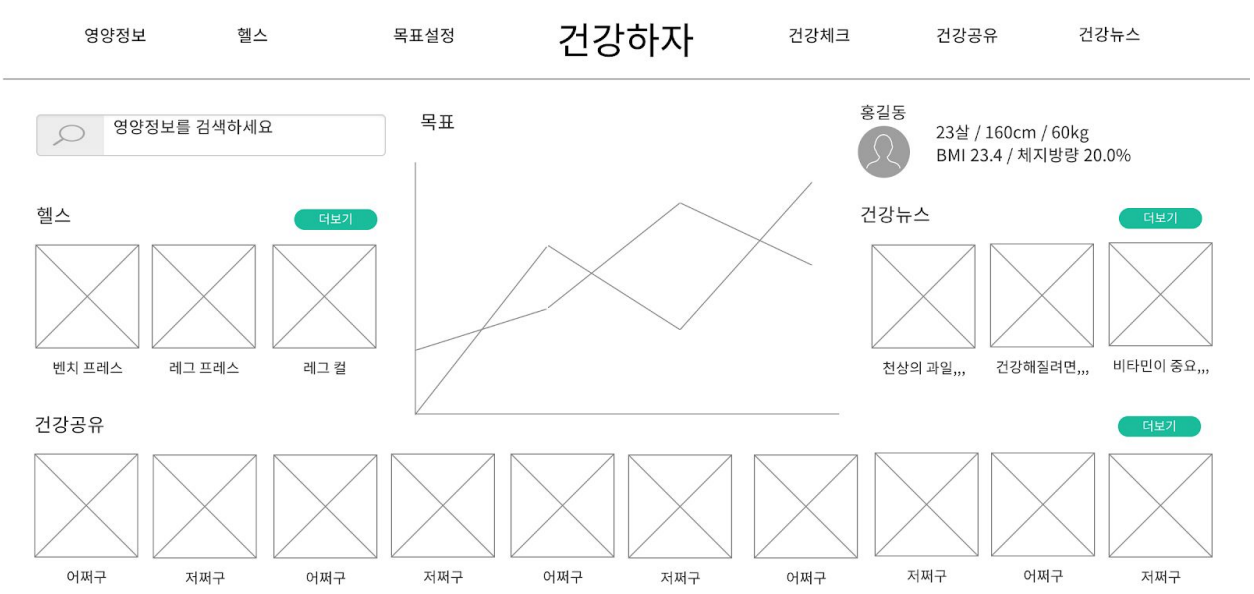
```
import time
from selenium import webdriver

driver = webdriver.Chrome('./chromedriver')  # 웹드라이버 실행 경로 지정, chromedriver는 폴더가 아니라 파일명
driver.get('https://www.google.co.kr/')

time.sleep(2)  # 2초간의 동작 확인
search_box = driver.find_element_by_name('q')  # element name이 q인 곳을 찾아서
search_box.send_keys('ChromeDriver')  # 키워드 입력
search_box.submit()  # 실행
time.sleep(2)  # 2초간의 동작 확인
driver.quit()  # 실행 후엔 웹드라이버 종료
```

아래는 Selenium과 BeautifulSoup을 응용한 예제로, 아카데미상(오사카상)의 리스트를 크롤링 하였습니다.

```
import time
import pandas as pd  # pandas는 dataframe을 주고 다루기 위한 라이브러리이다.

from selenium import webdriver
from bs4 import BeautifulSoup as bs

driver = webdriver.Chrome('./chromedriver')  # 웹드라이버 실행 경로 지정, chromedriver는 폴더가 아니라 파일명
driver.get('https://oscar.go.com/winners')
time.sleep(1)  # 1초간의 동작 확인

html = driver.page_source  # 연결한 웹사이트의 소스 가져오기
soup = bs(html, 'html.parser')  # BeautifulSoup 객체 생성

category = soup.select('div.winners-list__info > a')  # BeautifulSoup 객체의 select()를 이용해 html 내에 필요한 부분 선택하기
movie = soup.select('div.winners-list__info > h3 > a')
content = soup.select('div.winners-list__info > p')

oscars_2020 = []
for item in zip(category, movie, content):  # zip()은 동일한 개수로 이루어진 자료형을 묶어주는 역할을 한다.
  oscars_2020.append({'카테고리' : item[0].text, '영화' : item[1].text, '수상내역' : item[2].text})

data = pd.DataFrame(oscars_2020)  # pandas의 DataFrame()을 이용하여 DataFrame 객체를 생성한다.
print(data)
data.to_csv('oscars_2020.csv')  # pandas의 DataFrame.to_csv()를 이용하여 DataFrame을 csv 파일로 내보냅니다.

driver.quit()  # 실행 후엔 웹드라이버 종료
```

2. 본인의 웹 페이지 디자인 및 설계

저는 건강과 다이어트 관련된 웹 애플리케이션을 만들고자 하였습니다.



주요 요구사항은 다음과 같습니다.

요구사항명	요구사항 상세
식품 검색	사용자는 식품을 검색, 검색된 식품의 이미지, 영양정보를 표기
식품 제조사 정보 검색	제조사별로 식품을 검색
운동 검색	사용자는 운동을 검색, 검색된 운동의 이미지, 주의점 등을 표기
변화 그래프 제공	사용자의 몸 상태에 따른 변화 그래프를 제공
건강 정보 제공	사용자의 몸 상태에 따른 건강 정보를 제공
게시물 관리	사용자는 게시물을 생성, 읽기, 수정, 삭제 가능
뉴스 제공	건강과 관련된 뉴스 제공