

# Django 08 Authentication System 01

# • INDEX

---

- Cookie & Session
  - HTTP
  - 쿠키
  - 세션
- Django Authentication System
- Custom User model
  - User model 대체하기
- Login
- Logout
- Template with Authentication data

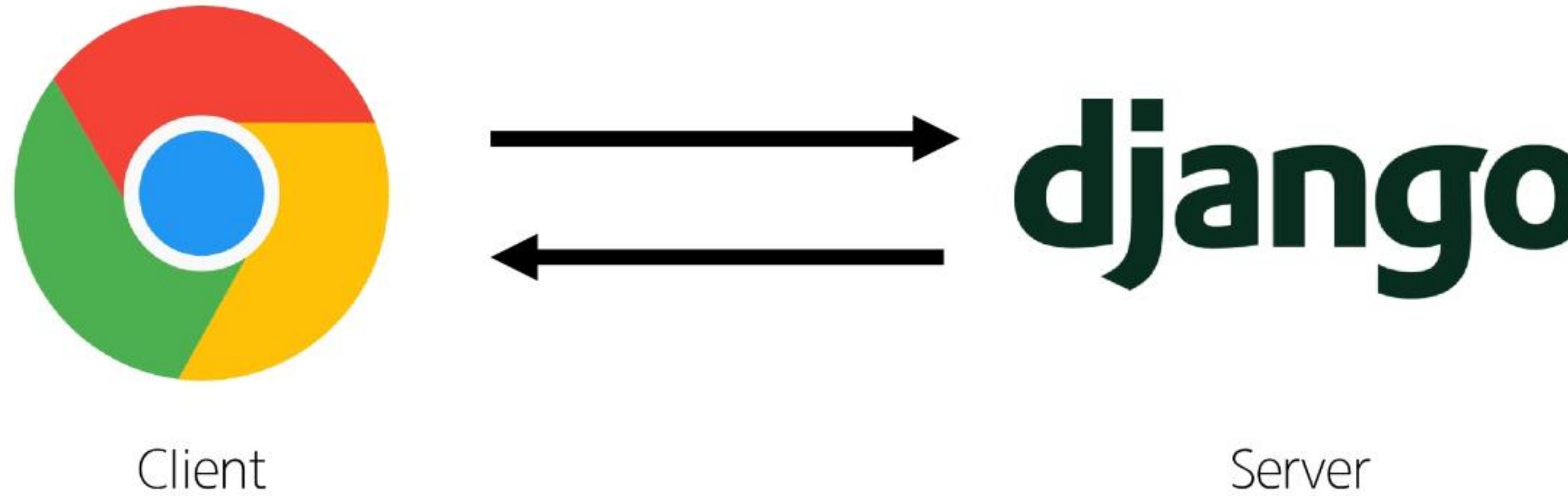
# • INDEX

---

- 참고
  - 쿠키의 수명
  - 쿠키와 보안
  - Django에서의 세션 관리
  - AuthenticationForm 내부 코드
  - AbstractUser class
  - User 모델 대체하기 Tip

# Cookie & Session

**HTTP**



**우리가 웹 페이지를 둘러볼 때  
우리는 서버와 서로 연결되어 있는 상태가 아니다.**

# HTTP

---

HTML 문서와 같은 리소스들을 가져올 수 있도록 해주는 규약  
웹(WWW)에서 이루어지는 모든 데이터 교환의 기초

## HTTP 특징

1. 비 연결 지향(connectionless)
  - 서버는 요청에 대한 응답을 보낸 후 연결을 끊음
2. 무상태(stateless)
  - 연결을 끊는 순간 클라이언트와 서버 간의 통신이 끝나며 상태 정보가 유지되지 않음

## 상태가 없다는 것은..

- 장바구니에 담은 상품을 유지할 수 없음
- 로그인 상태를 유지할 수 없음

쿠키

# 쿠키(Cookie)

---

서버가 사용자의 웹 브라우저에 전송하는 작은 데이터 조각

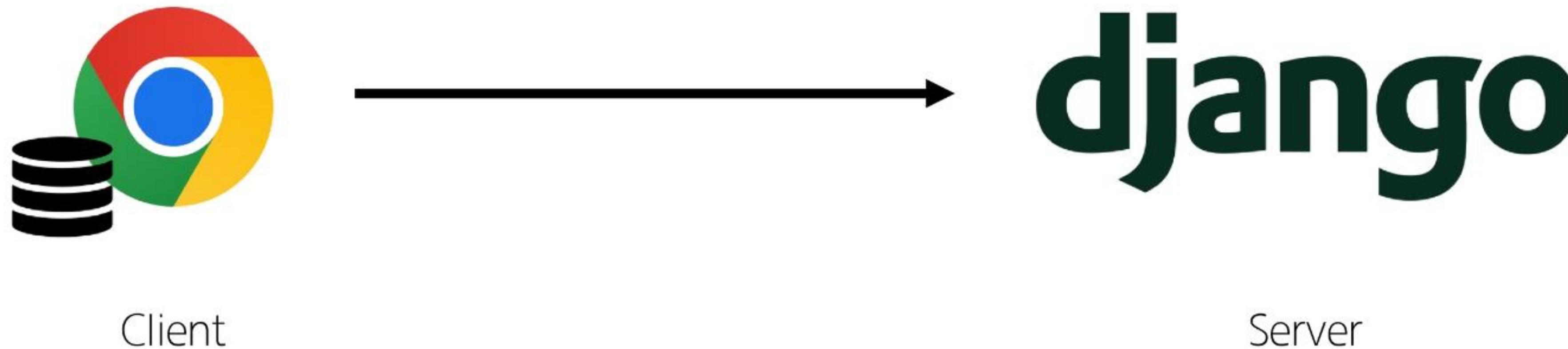
# 쿠키(Cookie)

-----  
서버가 사용자의 웹 브라우저에 전송하는 작은 데이터 조각

- 서버가 제공하여 클라이언트 측에서 저장되는 작은 데이터 파일
- 사용자 인증, 추적, 상태 유지 등에 사용되는 데이터 저장 방식

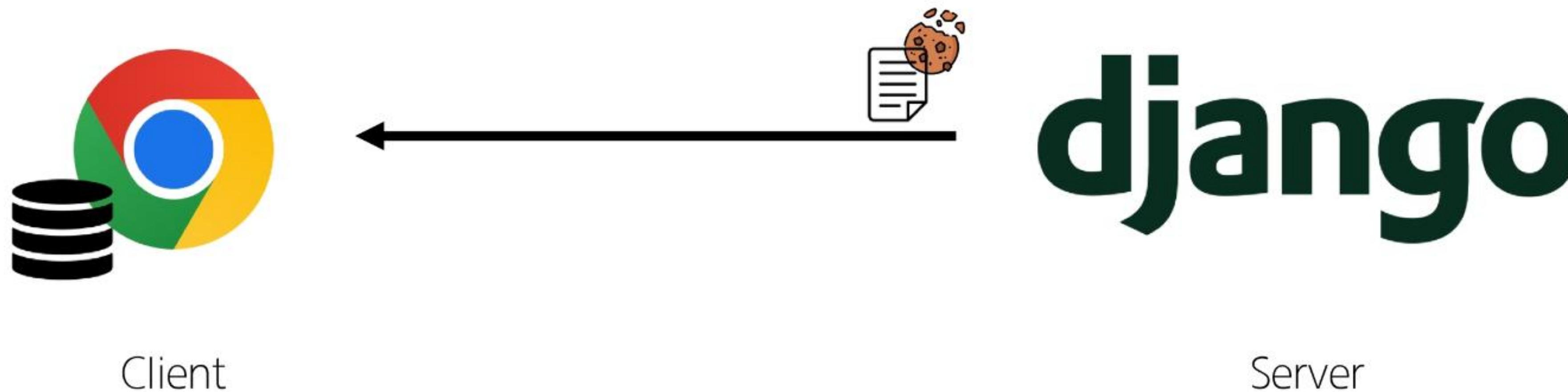
## 쿠키 동작 예시 (1/6)

1. 브라우저가 웹 서버에 웹 페이지를 요청



## 쿠키 동작 예시 (2/6)

2. 웹 서버는 요청된 페이지와 함께 쿠키를 포함한 응답을 브라우저에게 전송



## 쿠키 동작 예시 (3/6)

3. 브라우저는 받은 쿠키를 저장소에 저장  
쿠키의 속성(만료 시간, 도메인, 주소 등)도 함께 저장됨



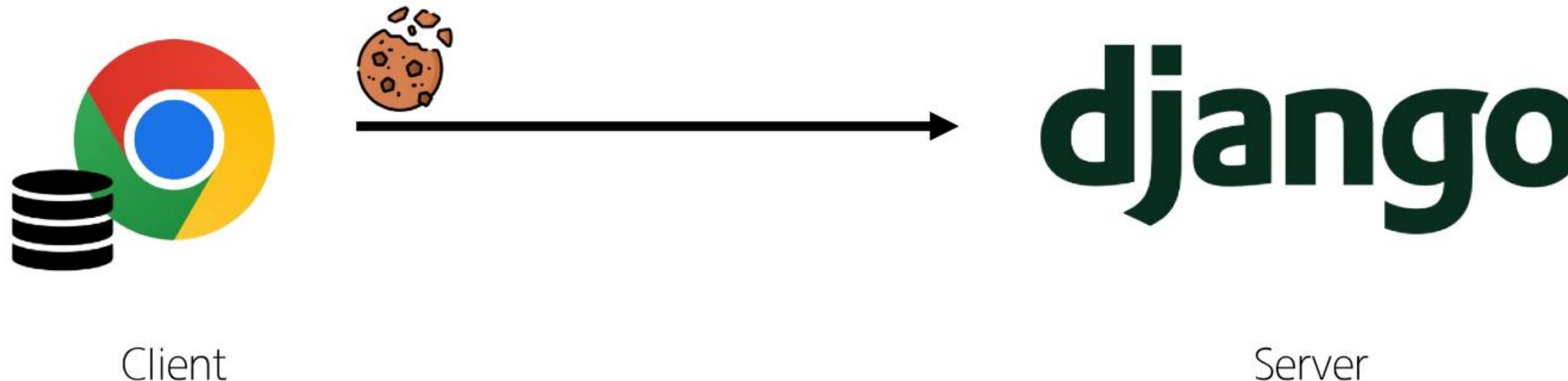
Client

**django**

Server

## 쿠키 동작 예시 (4/6)

4. 이후 브라우저가 같은 웹 서버에 웹 페이지를 요청할 때,  
저장된 쿠키 중 해당 요청에 적용 가능한 쿠키를 포함하여 함께 전송



## 쿠키 동작 예시 (5/6)

5. 웹 서버는 받은 쿠키 정보를 확인하고, 필요에 따라 사용자 식별, 세션 관리 등을 수행



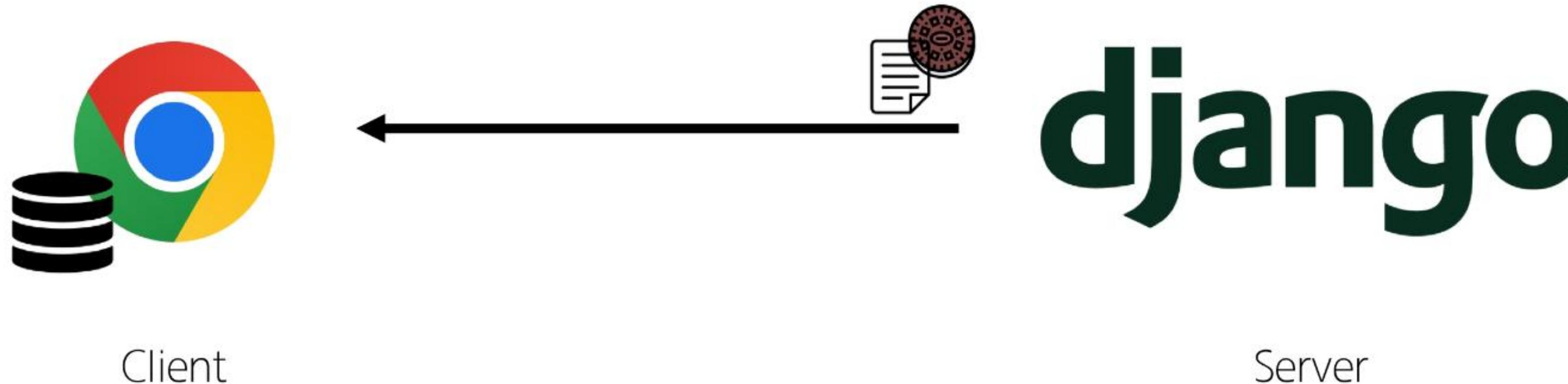
Client



Server

## 쿠키 동작 예시 (6/6)

6. 웹 서버는 요청에 대한 응답을 보내며, 필요한 경우 새로운 쿠키를 설정하거나 기존 쿠키를 수정할 수 있음



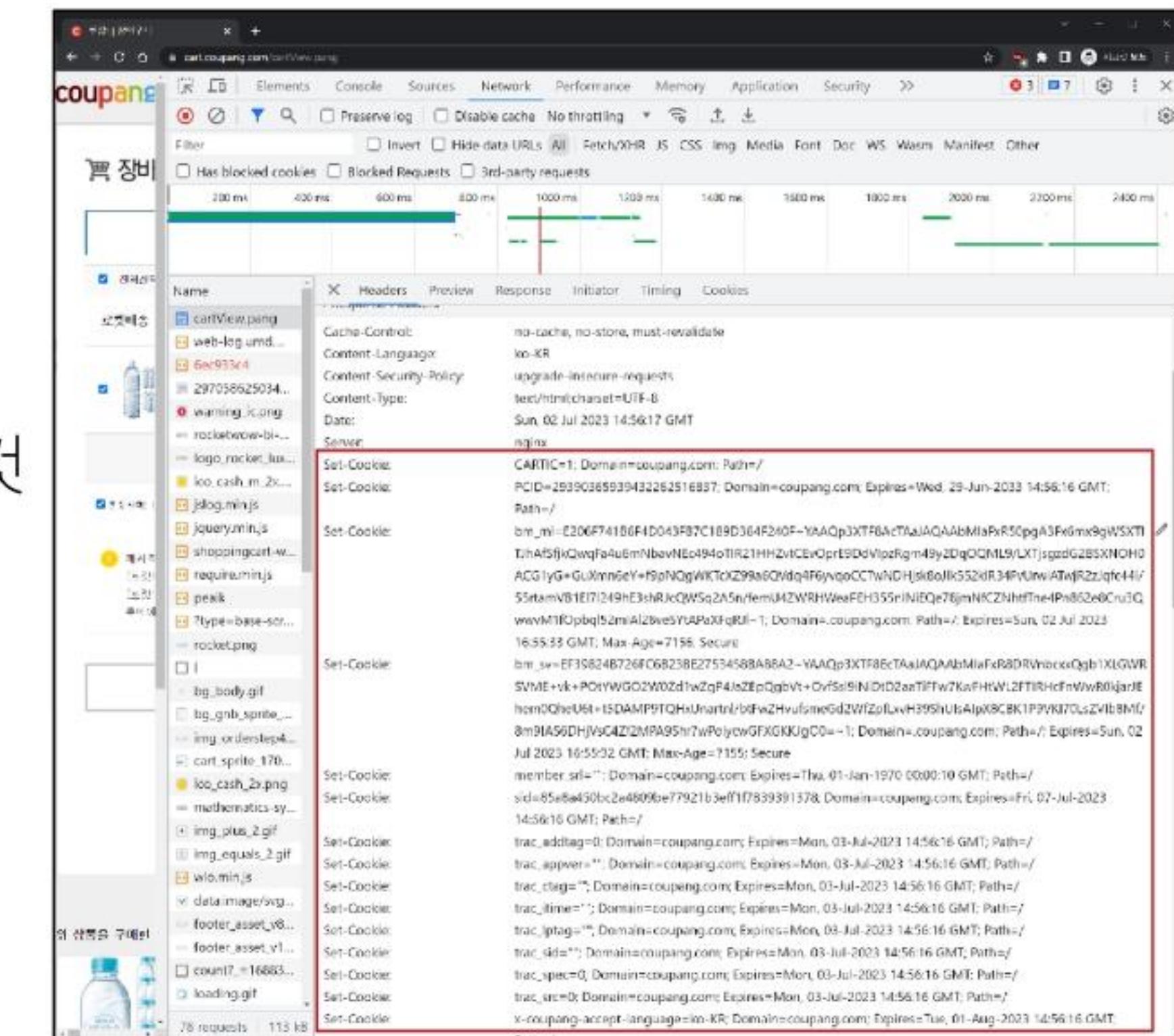
# 쿠키를 이용한 장바구니 예시 (1/5)

장바구니에 상품 담기

The screenshot shows the Coupang shopping cart interface. At the top, the Coupang logo is visible. Below it, the title "장바구니" (Cart) is displayed. To the right, a breadcrumb navigation shows the path: 01 옵션선택 > 02 장바구니 > 03 주문/결제 > 04 주문완료. The main area is divided into two tabs: "일반구매 (1)" (General Purchase (1)) and "정기배송 (0)" (Regular Delivery (0)). The "일반구매" tab is selected and highlighted with a blue border. Under this tab, there is a checkbox labeled "전체선택" (Select All) which is checked. To the right of the checkbox are columns for "상품정보" (Product Information), "상품금액" (Product Price), and "배송비" (Delivery Fee). Below this, a section for "로켓배송 상품" (Rocket Delivery Product) is shown, stating "무료배송 (19,800원 이상 구매가능)". A product item for "제주삼다수 그린 무라벨, 2L, 24개" is listed with an image of several water bottles. The quantity is set to 1, and the total price is 25,920원. The delivery option "내일(월) 7/3 도착 보장 (서울경기 기준)" is selected. The delivery fee is 25,920원, and the tip amount is 1,296원. The word "두료" is also present in the bottom right corner.

# 쿠키를 이용한 장바구니 예시 (2/5)

- 개발자 도구 - Network 탭 - cartView.pang 확인
- 서버는 응답과 함께 Set-Cookie 응답 헤더를  
브라우저에게 전송
  - 이 헤더는 클라이언트에게 쿠키를 저장하라고 전달하는 것



# 쿠키를 이용한 장바구니 예시 (3/5)

Cookie 데이터 자세히 확인

The screenshot shows the Network tab of the Chrome DevTools interface. The URL is `cart.coupaing.com/cart/view/pang`. The 'Cookies' tab is selected. The 'Request Cookies' section lists numerous cookies, many of which are highlighted in red. The 'Response Cookies' section also lists several cookies. The table columns include Name, Value, Do..., Path, Expl..., Size, Http..., Secu..., Sam..., Parti..., and Pr... . A red box highlights the 'Cookies' tab and the 'Request Cookies' table.

Name	Value	Do...	Path	Expl...	Size	Http...	Secu...	Sam...	Parti...	Pr...
sid	85a8a450bc2a4809be77921b3eff1f7...	cou...	/	202...	43					Med...
PCID	29390365939432262516837	cou...	/	202...	27					Med...
overrideAbTestGroup	%5B%5D	cou...	/	202...	25					Med...
MARKEID	29390365939432262516837	cou...	/	Sess...	31	✓	✓	None		Med...
x-coupaing-accept-langu...	ko-KR	cou...	/	202...	30					Med...
x-coupaing-target-market	KR	cou...	/	202...	25					Med...
ak_bmsc	70FCABD0E8BA297F554B63134A5D...	cou...	/	202...	475	✓				Med...
bm_sz	B868731896B8D4328FD60154E4438...	cou...	/	202...	358					Med...
searchKeyword	%EC%82%BC%EB%88%A4%EC%88%	cou...	/	202...	40					Med...
searchKeywordType	%7B%22%EC%82%BC%EB%88%A4%...	cou...	/	202...	60					Med...
FUN	"search":{ "reqUrl": "/search.pang", "is...	cou...	/	Sess...	58					Med...
bm_mi	E206F741B6F4D043FB7C189D364F2...	cou...	/	202...	372	✓				Med...
baby-isWide	small	cou...	/	202...	16					Med...
_abck	0C903C251C11E82BEEA3074C6DF8...	cou...	/	202...	485	✓				Med...
trac_src	0	cou...	/	202...	9					Med...
trac_spec	0	cou...	/	202...	10					Med...
trac_addtag	0	cou...	/	202...	12					Med...
trac_ctag	"	cou...	/	202...	11					Med...
trac_lptag	"	cou...	/	202...	12					Med...
trac_ltime	"	cou...	/	202...	12					Med...
trac_sid	"	cou...	/	202...	10					Med...
trac_appver	"	cou...	/	202...	13					Med...
CARTIC	1	cou...	/	Sess...	7					Med...
CCSID	0XGcVBVjCtoQ8BgVOzIP5Q%3D%3D	cou...	/	202...	33					Med...
SUBCARTIC	0	cou...	/	Sess...	10					Med...
bm_sv	EF39824B726FC6823BE2753458BA8...	cou...	/	202...	300	✓				Med...

Request Cookies

Response Cookies

# 쿠키를 이용한 장바구니 예시 (4/5)

메인 페이지 이동 - 장바구니 유지 상태 확인

The screenshot displays a browser window with the Coupang homepage loaded. The Network tab of the developer tools is open, showing a list of requests. One specific request to `https://www.coupang.com/` is selected, and its details are shown in the right panel. The 'Cookies' section of this panel is highlighted with a red border, displaying a long string of cookie data.

**Selected Request Details:**

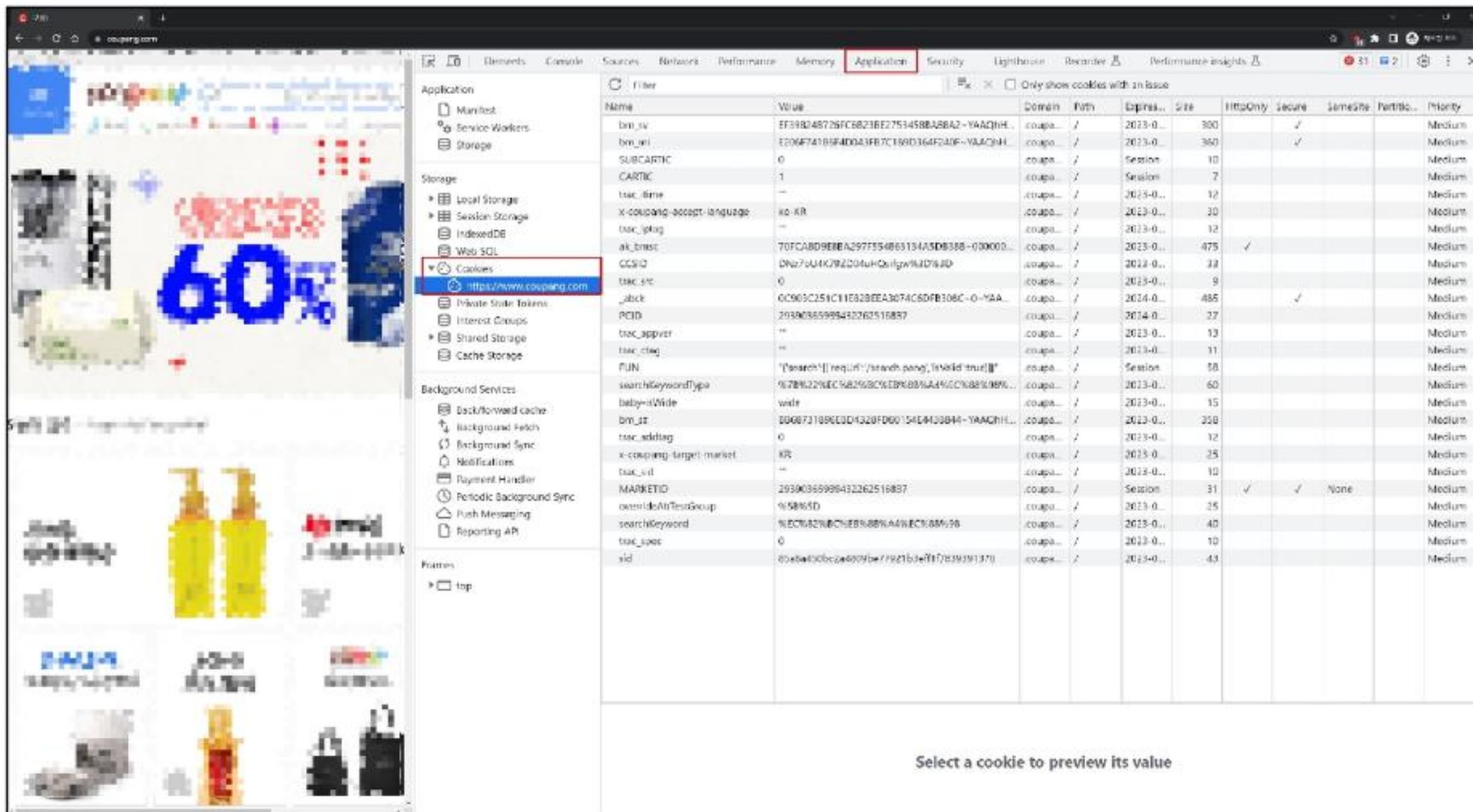
- Request URL: `https://www.coupang.com/`
- Request Method: GET
- Status Code: 200
- Remote Address: 23.40.45.205:443
- Referrer Policy: strict-origin-when-cross-origin

**Cookie Content (highlighted):**

```
sid=85e8a450bc2a4809be77921b3eff1f7839391378; PCID=29390365939432262516837; overridesAbTestGroup=%5B%5D; MARKETID=29390365939432262516837; k-coupa...  
accept-language=ko-KR; x-coupa-target-market=KR;  
ak_bmsc=70FC4BD0E88A297F5548631345DB388-00000000000000000000000000000000  
~YAAQHIXTF0KpxASIAQAAy8QoFtxMwriTeuh00gqWoztCvO5Olxshsg607Jliq8or/6DfY  
BZx8wnUfJ9fOV0lJuCgX+uJvhOf/3ODqz9EX+Wgj8MdB/yTB1vub+F9GsjQAt10wlXUMT  
ts8Ppl2G5e+GLq+xXVWiteTwLtd0s13EjlvVMP5qeFG068sDdyc4eKifG7JVCWqxGOVzO4chK  
8eI8QjAczNtsj0kHihdwqRdkbZ2ZBxt5uzWYg/Gpq5b3kpjhItOUuv6hPTr3feLxL4E4Gmrx  
79HkH0AJdgRd5e4D4QzjMV2qj(DNzZSWGlw3duegFnARQxDH6kyVg@5Vjk9xGq)8ag?Yru  
swaNIkzoedRVTT+Zbow2AmkBrHqTooYSutSfsgd1w=ic  
bm_sz=8866731896EBD4328FD60154E4438844~YAAQHIXTF0SqaASIAQAAyBQaFxQjD+B  
wG6w/uuMuuw128HSgsi+foLYovVQPNcvteCBoeLip5WNlLnw/vv95SVmC+eIr5jLz3BY  
BIO77imTNsFjCCVaqcwbsl79pxz9nFMVdkcPwkWW75hOMhba5BCx9te9MjsnB1vyPKG/Gc  
KmREON/VSPVFaadlw4xn3g91hlm8ixM3s2nTs1F29G+F8X2UMYNwWmlpB0X4ptUb1/x  
WCurUg7adHRCc0PWthvliQaNOjvn1GqNEsmV9bCFh9ouVTFrHj/j3DgOPg-4273458-44
```

# 쿠키를 이용한 장바구니 예시 (5/5)

- 개발자 도구 - Application 탭 - Cookies
- 마우스 우측 버튼 - Clear - 새로고침 - 장바구니가 빈 것을 확인



# 쿠키의 작동 원리와 활용 (1/2)

## 1. 쿠키 저장 방식

- 브라우저(클라이언트)는 쿠키를 KEY-VALUE의 데이터 형식으로 저장
- 쿠키에는 이름, 값 외에도 만료 시간, 도메인, 경로 등의 추가 속성이 포함 됨

## 2. 쿠키 전송 과정

- 서버는 HTTP 응답 헤더의 Set-Cookie 필드를 통해 클라이언트에게 쿠키를 전송
- 브라우저는 받은 쿠키를 저장해 두었다가, 동일한 서버에 재요청 시 HTTP 요청 Header의 Cookie 필드에 저장된 쿠키를 함께 전송

## 쿠키의 작동 원리와 활용 (2/2)

### 3. 쿠키의 주요 용도

- 두 요청이 동일한 브라우저에서 들어왔는지 아닌지를 판단할 때 주로 사용됨
  - 이를 이용해 사용자의 로그인 상태를 유지할 수 있음
  - 상태가 없는(stateless) HTTP 프로토콜에서 상태 정보를 기억시켜 주는 역할
- 서버에게 “나 로그인된 사용자야!”라는 인증 정보가 담긴 쿠키를 매 요청마다 계속 보내는 것

# 쿠키 사용 목적

1. 세션 관리 (Session management)
  - 로그인, 아이디 자동완성, 공지 하루 안 보기, 팝업 체크, 장바구니 등의 정보 관리
2. 개인화 (Personalization)
  - 사용자 선호 설정(언어 설정, 테마 등) 저장
3. 트래킹 (Tracking)
  - 사용자 행동을 기록 및 분석

# 세션

# 세션(Session)

---

서버 측에서 생성되어 클라이언트와 서버 간의 상태를 유지  
상태 정보를 저장하는 데이터 저장 방식

# 세션(Session)

-----  
서버 측에서 생성되어 클라이언트와 서버 간의 상태를 유지  
상태 정보를 저장하는 데이터 저장 방식

- 쿠키에 세션 데이터를 저장하여 매 요청시마다 세션 데이터를 함께 보냄

## 세션 작동 원리

1. 클라이언트가 로그인 요청 후 인증에 성공하면 서버가 **session** 데이터를 생성 후 저장
2. 생성된 **session** 데이터에 인증 할 수 있는 **session id**를 발급
3. 발급한 **session id**를 클라이언트에게 응답 (데이터는 서버에 저장, 열쇠만 주는 것)
4. 클라이언트는 응답 받은 **session id**를 쿠키에 저장
5. 클라이언트가 다시 동일한 서버에 접속하면 요청과 함께 쿠키(**session id**가 저장된)를 서버에 전달
6. 쿠키는 요청 때마다 서버에 함께 전송 되므로 서버에서 **session id**를 확인해 로그인 되어있다는 것을 계속해서 확인하도록 함

서버 측에서는  
세션 데이터를 생성 후 저장하고  
이 데이터에 접근할 수 있는 세션 ID를 생성

이 ID를 클라이언트 측으로 전달하고,  
클라이언트는 쿠키에 이 ID를 저장

이후  
**클라이언트가 같은 서버에 재요청 시마다  
저장해 두었던 쿠키도 요청과 함께 전송**

- 예를 들어 로그인 상태 유지를 위해  
로그인 되어있다는 사실을 입증하는 데이터를  
매 요청마다 계속해서 보내는 것

## 쿠키와 세션의 목적

---

클라이언트와 서버 간의 상태 정보를 유지하고  
사용자를 식별하기 위해 사용

# 이어서...

삼성 청년 SW 아카데미

# Django Authentication System

---

# Django Authentication System

---

사용자 인증과 관련된 기능을 모아 놓은 시스템

# Authentication

(인증)

사용자가 자신이 누구인지 확인하는 것  
(신원 확인)

## 사전 준비

- 두 번째 app **accounts** 생성 및 등록
  - ❖ auth와 관련한 경로나 키워드들을 django 내부적으로 accounts라는 이름으로 사용하고 있기 때문에 되도록 '**accounts**'로 지정하는 것을 권장

```
# accounts/urls.py

from django.urls import path
from . import views

app_name = 'accounts'
urlpatterns = [
    ...
]
```

```
# crud/urls.py

urlpatterns = [
    ...,
    path('accounts/', include('accounts.urls')),
]
```

# 이어서...

삼성 청년 SW 아카데미

# Custom User model

# User model 대체하기

# 기본 User Model의 한계

- 우리는 지금까지 별도의 User 클래스 정의 없이 내장된 auth 앱에 작성된 User 클래스를 사용했음
- Django의 기본 User 모델은 username, password 등 제공되는 필드가 매우 제한적
- 추가적인 사용자 정보(예: 생년월일, 주소, 나이 등)가 필요하다면 이를 위해 기본 User Model을 변경하기 어려움
  - 별도의 설정 없이 사용할 수 있어 간편하지만, 개발자가 직접 수정하기 어려움

## 내장된 auth 앱

```
# settings.py

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

# User Model 대체의 필요성

- 프로젝트의 특정 요구사항에 맞춰 사용자 모델을 확장할 수 있음
- 예를 들어 이메일을 username으로 사용하거나, 다른 추가 필드를 포함시킬 수 있음

# Custom User Model로 대체하기 (1/3)

- AbstractUser 클래스를 상속받는 커스텀 User 클래스 작성
  - 기존 User 클래스도 AbstractUser를 상속받기 때문에  
커스텀 User 클래스도 기존 User 클래스와 완전히 같은 모습을 가지게 됨

```
# accounts/models.py

from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    pass
```

# Custom User Model로 대체하기 (2/3)

- django 프로젝트에서 사용하는 기본 User 모델을 우리가 작성한 User 모델로 사용할 수 있도록 AUTH\_USER\_MODEL 값을 변경
  - 수정 전 기본 값은 ‘auth.User’

```
# settings.py

AUTH_USER_MODEL = 'accounts.User'
```

# Custom User Model로 대체하기 (3/3)

- admin site에 대체한 User 모델 등록
  - 기본 User 모델이 아니기 때문에 등록하지 않으면 admin 페이지에 출력되지 않기 때문

```
# accounts/admin.py

from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .models import User

admin.site.register(User, UserAdmin)
```

# AUTH\_USER\_MODEL

---

Django 프로젝트의 User를 나타내는 데  
사용하는 모델을 지정하는 속성

※ 주의 ※

**프로젝트 중간에**

**AUTH\_USER\_MODEL을 변경 할 수 없음**

- ❖ 이미 프로젝트가 진행되고 있을 경우 데이터베이스 초기화 후 진행

# 사용하는 User 테이블의 변화

- > articles\_article
- > auth\_group
- > auth\_group\_permissions
- > auth\_permission
- > auth\_user
- > auth\_user\_groups
- > auth\_user\_user\_permissions
- > django\_admin\_log
- > django\_content\_type
- > django\_migrations
- > django\_session



- > accounts\_user
- > accounts\_user\_groups
- > accounts\_user\_user\_permissions
- > articles\_article
- > auth\_group
- > auth\_group\_permissions
- > auth\_permission
- > django\_admin\_log
- > django\_content\_type
- > django\_migrations
- > django\_session

# 프로젝트를 시작하며 반드시 User 모델을 대체해야 한다.

- Django는 새 프로젝트를 시작하는 경우 비록 기본 User 모델이 충분하더라도 커스텀 User 모델을 설정하는 것을 **강력하게 권장**하고 있음
- 커스텀 User 모델은 **기본 User 모델과 동일하게 작동(1)** 하면서도 필요한 경우 나중에 맞춤 설정(2)할 수 있기 때문
  - ❖ 단, User 모델 대체 작업은 프로젝트의 모든 migrations 혹은 첫 migrate를 실행하기 전에 이 작업을 마쳐야 함

# 이어서...

삼성 청년 SW 아카데미

# Login

---

# Login

---

로그인은 ??????을 Create하는 과정

# Login

---

로그인은 Session을 Create하는 과정

# AuthenticationForm()

---

로그인 인증에 사용할 데이터를 입력 받는  
built-in form

# 로그인 페이지 작성 (1/2)

```
# accounts/urls.py

app_name = 'accounts'
urlpatterns = [
    path('login/', views.login, name='login'),
]

<!-- accounts/login.html --&gt;

&lt;h1&gt;로그인&lt;/h1&gt;
&lt;form action="{% url 'accounts:login' %}" method="POST"&gt;
    {% csrf_token %}
    {{ form.as_p }}
    &lt;input type="submit"&gt;
&lt;/form&gt;</pre>
```

```
# accounts/views.py

from django.contrib.auth.forms import AuthenticationForm

def login(request):
    if request.method == 'POST':
        pass
    else:
        form = AuthenticationForm()
    context = {
        'form': form,
    }
    return render(request, 'accounts/login.html', context)
```

## 로그인 페이지 작성 (2/2)

Document x +

127.0.0.1:8000/accounts/login/ 시크릿 모드(창 2개) :

로그인

Username:

Password:

제출

AuthenticationForm

# 로그인 로직 작성

```
# accounts/views.py

from django.shortcuts import render, redirect
from django.contrib.auth import login as auth_login

def login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        # form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            auth_login(request, form.get_user())
            return redirect('articles:index')
    else:
        form = AuthenticationForm()
    context = {
        'form': form,
    }
    return render(request, 'accounts/login.html', context)
```

# **login(request, user)**

---

AuthenticationForm을 통해  
인증된 사용자를 로그인 하는 함수

# get\_user()

---

AuthenticationForm의 인스턴스 메서드

- 유효성 검사를 통과했을 경우 로그인 한 사용자 객체를 반환

# 세션 데이터 확인하기 (1/2)

- 로그인 후 발급받은 세션 확인
  - django\_session 테이블에서 확인

The screenshot shows a SQLite database browser interface. On the left, a sidebar lists tables in the database, including articles\_article, auth\_group, auth\_group\_permissions, auth\_permission, auth\_user, auth\_user\_groups, auth\_user\_user\_permissions, django\_admin\_log, django\_content\_type, django\_migrations, django\_session (which is currently selected and highlighted in blue), and sqlite\_sequence. The main area displays a query window with the following content:

```
Properties DATA
SELECT * FROM django_session LIMIT 100
```

The results pane shows one row of data from the django\_session table. The columns are session\_key (varchar(40)), session\_data (TEXT), and expire\_date (datetime). The session\_key value is highlighted with a red box. The session\_data value is truncated at the end.

	*	session_key	session_data	expire_date
	> 1	rryidpralk8p63l7g8a9htmqh9e	.eJxVjEEOwiAQRe_C2pASBgd...	2023-10-24 10:15:45

# 세션 데이터 확인하기 (2/2)

## 2. 브라우저에서 확인

- 개발자도구 - Application - Cookies

The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, the Storage section is expanded, showing Local Storage, Session Storage, IndexedDB, Web SQL, and Cookies. Under Cookies, there are two entries: 'sessionid' and 'csrfToken'. Both entries have their values highlighted in red. The 'sessionid' value is 'qj258qt9vr2mdrygj79ic2m87vse5tj' and it expires on 2023-07-12. The 'csrfToken' value is 'kHptIbyYap8tk5Pq2xaaxQnR0l6FLUfU' and it expires on 2024-07-12. A message at the bottom of the DevTools window says 'Select a cookie to preview its value'.

Name	Value	Dom	Path	Expire	Size	Http	Secure	SameSite	Parti	Pri
sessionid	qj258qt9vr2mdrygj79ic2m87vse5tj	127....	/	2023...	41	✓	Lax		Med...	
csrfToken	kHptIbyYap8tk5Pq2xaaxQnR0l6FLUfU	127....	/	2024...	41		Lax		Med...	

# 로그인 링크 작성

- 메인 페이지에 로그인 페이지로 갈 수 있는 링크 작성

```
<!-- articles/index.html -->

<h1>Articles</h1>
Login
NEW
<hr>
```

# 이어서...

삼성 청년 SW 아카데미

# Logout

---

# Logout

---

로그아웃은 ??????을 Delete하는 과정

# Logout

---

로그아웃은 Session을 Delete하는 과정

# logout(request)

---

1. DB에서 현재 요청에 대한 Session Data를 삭제
2. 클라이언트의 쿠키에서도 Session Id를 삭제

# 로그아웃 로직 작성 (1/2)

```
# accounts/urls.py

urlpatterns = [
    path('login/', views.login, name='login'),
    path('logout/', views.logout, name='logout'),
]
```

```
<!-- articles/index.html -->

<h1>Articles</h1>
<a href="{% url 'accounts:login' %}">Login</a>
<form action="{% url 'accounts:logout' %}" method="POST">
    {% csrf_token %}
    <input type="submit" value="Logout">
</form>
```

```
# accounts/views.py

from django.contrib.auth import logout as auth_logout

def logout(request):
    auth_logout(request)
    return redirect('articles:index')
```

# 로그아웃 로직 작성 (2/2)

로그아웃 진행 및 세션 데이터 삭제 확인

The screenshot shows the Chrome DevTools Application tab for a web application titled "Articles". On the left, there's a sidebar with links for "Login", "Logout", and "NEW". The main area displays a table of session storage items. One item is highlighted: "csrfmiddlewaretoken" with the value "kHptbyYbp8tix5Rq2xaQnRn6JFLRU". The table includes columns for Name, Value, Domain, Path, Expires, Size, HttpOnly, Secure, SameSite, and Priority.

Name	Value	Dom...	Path	Expir...	Size	Http...	Secur...	Same...	Priority
csrfmiddlewaretoken	kHptbyYbp8tix5Rq2xaQnRn6JFLRU	127....	/	2024...	41			Lax	Med...

Select a cookie to preview its value

The screenshot shows a SQLite database browser interface connected to a database at "/Users/har.../db.sqlite3". The browser displays the schema of the "django\_session" table, which has 12 columns: session\_key (varchar(40)), session\_data (TEXT), and expire\_date (datetime). A query is shown: "SELECT \* FROM django\_session LIMIT 100". The browser has a "Properties" tab and a "DATA" tab, with the "DATA" tab currently active.

session_key	session_data	expire_date
session_key	session_data	expire_date

# 이어서...

삼성 청년 SW 아카데미

# Template with Authentication data

---

# 템플릿과 인증 데이터

# Template with Authentication data

---

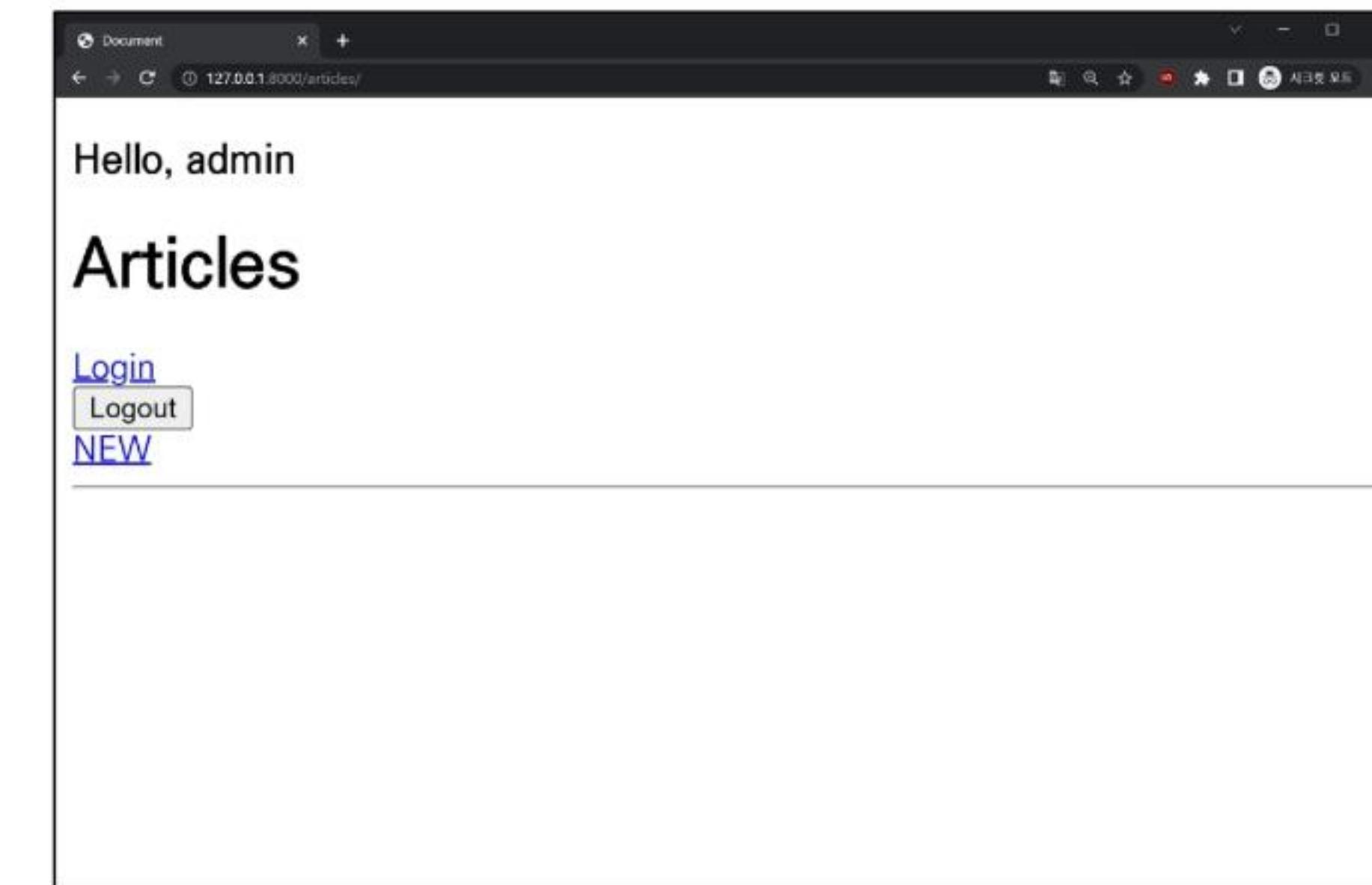
템플릿에서 인증 관련 데이터를 출력하는 방법

# 현재 로그인 되어있는 유저 정보 출력하기

user라는 context 데이터를 사용할 수 있는 이유는?

- django가 미리 준비한 context 데이터가 존재하기 때문(context processors)

```
<!-- articles/index.html -->  
  
<h3>Hello, {{ user.username }}</h3>
```



# context processors

- 템플릿이 렌더링 될 때 호출 가능한 컨텍스트 데이터 목록
  - 작성된 컨텍스트 데이터는 기본적으로 템플릿에서 사용 가능한 변수로 포함됨
- django에서 자주 사용하는 데이터 목록을 미리 템플릿에 로드 해 둔 것

```
# settings.py

TEMPLATES = [
    {
        ...
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]
```

# 이어서...

삼성 청년 SW 아카데미

# 참고

# 쿠키의 수명

# 쿠키 종류별 Lifetime (수명)

1. Session cookie
  - 현재 세션(current session)이 종료되면 삭제됨
  - 브라우저 종료와 함께 세션이 삭제됨
2. Persistent cookies
  - Expires 속성에 지정된 날짜 혹은 Max-Age 속성에 지정된 기간이 지나면 삭제됨

# 쿠키와 보안

# 쿠키의 보안 장치

- 제한된 정보
  - 쿠키에는 보통 중요하지 않은 정보만 저장. (사용자 ID나 세션 번호 같은 것)
- 암호화
  - 중요한 정보는 서버에서 암호화해서 쿠키에 저장
- 만료 시간
  - 쿠키에는 만료 시간을 설정 시간이 지나면 자동으로 삭제
- 도메인 제한
  - 쿠키는 특정 웹사이트에서만 사용할 수 있도록 설정할 수 있음

# 쿠키와 개인정보 보호

- 많은 국가에서 쿠키 사용에 대한 사용자 동의를 요구하는 법규를 시행
- 웹사이트는 쿠키 정책을 명시하고, 필요한 경우 사용자의 동의를 얻어야 함

# Django에서의 세션 관리

# 세션 in Django

- Django는 ‘database-backed sessions’ 저장 방식을 기본 값으로 사용
  - session 정보는 DB의 **django\_session** 테이블에 저장
  - Django는 요청안에 특정 session id를 포함하는 쿠키를 사용해서 각각의 브라우저와 사이트가 연결된 session 데이터를 알아냄
- Django는 우리가 session 메커니즘(복잡한 동작원리)에 대부분을 생각하지 않게끔 많은 도움을 줌

# AuthenticationForm

## 내부 코드

# django github 코드 참고

- **AuthenticationForm()**
  - <https://github.com/django/django/blob/main/django/contrib/auth/forms.py#L286>
- **AuthenticationForm()의 get\_user() 인스턴스 메서드**
  - <https://github.com/django/django/blob/main/django/contrib/auth/forms.py#L356>

# AbstractUser class

## ‘AbstractUser’ class

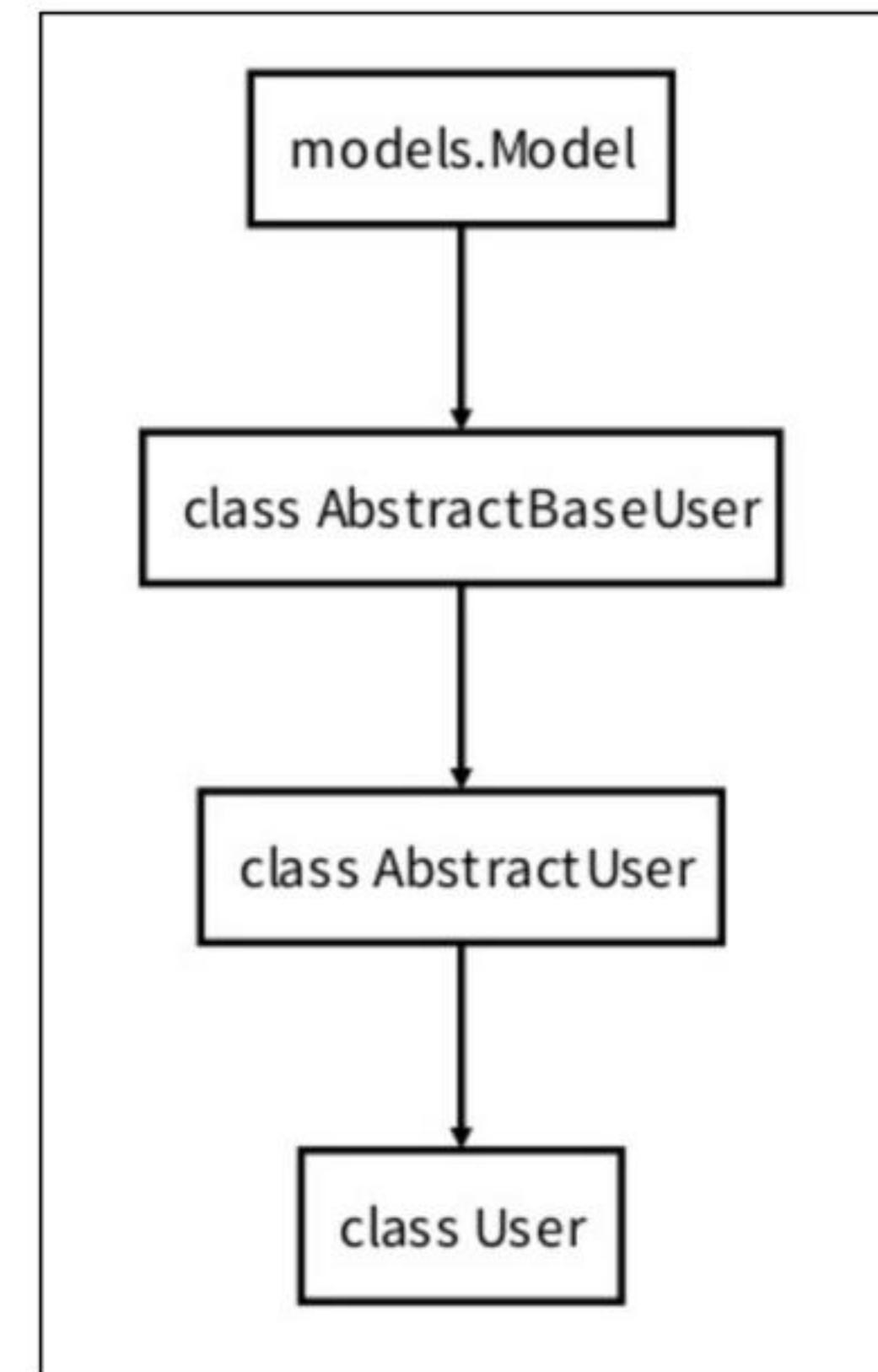
---

“관리자 권한과 함께 완전한 기능을 가지고 있는  
User model을 구현하는 추상 기본클래스”

# Abstract base classes (추상 기본 클래스)

- 몇 가지 공통 정보를 여러 다른 모델에 넣을 때 사용하는 클래스
- 데이터베이스 테이블을 만드는데 사용되지 않으며, 대신 다른 모델의 기본 클래스로 사용되는 경우 해당 필드가 하위 클래스의 필드에 추가됨

# User 모델 상속 관계



# User 모델 대체하기 Tip

# User 모델 대체하기 Tip

- User 모델을 대체하는 순서를 숙지하기 어려울 경우 해당 공식문서를 보면 순서대로 진행하는 것을 권장

다음 시간에  
만나요!

삼성 청년 SW 아카데미

# Django 09 Authentication System 02

# • INDEX

---

- 회원 가입
- 회원 탈퇴
- 회원정보 수정
- 비밀번호 변경
  - 세션 무효화 방지
- 로그인 사용자에 대한 접근 제한
  - `is_authenticated` 속성
  - `login_required` 데코레이터
- 참고
  - `is_authenticated` 코드
  - 회원가입 후 자동 로그인
  - 회원 탈퇴 개선
  - `PasswordChangeForm` 인자 순서
  - Auth built-in form 코드

# 회원가입

# 회원 가입

---

User 객체를 Create 하는 과정

# UserCreationForm()

---

회원 가입시 사용자 입력 데이터를 받는  
built-in ModelForm

# 회원가입 페이지 작성 (1/2)

```
# accounts/urls.py

app_name = 'accounts'
urlpatterns = [
    ...,
    path('signup/', views.signup, name='signup'),
]
```

```
<!-- accounts/signup.html -->

<h1>회원가입</h1>
<form action="{% url 'accounts:signup' %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit">
</form>
```

```
# accounts/views.py

from django.contrib.auth.forms import UserCreationForm

def signup(request):
    if request.method == 'POST':
        pass
    else:
        form = UserCreationForm()
    context = {
        'form': form,
    }
    return render(request, 'accounts/signup.html', context)
```

# 회원가입 페이지 작성 (2/2)

- 회원가입 페이지 확인

The screenshot shows a web browser window with a dark theme. The title bar says 'Document'. The address bar shows the URL '127.0.0.1:8000/accounts/signup/'. The main content area has a title '회원가입'. It contains fields for 'Username' and 'Password', both with placeholder text. Below the password field is a list of four password requirements. There is also a 'Password confirmation' field and a red '제출' button at the bottom.

회원가입

Username:  Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:  Enter the same password as before, for verification.

제출

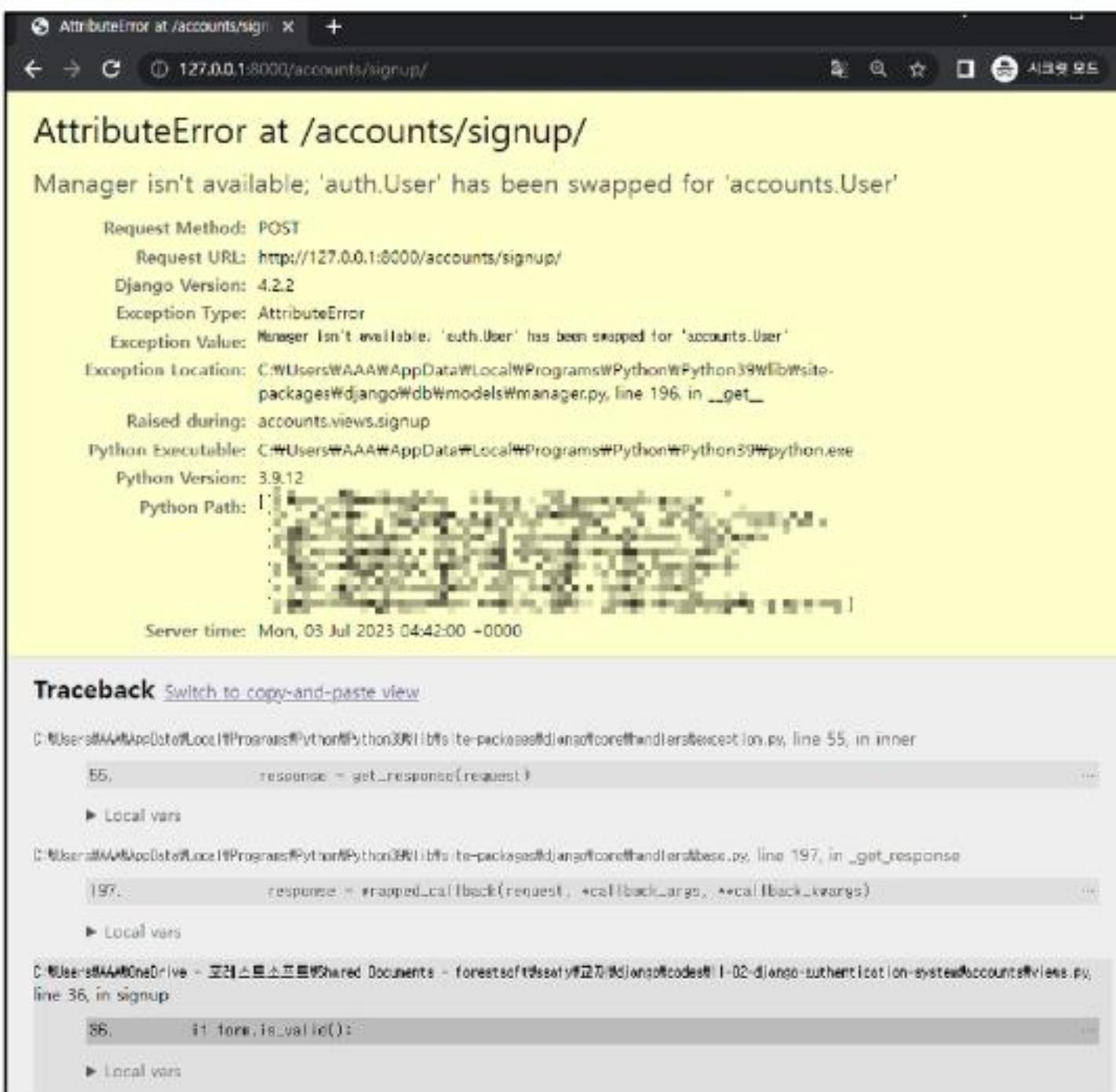
# 회원가입 로직 작성

```
# accounts/views.py

def signup(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('articles:index')
    else:
        form = UserCreationForm()
    context = {
        'form': form,
    }
    return render(request, 'accounts/signup.html', context)
```

# 회원가입 로직 에러 (1/2)

- 회원가입 시도 후 에러 페이지 확인
  - Manager isn't available; 'auth.User' has been swapped for 'accounts.User'



## 회원가입 로직 에러 (2/2)

- 회원가입에 사용하는 `UserCreationForm`이 대체한 커스텀 유저 모델이 아닌 과거 Django의 기본 유저 모델로 인해 작성된 클래스이기 때문

```
class Meta:  
    model = User  
    fields = ("username",)  
    field_classes = {"username": UsernameField}
```

# 커스텀 유저 모델을 사용하려면 다시 작성해야 하는 Form

UserCreationForm

UserChangeForm

두 Form 모두

class Meta: model = User

가 작성된 Form이기 때문에 재작성 필요

# UserCreationForm과 UserChangeForm 커스텀

- Custom User model을 사용할 수 있도록 상속 후 일부분만 재작성

```
# accounts/forms.py

from django.contrib.auth import get_user_model
from django.contrib.auth.forms import UserCreationForm, UserChangeForm

class CustomUserCreationForm(UserCreationForm):
    class Meta(UserCreationForm.Meta):
        model = get_user_model()

class CustomUserChangeForm(UserChangeForm):
    class Meta(UserChangeForm.Meta):
        model = get_user_model()
```

# get\_user\_model()

-----  
“현재 프로젝트에서 활성화된 사용자 모델(active user model)”  
을 반환하는 함수

# User 모델을 직접 참조하지 않는 이유

- `get_user_model()`을 사용해 User 모델을 참조하면 커스텀 User 모델을 자동으로 반환해주기 때문
  - Django는 필수적으로 User 클래스를 직접 참조하는 대신 `get_user_model()`을 사용해 참조해야 한다고 강조하고 있음
- User model 참조에 대한 자세한 내용은 추후 모델 관계에서 다룰 예정

# 회원가입 로직 완성

- CustomUserCreationForm으로 변경

```
# accounts/views.py

from .forms import CustomUserCreationForm

def signup(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('articles:index')
    else:
        form = CustomUserCreationForm()
    context = {
        'form': form,
    }
    return render(request, 'accounts/signup.html', context)
```

# 이어서...

삼성 청년 SW 아카데미

# 회원 탈퇴

# 회원 탈퇴

---

User 객체를 Delete 하는 과정

## 회원 탈퇴 로직 작성 (1/2)

```
# accounts/urls.py

app_name = 'accounts'
urlpatterns = [
    ...,
    path('delete/', views.delete, name='delete'),
]
```

```
<!-- accounts/index.html -->

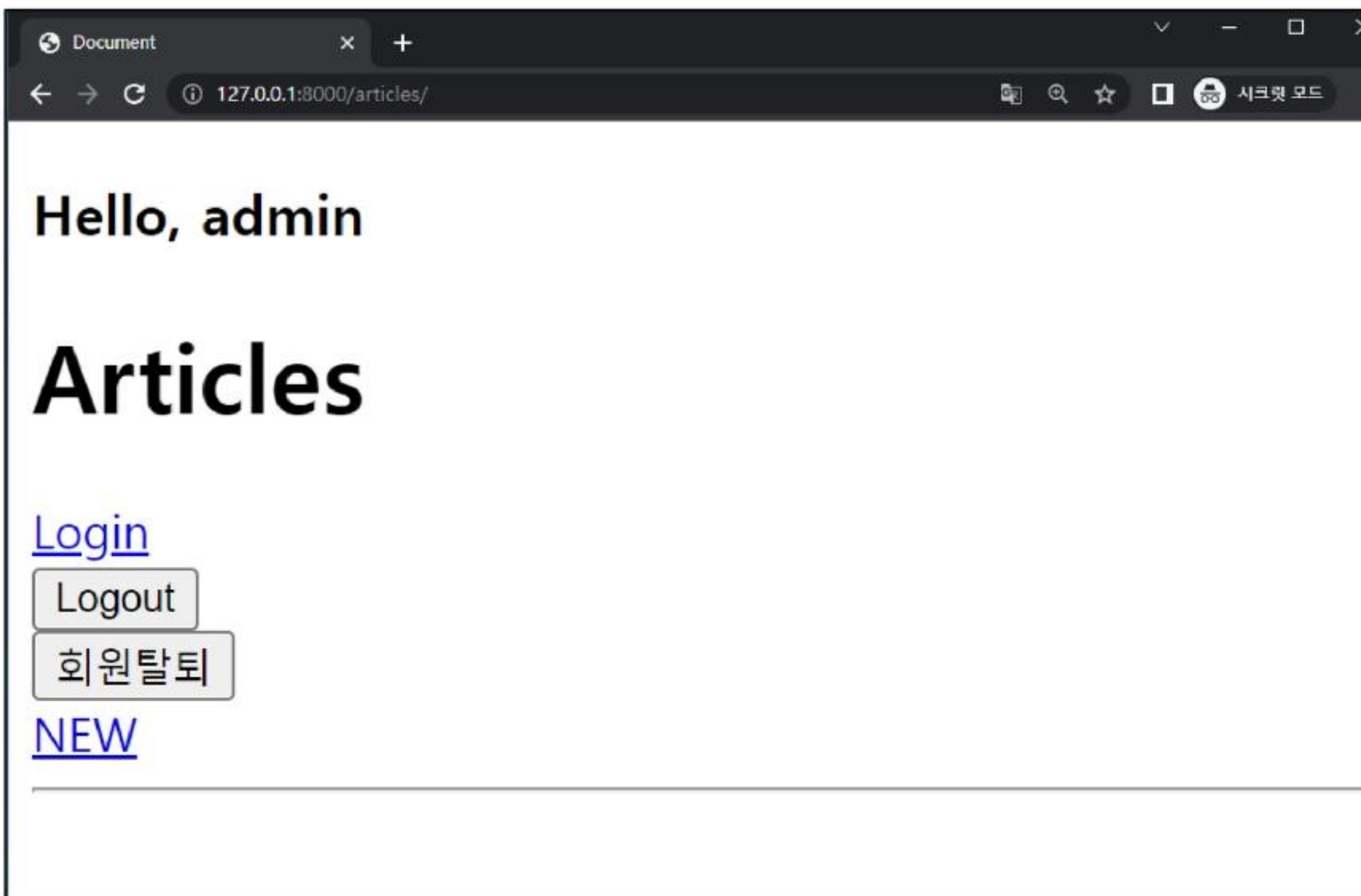
<form action="{% url 'accounts:delete' %}" method="POST">
    {% csrf_token %}
    <input type="submit" value="회원탈퇴">
</form>
```

```
# accounts/views.py

def delete(request):
    request.user.delete()
    return redirect('articles:index')
```

## 회원 탈퇴 로직 작성 (2/2)

- 회원 탈퇴 진행



# 이어서...

삼성 청년 SW 아카데미

# 회원정보 수정

---

# 회원정보 수정

---

User 객체를 Update 하는 과정

# UserChangeForm()

---

회원정보 수정 시 사용자 입력 데이터를 받는  
built-in ModelForm

# 회원정보 수정 페이지 작성 (1/3)

```
# accounts/urls.py

app_name = 'accounts'
urlpatterns = [
    ...,
    path('update/', views.update, name='update'),
]

<!-- accounts/update.html --&gt;

&lt;h1&gt;회원정보 수정&lt;/h1&gt;
&lt;form action="{% url 'accounts:update' %}" method="POST"&gt;
    {% csrf_token %}
    {{ form.as_p }}
    &lt;input type="submit"&gt;
&lt;/form&gt;</pre>
```

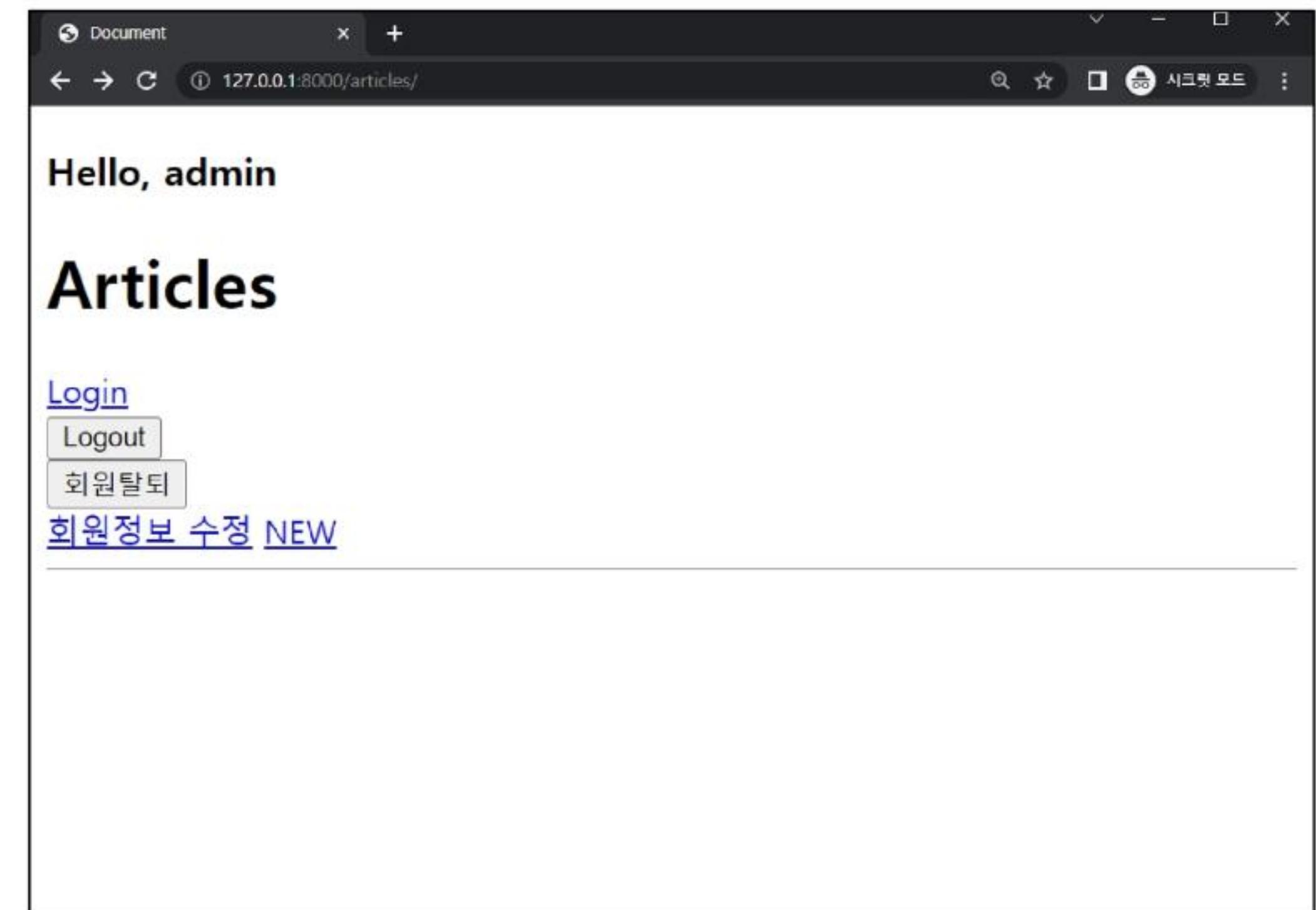
```
# accounts/views.py

from .forms import CustomUserChangeForm

def update(request):
    if request.method == 'POST':
        pass
    else:
        form = CustomUserChangeForm(instance=request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/update.html', context)
```

# 회원정보 수정 페이지 작성 (2/3)

```
# accounts/index.html  
  
<a href="{% url 'accounts:update' %}">회원정보 수정</a>
```



# 회원정보 수정 페이지 작성 (3/3)

- 회원정보 수정 페이지 확인

The screenshot shows a web browser window with the URL `127.0.0.1:8000/accounts/update/`. The page title is "회원정보 수정". It displays a user's information:

- Password:** pbkdf2\_sha256 iterations: 600000 salt: lo0An1\*\*\*\*\* hash: 33X18f\*\*\*\*\*
- Last login:** 2023-07-03 05:28:26
- Superuser status:**  Designates that this user has all permissions without explicitly assigning them.
- Groups:** accounts | user | Can add user  
accounts | user | Can change user  
accounts | user | Can delete user  
User permissions: accounts | user | Can view user
- Username:** admin
- First name:**
- Last name:**
- Email address:**
- Staff status:**  Designates whether the user can log into this admin site.
- Active:**  Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
- Date joined:** 2023-07-02 17:21:51

At the bottom right of the form, there is a button labeled "제출".

# UserChangeForm 사용 시 문제점

- User 모델의 모든 정보들(**fields**)까지 모두 출력됨
  - 일반 사용자들이 접근해서는 안되는 정보는 출력하지 않도록 해야 함
- CustomUserChangeForm에서 출력 필드를 다시 조정하기

# CustomUserChangeForm 출력 필드 재정의 (1/2)

- User Model의 필드 목록 확인
  - <https://docs.djangoproject.com/en/4.2/ref/contrib/auth/>

```
# accounts/forms.py

class CustomUserChangeForm(UserChangeForm):

    class Meta(UserChangeForm.Meta):
        model = get_user_model()
        fields = ('first_name', 'last_name', 'email',)
```

# CustomUserChangeForm 출력 필드 재정의 (2/2)

- 회원정보 수정 페이지 확인

회원정보 수정

First name:

Last name:

Email address:

Password:

No password set.  
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).

제출

# 회원정보 수정 로직 완성

```
# accounts/views.py

def update(request):
    if request.method == 'POST':
        form = CustomUserChangeForm(request.POST, instance=request.user)
        # form = CustomUserChangeForm(data=request.POST, instance=request.user)
        if form.is_valid():
            form.save()
            return redirect('articles:index')
    else:
        form = CustomUserChangeForm(instance=request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/update.html', context)
```

# 이어서...

삼성 청년 SW 아카데미

# 비밀번호 변경

# 비밀번호 변경

---

인증된 사용자의 Session 데이터를 Update 하는 과정

# PasswordChangeForm()

---

비밀번호 변경 시 사용자 입력 데이터를 받는  
built-in Form

# 비밀번호 변경 페이지 작성 (1/3)

django는 비밀번호 변경 페이지를 회원정보 수정 form 하단에서 별도 주소로 안내

➤ /user\_pk/password/

The screenshot shows a browser window with developer tools open. The left pane displays a '회원정보 수정' (User Information Update) form with fields for First name, Last name, Email address, and Password. A note states 'No password set.' and provides a link to change it. The right pane shows the browser's DOM structure in the 'Elements' tab, highlighting the password field and its associated help text.

회원정보 수정

First name:

Last name:

Email address:

Password:

No password set.  
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).

제출

Elements Console Sources Network Performance Memory > 4 ⚙ :

```
<html>
  <head></head>
  <body>
    <h1>회원정보 수정</h1>
    <form action="/accounts/update/" method="POST">
      <input type="hidden" name="csrfmiddlewaretoken" value="gf9uIaiCKZHTBLqbz92Fr3PA00Eb30uZZz2SrtooBGsj7NT4khwKKgHQDWKln01">
      <p>...</p>
      <p>...</p>
      <p>...</p>
      <p>...</p>
      <div disabled id="id_password">...</div>
      <span class="helptext">
        "Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using "
        <a href="../../1/password/">this form</a> == $0
      </span>
    </form>
  </body>
</html>
```

## 비밀번호 변경 페이지 작성 (2/3)

```
# crud/urls.py

from accounts import views

urlpatterns = [
    ...
    path('<int:user_pk>/password/', views.change_password,
name='change_password'),
]
```

```
<!-- accounts/change_password.html -->

<h1>비밀번호 변경</h1>
<form action="{% url 'change_password' user.pk %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit">
</form>
```

```
# accounts/views.py

from django.contrib.auth.forms import PasswordChangeForm

def change_password(request, user_pk):
    if request.method == 'POST':
        pass
    else:
        form = PasswordChangeForm(request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/change_password.html',
context)
```

# 비밀번호 변경 페이지 작성 (3/3)

- 비밀번호 변경 페이지 확인

The screenshot shows a password change form titled "비밀번호 변경" (Password Change) at the URL 127.0.0.1:8000/1/password/. The form includes fields for "Old password", "New password", and "New password confirmation". It also lists several password requirements:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

At the bottom left is a "제출" (Submit) button.

# 비밀번호 변경 로직 완성

```
# accounts/views.py

def change_password(request, user_pk):
    if request.method == 'POST':
        form = PasswordChangeForm(request.user, request.POST)
        # form = PasswordChangeForm(user=request.user, data=request.POST)
        if form.is_valid():
            form.save()
            return redirect('articles:index')
    else:
        form = PasswordChangeForm(request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/change_password.html', context)
```

# 세션 무효화 방지

## 암호 변경 시 세션 무효화

- 비밀번호가 변경되면 기존 세션과의 회원 인증 정보가 일치하지 않게 되어 버려 로그인 상태가 유지되지 못하고 로그아웃 처리됨
- 비밀번호가 변경되면서 기존 세션과의 회원 인증 정보가 일치하지 않기 때문

# **update\_session\_auth\_hash(request, user)**

---

암호 변경 시 세션 무효화를 막아주는 함수

- 암호가 변경되면 새로운 password의 Session Data로 기존 session을 자동으로 갱신

# update\_session\_auth\_hash 적용

```
# accounts/views.py

from django.contrib.auth import update_session_auth_hash

def change_password(request):
    if request.method == 'POST':
        form = PasswordChangeForm(request.user, request.POST)
        # form = PasswordChangeForm(user=request.user, data=request.POST)
        if form.is_valid():
            user = form.save()
            update_session_auth_hash(request, user)
            return redirect('articles:index')
    else:
        form = PasswordChangeForm(request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/change_password.html', context)
```

# 이어서...

삼성 청년 SW 아카데미

# 인증된 사용자에 대한 접근 제한

## 로그인 사용자에 대해 접근을 제한하는 2가지 방법

---

1. `is_authenticated` 속성
2. `login_required` 데코레이터

**is\_authenticated 속성**

# **is\_authenticated**

-----  
사용자가 인증 되었는지 여부를 알 수 있는 User model의 속성

- 모든 User 인스턴스에 대해 항상 True인 읽기 전용 속성
  - 비인증 사용자에 대해서는 항상 False

# is\_authenticated 적용하기 (1/2)

로그인과 비로그인 상태에서 화면에 출력되는 링크를 다르게 설정하기

```
<!-- articles/index.html -->

{% if request.user.is_authenticated %}
    <h3>Hello, {{ user.username }}</h3>
    <a href="{% url 'articles:create' %}">NEW</a>
    <form action="{% url 'accounts:logout' %}" method="POST">
        {% csrf_token %}
        <input type="submit" value="Logout">
    </form>
    <form action="{% url 'accounts:delete' %}" method="POST">
        {% csrf_token %}
        <input type="submit" value="회원탈퇴">
    </form>
    <a href="{% url 'accounts:update' %}">회원정보 수정</a>
{% else %}
    <a href="{% url 'accounts:login' %}">Login</a>
    <a href="{% url 'accounts:signup' %}">Signup</a>
{% endif %}
```

## is\_authenticated 적용하기 (2/2)

인증된 사용자라면 로그인/회원가입 로직을 수행할 수 없도록 하기

```
# accounts/views.py

def login(request):
    if request.user.is_authenticated:
        return redirect('articles:index')
    ...

def signup(request):
    if request.user.is_authenticated:
        return redirect('articles:index')
    ...
```

**login\_required**  
**데코레이터**

# login\_required

---

인증된 사용자에 대해서만 view 함수를 실행시키는 데코레이터

- 비인증 사용자의 경우 /accounts/login/ 주소로 redirect 시킴

# login\_required 적용하기 (1/2)

인증된 사용자만 게시글을 작성/수정/삭제 할 수 있도록 수정

```
# articles/views.py

from django.contrib.auth.decorators import login_required

@login_required
def create(request):
    pass

@login_required
def delete(request, article_pk):
    pass

@login_required
def update(request, article_pk):
    pass
```

## login\_required 적용하기 (2/2)

인증된 사용자만 로그아웃/탈퇴/수정/비밀번호 변경 할 수 있도록 수정

```
# accounts/views.py

from django.contrib.auth.decorators import login_required

@login_required
def logout(request):
    pass

@login_required
def delete(request):
    pass

@login_required
def update(request):
    pass

@login_required
def change_password(request):
    pass
```

# 이어서...

삼성 청년 SW 아카데미

# 참고

**is\_authenticated 코드**

# is\_authenticated 속성 코드

- 메서드가 아닌 속성 값임을 주의

```
@property
def is_authenticated(self):
    """
    Always return True. This is a way to tell if the user has been
    authenticated in templates.
    """
    return True
```

## 회원가입 후 자동 로그인

# 회원가입 후 로그인까지 이어서 진행하려면?

- 회원가입 성공한 user 객체를 활용해 login 진행

```
# accounts/views.py

def signup(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            auth_login(request, user)
            return redirect('articles:index')
    else:
        form = CustomUserCreationForm()
    context = {
        'form': form,
    }
    return render(request, 'accounts/signup.html', context)
```

```
def save(self, commit=True):
    user = super().save(commit=False)
    user.set_password(self.cleaned_data["password1"])
    if commit:
        user.save()
    return user
```

❖ UserCreationForm의 save 메서드

# 회원 탈퇴 개선

# 탈퇴와 함께 기존 사용자의 Session Data 삭제 방법

- 사용자 객체 삭제 이후 로그아웃 함수 호출
- 단, "탈퇴(1) 후 로그아웃(2)"의 순서가 바뀌면 안됨
- 먼저 로그아웃이 진행되면 해당 요청 객체 정보가 없어지기 때문에 탈퇴에 필요한 유저 정보 또한 없어지기 때문

```
# accounts/views.py

def delete(request):
    request.user.delete()
    auth_logout(request)
```

# PasswordChangeForm

## 인자 순서

# PasswordChangeForm의 인자 순서

- PasswordChangeForm이 다른 Form과 달리 user 객체를 첫번째 인자로 받는 이유
- 부모 클래스인 SetPasswordForm의 생성자 함수 구성을 따르기 때문

```
class SetPasswordForm(forms.Form):
    """
    A form that lets a user set their password without entering the old
    password
    """

    error_messages = {
        "password_mismatch": _("The two password fields didn't match."),
    }
    new_password1 = forms.CharField(
        label=_("New password"),
        widget=forms.PasswordInput(attrs={"autocomplete": "new-password"}),
        strip=False,
        help_text=password_validation.password_validators_help_text_html(),
    )
    new_password2 = forms.CharField(
        label=_("New password confirmation"),
        strip=False,
        widget=forms.PasswordInput(attrs={"autocomplete": "new-password"}),
    )

    def __init__(self, user, *args, **kwargs):
        self.user = user
        super().__init__(*args, **kwargs)
```

# Auth built-in form 코드

# Auth built-in form github 코드

- UserCreationForm()
  - <https://github.com/django/django/blob/4.2/django/contrib/auth/forms.py#L149>
- UserChangeForm()
  - <https://github.com/django/django/blob/4.2/django/contrib/auth/forms.py#L170>
- PasswordChangeForm()
  - <https://github.com/django/django/blob/4.2/django/contrib/auth/forms.py#L422>

다음 시간에  
만나요!

삼성 청년 SW 아카데미

# 05 PJT

## 챕터의 포인트

- 관통 Ver1. 키워드 검색량 분석을 위한 데이터 수집
- 관통 Ver2. 인증 페이지 구현

## | 라이브 진행 순서

- 목표
- 웹 크롤링 이해하기
- 웹 크롤링 실습
  - 데이터 저장하기
  - 유저 인증 기능 추가하기 (로그인, 로그아웃, 인증된 유저만 데이터 조회)

# 목표

## 프로젝트 파악하기

- 친구들끼리 같이 먹을 음식을 주문하기로 했다.
- 갑자기 궁금해졌다.

사람들이

깐풍기를 더 선호할까?

탕수육을 더 선호할까?



금융상품비교

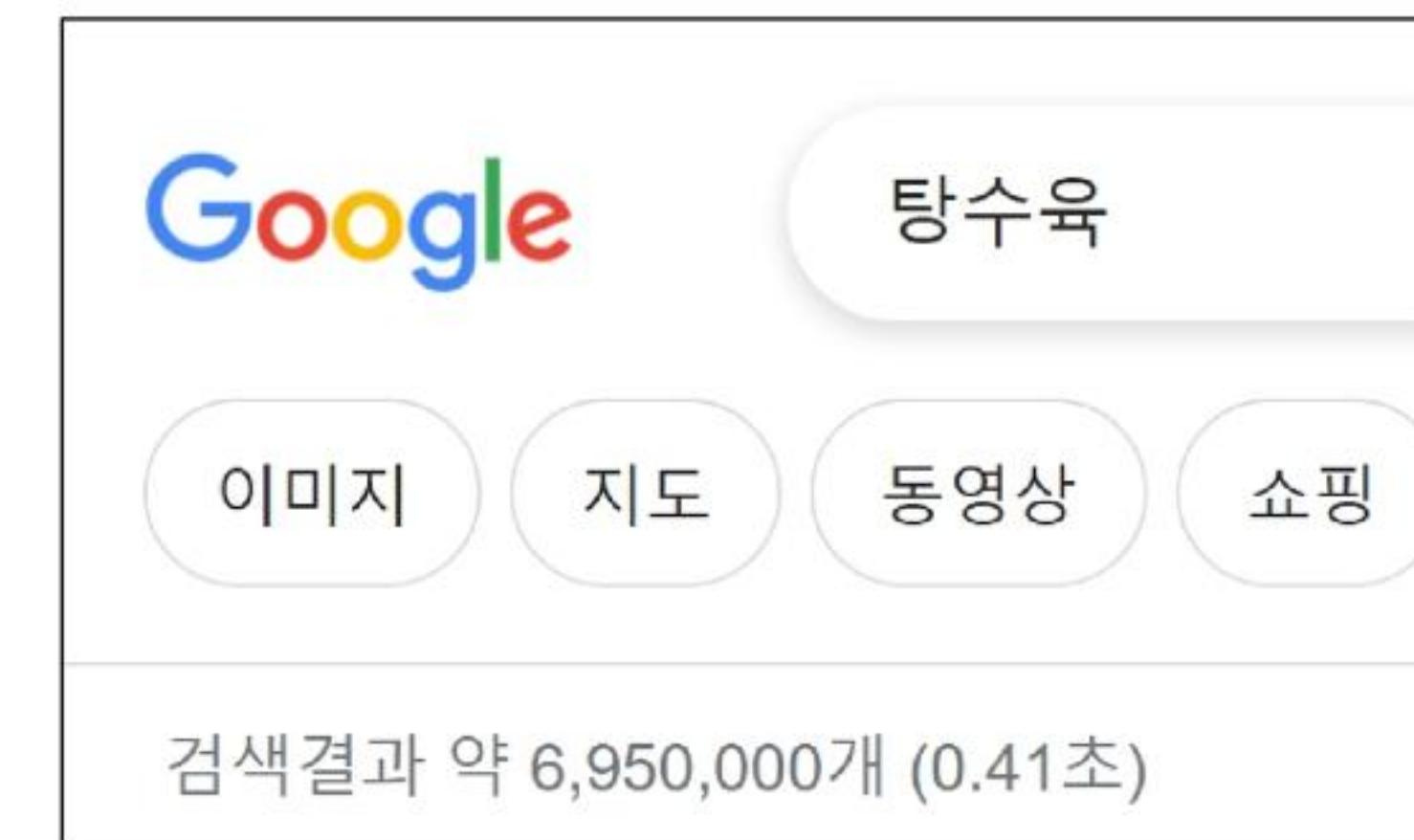
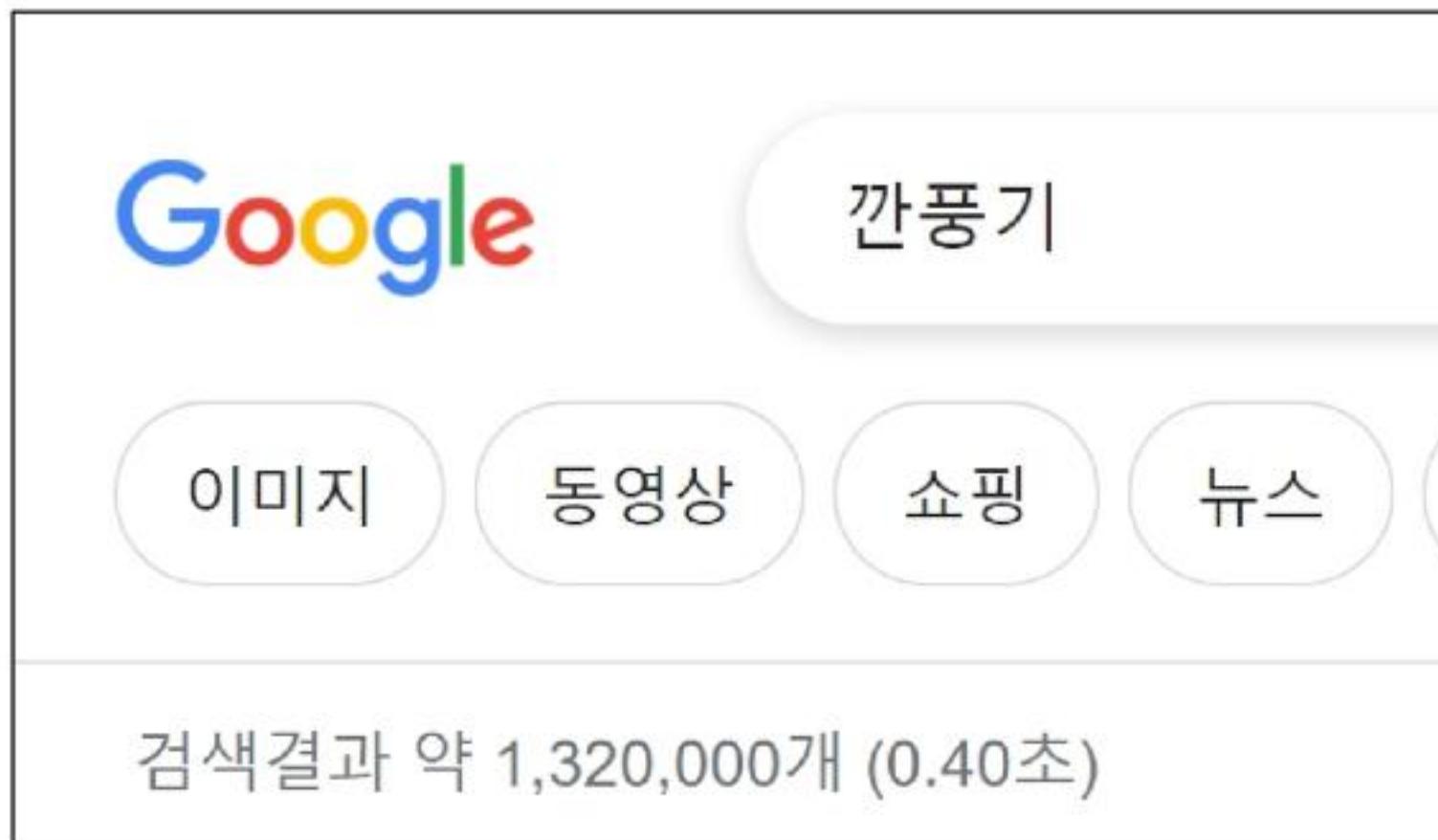
영화추천서비스

## 프로젝트 파악하기

금융상품비교

영화추천서비스

- 구글에 검색해보고 어떤 메뉴가 더 많이 검색되는지로 판별해보자 합니다.



- 어떻게 위와 같은 웹 페이지의 결과를 코드에서 활용할 수 있을까요?

## | 파이썬으로 웹 페이지에 있는 정보를 가져오는 방법

- 크게 세 가지 방법으로 가져올 수 있습니다.
  1. 누군가 업로드해 둔 데이터를 다운로드 받기 (ex. 캐글)
  2. 누군가 만들어 둔 API Server를 활용하여 정보를 받아오기
    - 아마, 깐풍기와 탕수육 API Server는 아무도 만들어 두지 않았을 거 같다
  3. 사람이 검색하는 것처럼 파이썬이 자동으로 검색 후 결과를 수집하는 방법
    - 이러한 기술을 **크롤링(Crawling)** 이라고 합니다.
    - 이번 프로젝트에서 사용할 기술입니다.

## Quiz

금융상품비교

영화추천서비스

- 데이터 사이언스에서는 먼저 데이터를 수집하는 것이 중요합니다.

Kaggle 같은 데이터 공유 플랫폼에서 수집된 데이터들을 쉽게 다운로드 받을 수 있지만

우리는 XXX라는 기술을 사용하여 직접 데이터를 수집하고자 합니다.

이 기술의 이름은 무엇인가요 ?

## 프로젝트 목표

금융상품비교

영화추천서비스

1. Django 없이, 크롤링 하는 방법 학습
2. 구글 검색 수를 크롤링하여 어떤 키워드가 더 많이 검색되는지 조사하기

## 웹 크롤링 이해하기

금융상품비교

영화추천서비스

## [복습] 데이터 사이언스 프로세스

- 필요한 정보를 추출하는 5가지 단계
  1. 문제 정의 : 해결하고자 하는 문제 정의
  2. 데이터 수집 : 문제 해결에 필요한 데이터 수집
  3. 데이터 전처리(정제) : 실질적인 분석을 수행하기 위해 데이터를 가공하는 단계
    - 수집한 데이터의 오류 제거(결측치, 이상치), 데이터 형식 변환 등
  4. 데이터 분석 : 전처리가 완료된 데이터에서 필요한 정보를 추출하는 단계
  5. 결과 해석 및 공유 : 의사 결정에 활용하기 위해 결과를 해석하고 시각화 후 공유하는 단계

금융상품비교

영화추천서비스

## [복습] 데이터 수집

- 데이터 수집은 다양한 기술과 방법을 활용할 수 있습니다.
  - 웹 스크래핑(Web Scraping): 웹 페이지에서 데이터를 추출하는 기술
  - 웹 크롤링(Web Crawling): 웹 페이지를 자동으로 탐색하고 데이터를 수집하는 기술
  - Open API 활용: 공개된 API를 통해 데이터를 수집
  - 데이터 공유 플랫폼 활용: 다양한 사용자가 데이터를 공유하고 활용할 수 있는 온라인 플랫폼
  - 종류: 캐글(Kaggle), Data.world, 데이콘(Dacon), 공공데이터포털 등

## 웹 크롤링이란?

금융상품비교

영화추천서비스

- 여러 웹 페이지를 돌아다니며 원하는 정보를 모으는 기술
- 원하는 정보를 추출하는 스크래핑(Scraping) 과 여러 웹 페이지를 자동으로 탐색하는 크롤링(Crawling)의 개념을 합쳐 웹 크롤링이라고 부름
- 즉, 웹 사이트들을 돌아다니며 필요한 데이터를 추출하여 활용할 수 있도록 자동화된 프로세스

## 웹 크롤링 프로세스

금융상품비교

영화추천서비스

- 웹 페이지 다운로드
  - 해당 웹 페이지의 HTML, CSS, JavaScript 등의 코드를 가져오는 단계
- 페이지 파싱
  - 다운로드 받은 코드를 분석하고 필요한 데이터를 추출하는 단계
- 링크 추출 및 다른 페이지 탐색
  - 다른 링크를 추출하고, 다음 단계로 이동하여 원하는 데이터를 추출하는 단계
- 데이터 추출 및 저장
  - 분석 및 시각화에 사용하기 위해 데이터를 처리하고 저장하는 단계

## 웹 크롤링 실습

## 준비 단계

금융상품비교

영화추천서비스

- 실습 및 도전 과제에는 구글 검색 결과 페이지를 크롤링합니다.
- 아래 필수 라이브러리를 설치 후 진행합니다.
  - **requests**: HTTP 요청을 보내고 응답을 받을 수 있는 모듈
  - **BeautifulSoup**: HTML 문서에서 원하는 데이터를 추출하는 데 사용되는 파이썬 라이브러리
  - **Selenium**: 웹 애플리케이션을 테스트하고 자동화하기 위한 파이썬 라이브러리
    - 웹 페이지의 동적인 컨텐츠를 가져오기 위해 사용함 (검색 결과 등)
- **\$ pip install requests beautifulsoup4 selenium**

## 기본 예제 실습

금융상품비교

영화추천서비스

- [Quotes to Scrape](#) 사이트를 활용합니다
  - 여러 가지 주제에 관한 명언들을 모아 둔 데모 사이트
- requests 및 BeautifulSoup 라이브러리 활용 연습
  - examples/example.py

# 구글 기본 예제

- examples/example1.py

```
from bs4 import BeautifulSoup
from selenium import webdriver

def get_google_data(keyword):
    url = f"https://www.google.com/search?q={keyword}"
    # 크롬 브라우저가 열린다. 이 때, 동적인 내용들이 모두 채워짐
    driver = webdriver.Chrome()
    driver.get(url)

    # 열린 페이지 소스를 받아옴
    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")

    # 눈으로 보기 좋게 출력
    print(soup.prettify())

    # 파일로 저장하여 확인하기
    with open('soup.txt', 'w', encoding="utf-8") as file:
        file.write(soup.prettify())

    # 검색 키워드 설정
    keyword = "파이썬"
    get_google_data(keyword)
```

- 실행 결과1. 파이썬을 검색 한 구글 창



- 실행 결과2. 엄청 나게 긴 페이지 코드 (분석 불가)

```
ed=1/dg=2;br=1/rs=ACT90oEgzTU-WoZSUnoAV0UvycR8HFVTsw/m=kMFpHd,sy2d,bm51tf?xjs=s3">
</script>
<script async="" nonce="" src="/_js/k=xjs.s.ko.VNESo4_-d7Q.O/cx=xjs.s.BHFJhKov9JI.L.W.O/
am=CggBIAAgGoRTABtAAPgnDAAAEBAAAAAAFACYEAgeP8JAQAAQEQMQQwAJBQAiYFAAdg9EMEGACAAGIACgAARQAcNAQq
AAIAAAAgfwDMeQAgwkLAAAAAAAAAAIYAmCwQSKAgAAQAAAAAAACAKpm8PCEAACAC/d=0/excm=A1Sy2b,ABxRVc,AD6
AIb,AOTkuc,CWp5c,CnT5wd,D1J6He,FXUdw,FmnE6b,FuQWyc,GRJ32c,HfxK9d,JxE93,KrUr5e,LtNDTb,MRb7nf,Mr
kcAd,Mxvwsd,NhUbHc,NmR9jd,NsEUGe,NzGbYd,Oa7Qpb,Ok4XMd,PoJj8d,SKZSKc,SLDaee,T00csb,U30vcc,U6n1Je
,UQpTU,UZNwo,UbcHRb,V9W1ad,WaSRUb,Wx0Z2d,WxJ6g,XH06qe,X0eh0c,XTkmZd,Xk0c,Y0dpFc,Yltq7c,ZrXR8b,Z
udxcb,a0nyD,bXPzdz,bXyZdf,cKV22c,dyUEmd,eTv59e,ee9Gld,f26on,hWJj1F,hfJ9hb,jkRPje,kOSi0d,llagHf,
mL4hG,pMwOEe,pQk1fc,pqUxUc,rL2AR,smKWJb,tzTB5,vJPfse,vPi79c,y25qZb,y6Ihab,yChgtb,yuQBec,z0Ft6e/
ed=1/dg=2;br=1/rs=ACT90oEgzTU-WoZSUnoAV0UvycR8HFVTsw/m=syk8,syk9,dt4g2b?xjs=s3">
</script>
</body>
</html>
```

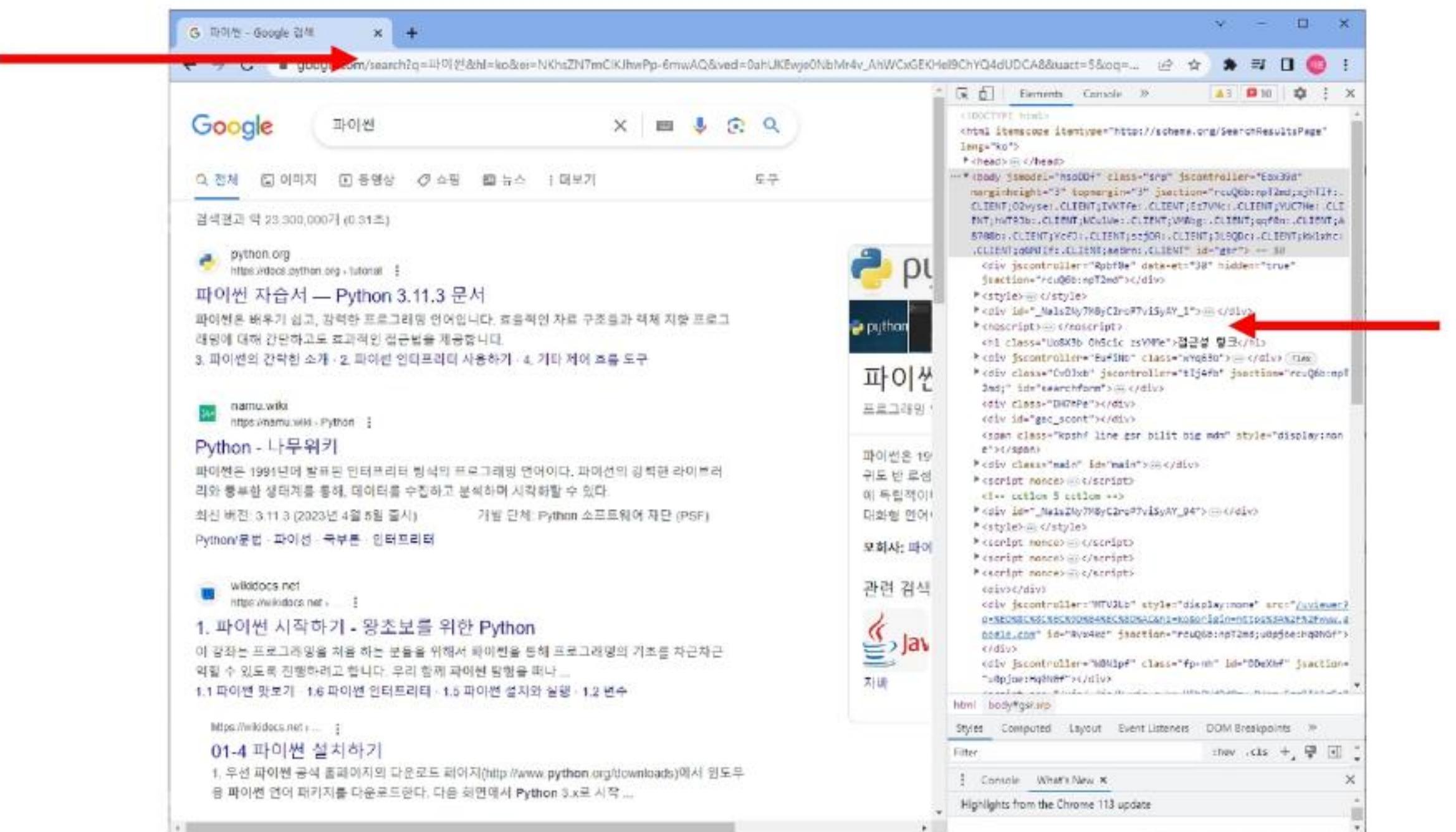
금융상품비교

영화추천서비스

## | 구글 검색 결과 분석하기(1/4)

- “F12” 혹은 “우측 클릭 - 검사”로 크롬 개발자 도구를 열어 활용합니다.

q=파이썬



### HTML, CSS, JavaScript 코드

- id 와 class 이름이 이상하다!**
- id 는 새로고침마다 변한다.**

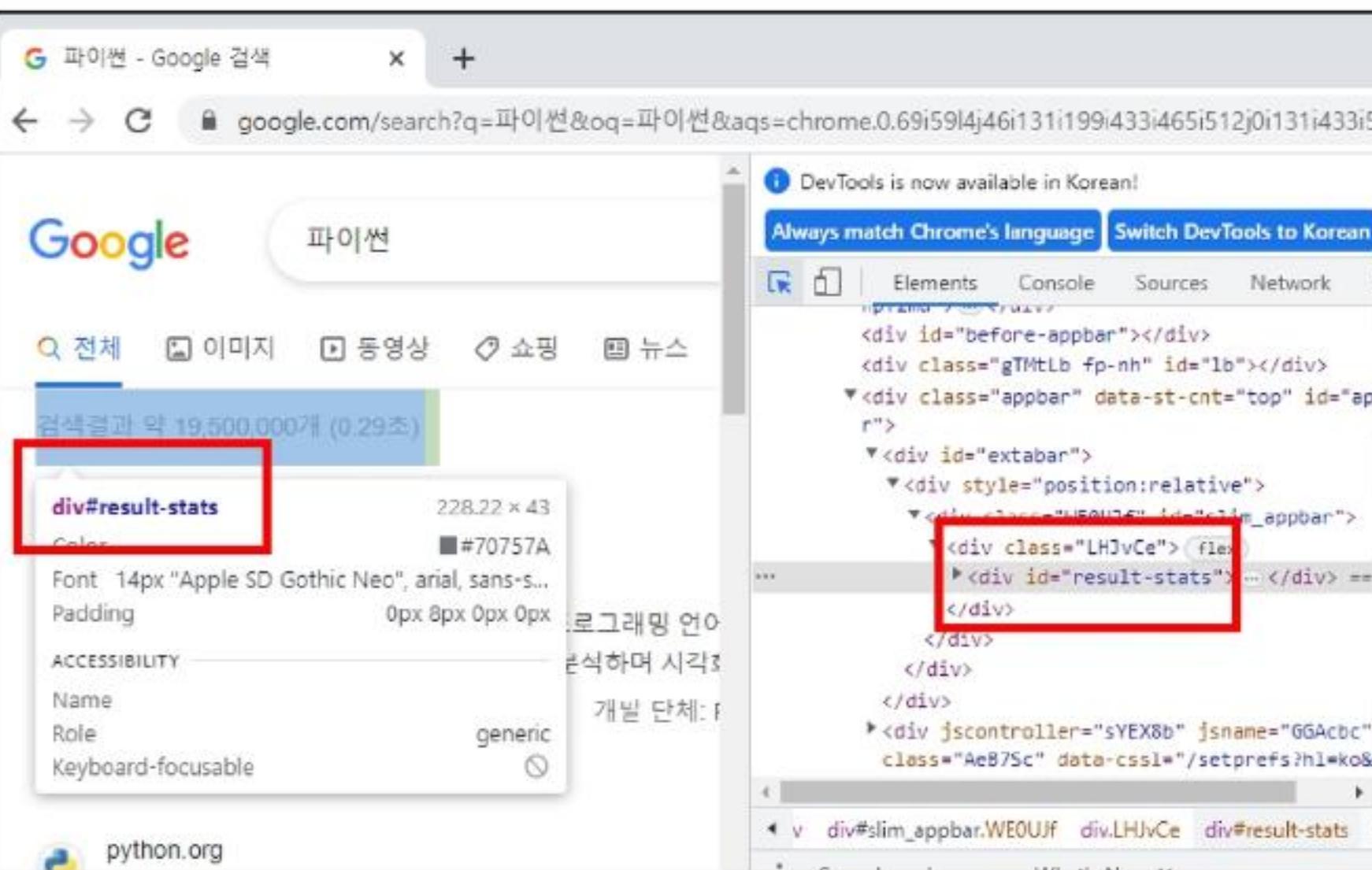
사람이 정하는 것이 아니라,

프로그래밍 되어 있다.

즉, id 값이 아닌 class 와 태그를 기준으로 정보를 추출해야 한다.

## | 구글 검색 결과 분석하기(2/4)

- 예시1. 검색 결과 개수 출력
- div 태그 이면서 id 가 “result-stats” 이다



```
from bs4 import BeautifulSoup
from selenium import webdriver

def get_google_data(keyword):
    url = f"https://www.google.com/search?q={keyword}"
    # 크롬 브라우저가 열린다. 이 때, 동적인 내용들이 모두 채워짐
    driver = webdriver.Chrome()
    driver.get(url)

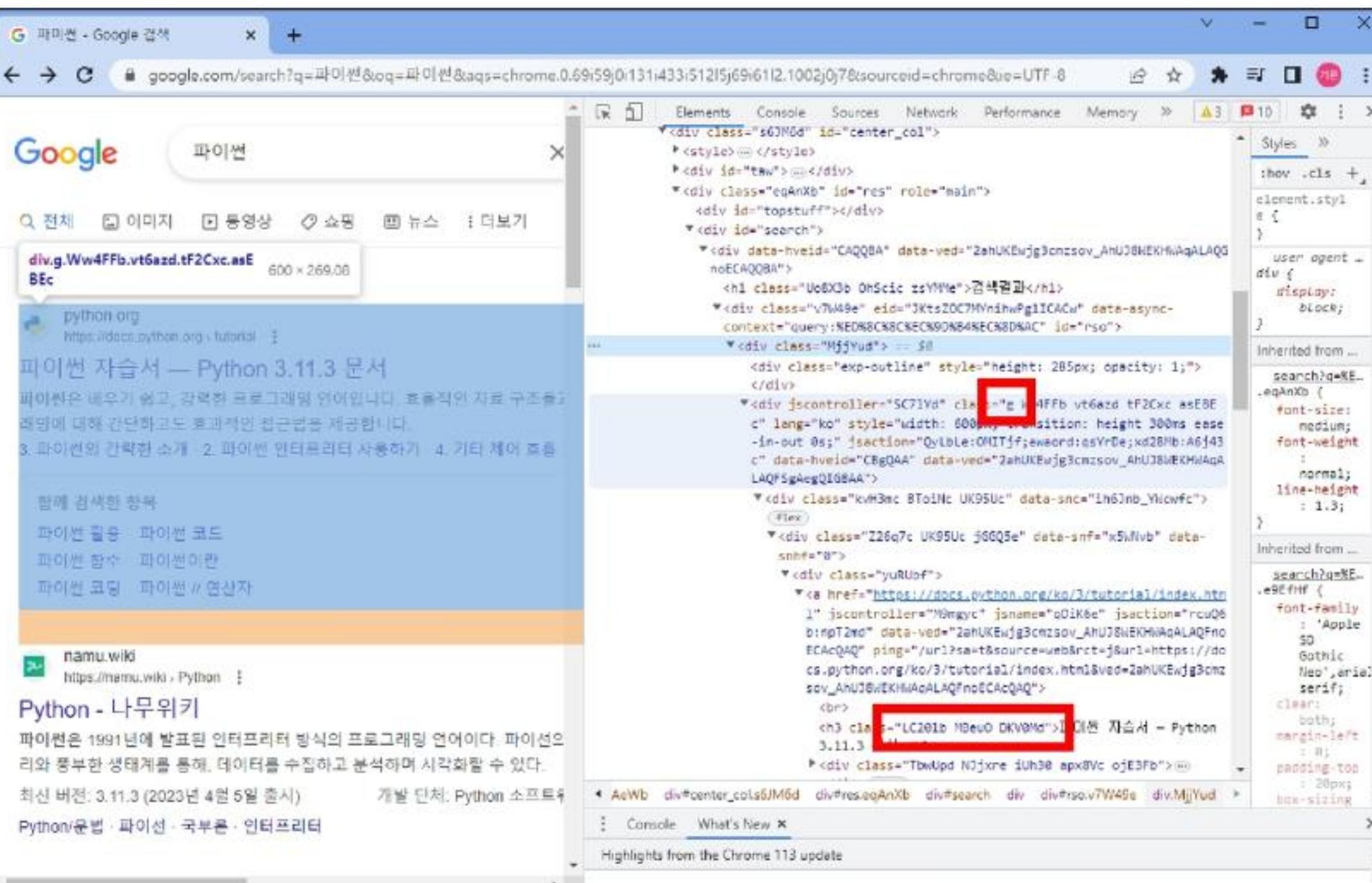
    # 열린 페이지 소스를 받아옴
    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")

    # div 태그 중 id 가 result-stats 인 요소 검색
    result_stats = soup.select_one("div#result-stats")
    print(result_stats)

    # 검색 키워드 설정
    keyword = "파이썬"
    get_google_data(keyword)
```

## | 구글 검색 결과 분석하기(3/4)

- 예시2. 검색 결과 페이지들의 제목 가져오기



공통적으로

결과를 감싸는 div에는 “g” 클래스

제목에는 “LC20lb MBeuO DKV0Md”

클래스를 가지고 있습니다.

## | 구글 검색 결과 분석하기(4/4)

- 예시2. 검색 결과 페이지들의 제목 가져오기
- example3.py

```
from bs4 import BeautifulSoup
from selenium import webdriver

def get_google_data(keyword):
    url = f"https://www.google.com/search?q={keyword}"
    # 크롬 브라우저가 열린다. 이 때, 동적인 내용들이 모두 채워짐
    driver = webdriver.Chrome()
    driver.get(url)

    # 열린 페이지 소스를 받아옴
    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")

    # div 태그 중 g 클래스를 가진 모든 요소 선택
    g_list = soup.select("div.g")
    # 해당 요소를 반복하며
    for g in g_list:
        # 요소 안에 LC20lb MBeuO DKV0Md 클래스를 가진 특정 요소 선택
        title = g.select_one(".LC20lb.MBeuO.DKV0Md")
        # 요소가 존재 한다면
        if title is not None:
            title_text = title.text
            print('제목 = ', title_text)

    # 검색 키워드 설정
    keyword = "파이썬"
    get_google_data(keyword)
```

### • 출력 결과

```
제목 = Python - 나무위키
제목 = 파이썬 자습서 – Python 3.11.3 문서
제목 = 1. 파이썬 시작하기 - 왕초보를 위한 Python
제목 = [Python] Python이란? - Maker's VAP - 티스토리
제목 = Python란 무엇인가요? - Python 언어 설명 - Amazon AWS
제목 = 파이썬 - 위키백과, 우리 모두의 백과사전
제목 = Python란 무엇인가요? - Python 언어 설명 - Amazon AWS
제목 = 최신 파이썬 코딩 무료 강의 - 5시간만 투자하면 개발자가 됩니다
제목 = 1 장 파이썬(Python) 입문 | 파이썬 프로그래밍 기초
제목 = 1) 파이썬 개요 - 코딩의 시작, TCP School
제목 = 파이썬 코딩을 시작하기 좋은 쉬운 아이디어들 - freeCodeCamp
```

## [참고] BeautifulSoup4 요소 선택 메서드 종류

- **find()**
  - 태그를 사용하여 요소를 검색. 첫 번째로 일치하는 요소를 반환
- **find\_all()**
  - 태그를 사용하여 요소를 검색. 모든 일치하는 요소를 리스트로 반환
- **select()**
  - CSS 선택자를 사용하여 요소를 검색. 모든 일치하는 요소를 리스트로 반환
- **select\_one()**
  - CSS 선택자를 사용하여 요소를 검색. 첫 번째로 일치하는 요소를 반환
- **find\_parent() / find\_next\_sibling() / find\_previous\_sibling()**
  - 태그를 사용하여 요소를 검색. 각각 일치하는 요소의 부모/다음 형제 요소/이전 형제 요소를 반환
- **공식문서 참고**

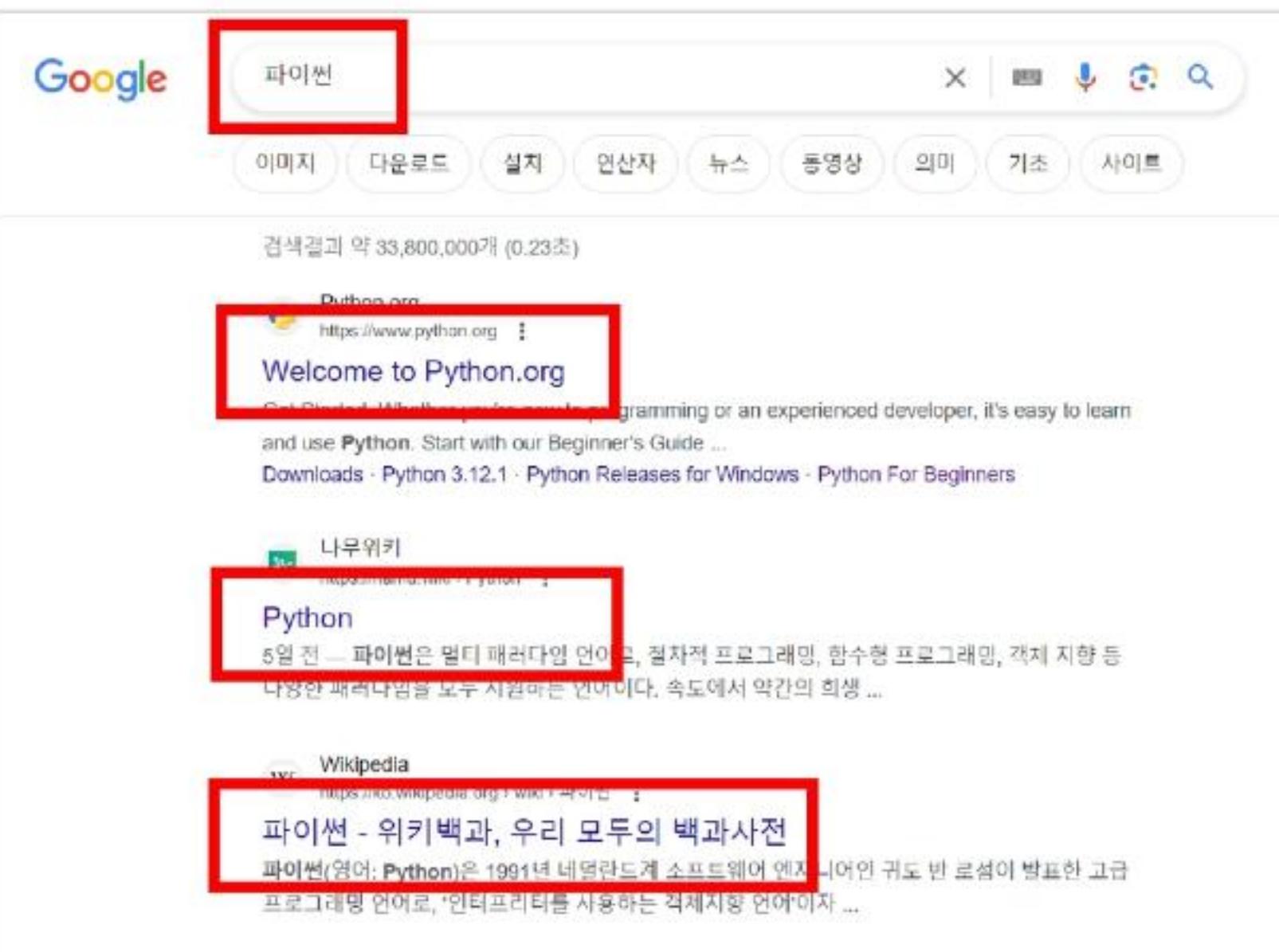
## Django 에서 활용하기

## 크롤링 결과 저장하기

금융상품비교

영화추천서비스

- Django에서 크롤링을 진행 후 아래 데이터를 데이터베이스에 저장합니다.
- 검색어와 게시글 제목을 DB에 저장합니다.



금융상품비교

영화추천서비스

## 크롤링 결과 저장하기

- 저장하고 싶은 데이터 : 검색어, 게시글 제목들
- models.py 를 어떻게 작성 해야할까 ?

금융상품비교

영화추천서비스

## | 유저 인증 기능 추가하기

1. 로그인
2. 로그아웃
3. 인증된 유저만 크롤링 결과를 볼 수 있도록 구성하기

## 도전 과제

# 금융 상품 비교 앱 PJT 05

## 관통 Ver1 - PJT05 도전 과제

금융상품비교

영화추천서비스

- **프로젝트명:** 키워드 검색량 분석을 위한 데이터 수집
- **목표**
  - 크롤링을 통한 데이터 수집
  - 수집한 데이터를 DB에 저장하고, 저장한 데이터 활용하기
- **특징**
  - 데이터 사이언스 패키지 사용
  - 수집한 데이터를 저장하고 활용할 수 있도록 DB 설계

## 영화 추천 서비스 PJT 05

금융상품비교

영화추천서비스

## 관통 Ver2 - PJT05 도전 과제

- **프로젝트명:** 인증 페이지 구현
- **목표**
  - 로그인, 로그아웃, 회원가입, 회원탈퇴, 회원정보수정, 비밀번호 변경이 가능한 애플리케이션 완성
- **특징**
  - 완성 화면 예시를 보며 기능 구현