8기 파이썬반 PJT - 6 (1)

Created	@2022년 10월 8일 오후 7:02
Last Edited Time	@2022년 10월 9일 오후 7:03
Type	
Status	
Created By	
Last Edited By	
Stakeholders	

1. 기초 설정

맨 처음, 가상환경부터 생성.

나는 홈디렉토리에 만드는 습관이 있다.

```
$ python -m venv ~/venv
$ source ~/venv/Scripts/activate
```

그리고 requrememts.txt 파일을 불러오자.

```
$ pip install -r ./requirements.txt
```

프로젝트와 필요한 앱 생성한다.

```
$ django-admin startproject project .
$ python manage.py startapp accounts
$ python manage.py startapp movies
```

일단

, `settings.py` 부터 건드린다.

```
# settings.py

INSTALLED_APPS = [
    'accounts',
    'movies',
]

TEMPLATES = [
    {
        'DIRS': [BASE_DIR / 'templates', ],
    }
]

LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'

AUTH_USER_MODEL = 'accounts.User'
```

다음, 전역 `urls.py` 설정하자.

```
# urls.py

from django.contrib import admin
from django.urls import path, include

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

이제 `accounts` 앱을 설정하자.

`accounts/models.py` 는 다음과 같다.

```
# accounts/models.py

from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    pass
```

`auth.models.AbstractUser` 를 상속받은 `User` 모델을 사용할 것이다. `pass` 라고 되어 있지만, 실제로 상속된 내용 모두가 들어가며, 이것이 커스텀유저 모델이 된다.

`accounts/admin.py` 는 다음과 같다.

```
# accounts/admin.py

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


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

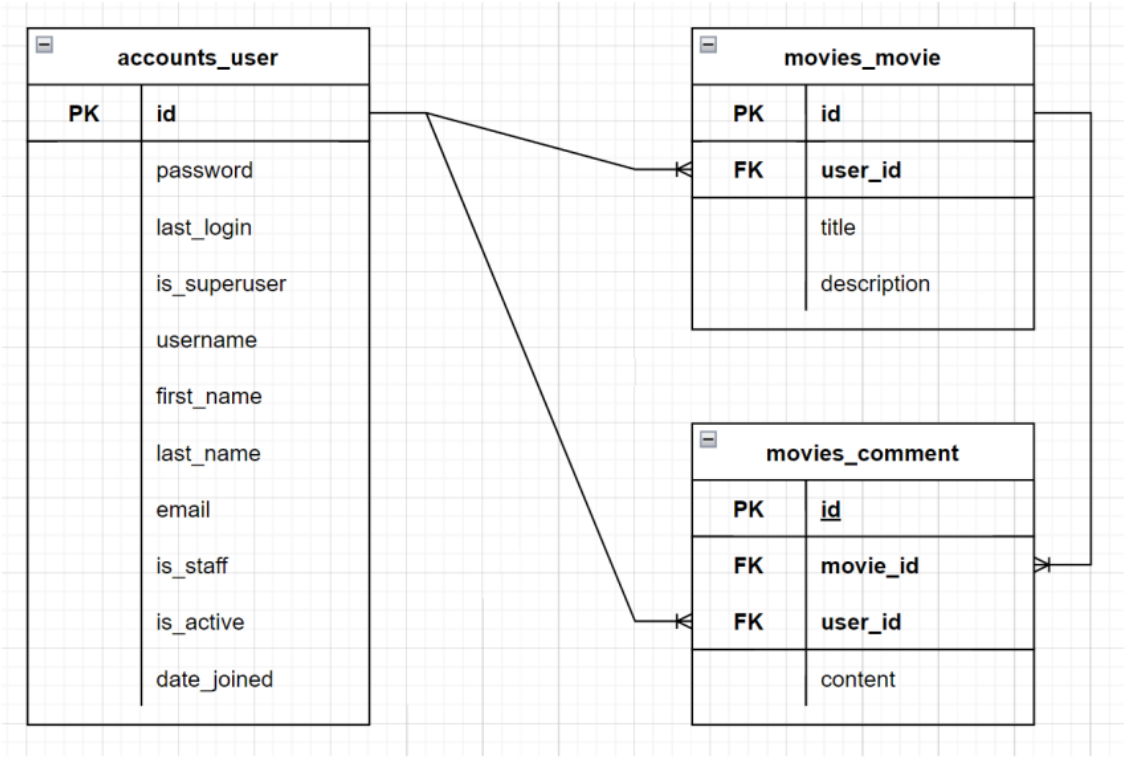
어드민 사이트에 커스텀유저 모델을 등록한다.

마지막으로, `accounts/urls.py` 는 다음과 같이 작성한다.

```
# accounts/urls.py

from django.urls import path
from . import views

app_name = 'accounts'

urlpatterns = []
```

다음, `movies` 앱을 설정하자.

우리가 구현할 ERD 는 다음과 같은 형태이다.



이 중, `accounts_user` 는 굉장히 복잡해보이지만 하나도 신경쓸 게 없다. 왜냐면, 저 필드들은 일일히 구현하는 게 아니라, 이미 Django 에서 제공하는 커스텀유저의 필드들이기 때문이다.

다만, `accounts_user`, `movies_movie`, `movies_comments` 의 관계가 중요하다. 모두 1 대 N 관계이며, PK 와 FK 의 위치를 고려했을 때 다음과 같은 결론이 나온다.

`movies/models.py` 는 다음과 같다.

```
# movies/models.py

from django.db import models
from django.conf import settings

class Movie(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    title = models.CharField(max_length=20)
    description = models.TextField()

class Comment(models.Model):
    movie = models.ForeignKey(Movie, on_delete=models.CASCADE)
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    content = models.CharField(max_length=100)
```

`movies/admin.py` 는 다음과 같다.

```
# movies/admin.py

from django.contrib import admin
from .models import Movie, Comment


admin.site.register(Movie)
admin.site.register(Comment)
```

자, 이제 사용자가 `localhost:8000/movies`로 접속했을 때 보일 간단한 index 창을 만들자.

`movies/urls.py` 를 생성한 후, 다음과 같이 작성한다.

```
# movies/urls.py

from django.urls import path
from . import views

app_name = 'movies'

urlpatterns = [
    path('', views.index, name='index'),
]
```

`movies/views.py` 는 다음과 같다.

```
# movies/views.py

from django.shortcuts import render

# Create your views here.
def index(request):
    return render(request, 'movies/index.html')
```

프로젝트 루트 경로 바로 아래에, `templates/base.html` 을 다음과 같이 작성한다.

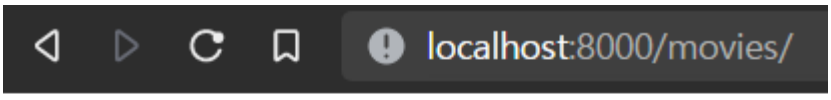
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div>
      {% block content %}
      {% endblock content %}
    </div>
  </body>
</html>
```

`movies/templates/movies/index.html` 을 생성하고, 다음과 같이 작성한다.

```
# movies/index.html
{% extends 'base.html' %}

{% block content %}
  <h1>INDEX</h1>
{% endblock %}
```

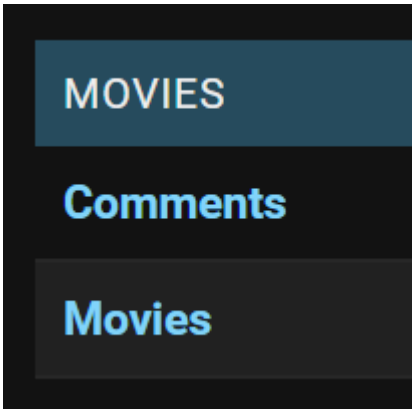
이제 마이그레이션 생성 및 마이그레이트, 관리자 생성 후, 테스트서버를 구동해보자.

```
$ python manage.py makemigrations
$ python manage.py migrate
$ python manage.py createsuperuser
$ python manage.py runserver
```



INDEX

`localhost:8000/movies` 로 접속 시, 다음과 같이 확인되어야 하며,



관리자 페이지에서 Comments 와 Movies 테이블이 확인되어야한다.

2. accounts

미리 `urls.py` 를 정의해두자.

다음 조건에 맞춰서 작성한다.

URL 패턴	역할
/accounts/login/	로그인 페이지 조회 & 세션 데이터 생성 및 저장 (로그인)
/accounts/logout/	세션 데이터 삭제 (로그아웃)
/accounts/signup/	회원 생성 페이지 조회 & 단일 회원 데이터 생성 (회원가입)
/accounts/delete/	단일 회원 데이터 삭제 (회원탈퇴)
/accounts/update/	회원 수정 페이지 조회 & 단일 회원 데이터 수정 (회원정보수정)
/accounts/password/	비밀번호 수정 페이지 조회 & 단일 비밀번호 데이터 수정 (비밀번호변경)

```
# accounts/urls.py

from django.urls import path
from . import views


app_name = 'accounts'

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

```
    path('signup/', views.signup, name='signup'),
    path('delete/', views.delete, name='delete'),
    path('update/', views.update, name='update'),
    path('password/', views.change_password, name='change_password'),
]
```

`views.py` 작성 전, `forms.py` 를 만들겠다.

```
# accounts/forms.py

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

class CustomUserCreationForm(UserCreationForm):
    class Meta:
        model = get_user_model()
        fields = ('username', 'email',)

class CustomUserChangeForm(UserChangeForm):
    class Meta:
        model = get_user_model()
        fields = ('email',)
```

`UserCreationForm` 과 `UserChangeForm` 을 각각 `CustomUserCreationForm` , `CustomUserChangeForm`으로 상속받아 오버라이딩할 것이다.

`get_user_model()` 을 사용해 Django 에서 제공하는 유저 모델을 그대로 가져다쓰고,

사용하고자 하는 필드만 적어준다. 유저 생성에선 `username` 과 `email` 만 가져다쓰겠다. 유저 수정은 `email` 만 바꾸도록 허용한다.

비밀번호 변경 폼은 Django 에서 기본적으로 제공하니 신경쓰지 않겠다.

`signup` 함수를 `views.py` 에 작성해보자.

```
# accounts/views.py

from django.shortcuts import render, redirect
from django.contrib.auth import login as auth_login
from django.views.decorators.http import require_http_methods
from .forms import CustomUserCreationForm

# Create your views here.
def login(request):
    pass
```

```
def logout(request):
    pass

@require_http_methods(['GET', 'POST'])
def signup(request):
    if request.user.is_authenticated:
        return redirect('movies:index')

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

def delete(request):
    pass

def update(request):
    pass

def change_password(request):
    pass
```

`urls.py` 를 미리 작성했기에, 에러 방지를 위해 미리 함수들을 만들어두었다.

사용자가 로그인한 상태라면 회원가입 페이지에 들어오지 못하고 `movie:index` 로 향한다.

로그인 안 한 상태일 경우라면 POST 일 경우와 GET 일 경우로 달라진다.

POST 는 `signup.html` 에서만 보낼 수 있다. 사용자가 폼을 다 작성 후에 `submit` 버튼을 눌러야만 동작하는 부분이다.

사용자가 입력한 폼이 양식에 맞는지(`is_valid`) 판단한 후, 맞으면 회원가입 후 로그인하고, `movies:index` 로 리다이렉트한다.

만약, 양식에 맞지 않다면 if 문은 실행되지 않고 곧바로 `context` 생성 부분으로 넘어가게 되어, 다시 `signup.html` 페이지가 보여지고, 사용자가 입력한 내용이 그대로 유지되도록 한다.

GET 은 회원가입 링크를 클릭했을 때 동작하는데, 코드 상에선 `else` 부분이다. 회원가입 폼을 받아서, `context` 를 만든 다음 사용자에게 보여주게된다.

회원가입 페이지를 보여주기 위해, 먼저 `base.html` 을 수정한다.

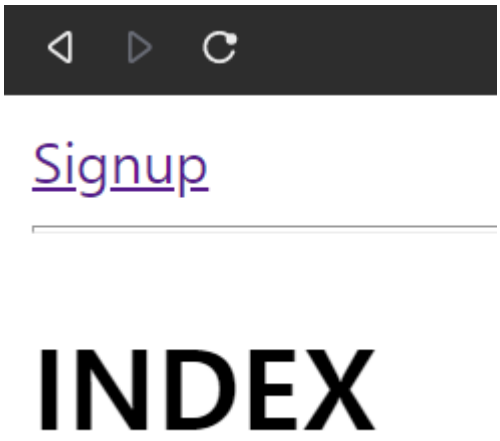
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <nav>
      {% if user.is_authenticated %}
        <h3>Hello, {{ user.username }}</h3>
      {% else %}
        <a href="{% url 'accounts:signup' %}">Signup</a>
      {% endif %}
    </nav>
    <hr>
    <div>
      {% block content %}
      {% endblock content %}
    </div>
  </body>
</html>
```

`<nav>` 태그 안에, 만약 사용자가 로그인한 상태라면 이름이 출력되고, 그렇지 않다면 회원가입 링크를 클릭할 수 있도록 만들었다.

다음, `accounts/templates/accounts/signup.html` 을 다음과 같이 작성한다.

```
# accounts/signup.html

{% extends 'base.html' %}

{% block content %}
  <h1>Signup</h1>
  <form action="{% url 'accounts:signup' %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
  </form>
{% endblock content %}
```

`forms.py` 에서 만든 폼을 `<p>` 태그 형식으로 사용할 것이다. POST 방식으로 보낼 것이기에 `csrf_token` 을 빼먹지 말자.



로그인 하지 않은 상태에선 다음과 같이 나온다.

기본 제공되는 폼이 확인된다. 먼저 유저를 1명 생성해보겠다.

Hello, jony123

INDEX

페이지가 리다이렉트 되면서, 회원가입 링크는 더이상 보이지 않고, “Hello, 유저이름” 이 잘 나타남이 확인된다.

이 사용자를 로그아웃시켜야하므로, 로그아웃 기능을 제작해보자.

먼저, `base.html` 의 `<nav>` 부분만 다음과 같이 수정한다.

```
<nav>
  {% if user.is_authenticated %}
    <h3>Hello, {{ user.username }}</h3>
    <form action="{% url 'accounts:logout' %}" method="POST">
      {% csrf_token %}
      <input type="submit" value="Logout" />
    </form>
  {% else %}
    <a href="{% url 'accounts:signup' %}">Signup</a>
  {% endif %}
</nav>
```

자세히보면, 로그아웃이 추가되었다. 당연히, 로그인이 된 상태에서만 보여야한다.

`views.py` 에 `logout` 함수를 다음과 같이 작성한다.

```
# accounts/views.py
from django.contrib.auth import logout as auth_logout
from django.views.decorators.http import require_http_methods, require_POST

@require_POST
def logout(request):
```

```
    if request.user.is_authenticated:
        auth_logout(request)
    return redirect('movies:index')
```

만약 사용자가 로그인된 상태라면 로그아웃 진행하고 `movies:index` 로 리다이렉트, 로그인 안한 상태라면 그냥 리다이렉트이다.

로그아웃이 생겼고,

클릭하면 정상적으로 로그아웃된다.

유저를 두 명 더 만들어보자.

다음, 회원 탈퇴를 만들어보자.

```
<nav>
  {% if user.is_authenticated %}
    <h3>Hello, {{ user.username }}</h3>
    <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>
  {% else %}
    <a href="{% url 'accounts:signup' %}">Signup</a>
  {% endif %}
</nav>
```

회원탈퇴를 담당하는 `<form>` 을 하나 더 추가하자.

`accounts/views.py` 의 `delete` 를 다음과 같이 작성한다.

```
# accounts/views.py

@require_POST
def delete(request):
    if request.user.is_authenticated:
        request.user.delete()
        auth_logout(request)
    return redirect('movies:index')
```

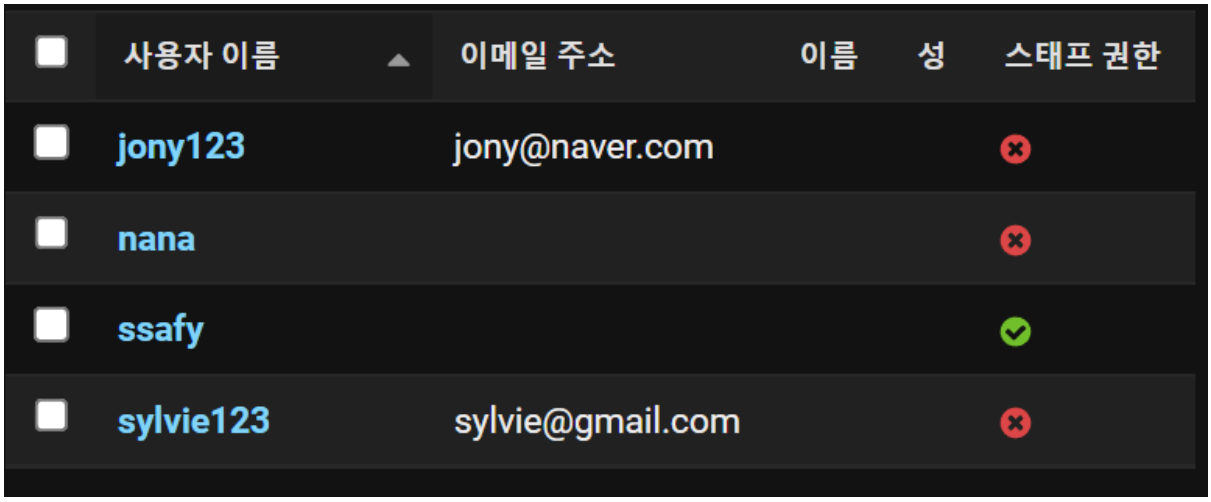
만약 로그인된 상태라면, 유저를 삭제하고 로그아웃한다.

한번 더 강조한다. 순서를 지켜야하는데, 유저 삭제한 후, 로그아웃해야한다.

이후 `movies:index` 로 리다이렉트한다.

로그인한 상태에서, 회원탈퇴 버튼이 보인다.

누르면, 다음 화면으로 바뀌고, 관리자페이지를 확인해보자.



zzzz 라는 사용자는 더는 존재하지 않음을 확인할 수 있다.

이제 로그인을 만들어보자. `base.html` 은 다음과 같이 수정한다.

```
<nav>
  {% if user.is_authenticated %}
    <h3>Hello, {{ user.username }}</h3>
    <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>
  {% else %}
    <a href="{% url 'accounts:login' %}">Login</a>
    <a href="{% url 'accounts:signup' %}">Signup</a>
  {% endif %}
</nav>
```

`else` 구문에서, 회원가입 바로 위에 로그인 링크를 걸었다.

`accounts/views.py` 에서 `login` 함수를 작성해보자.

```
# accounts.py

from django.contrib.auth.forms import AuthenticationForm

@require_http_methods(['GET', 'POST'])
```

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

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

`signup` 함수와 형태가 매우 비슷하니 자세한 설명은 생략한다.

다만, 여기선 로그인 폼을 사용하기 위해 Django 에서 기존 제공하는 `AuthenticationForm` 을 사용했고, `get_user` 함수를 사용해 로그인할 유저가 누구인지 가져왔다.

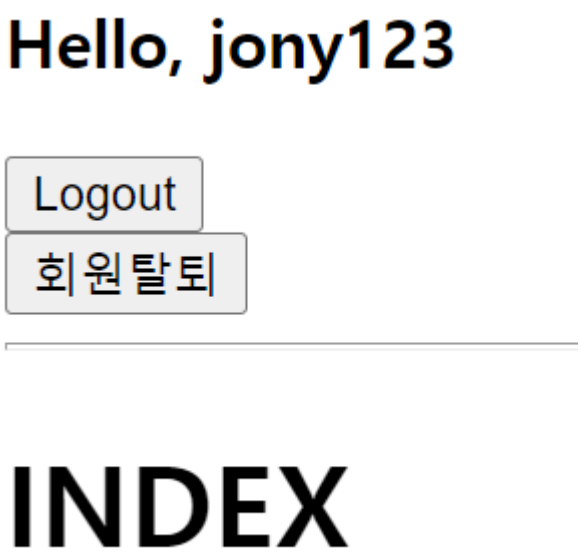
`accounts/login.html` 을 만들어보자.

```
{% extends 'base.html' %}

{% block content %}
  <h1>Login</h1>
  <form action="{% url 'accounts:login' %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
  </form>
{% endblock content %}
```

`signup.html` 과 구조상 큰 차이는 없다.

로그인이 생겼고,

로그인 페이지가 생겼다.



로그인 성공 시 화면.

나머지 두 명의 유저로도 로그인 시도해보자.

이제 남은 건 회원정보 수정이다.

`base.html` 로 향하자.

```
<nav>
  {% if user.is_authenticated %}
    <h3>Hello, {{ user.username }}</h3>
    <a href="{% url 'accounts:update' %}">회원정보수정</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>
  {% else %}
    <a href="{% url 'accounts:login' %}">Login</a>
    <a href="{% url 'accounts:signup' %}">Signup</a>
  {% endif %}
</nav>
```

`<h3>` 태그 바로 아래에 `<a>` 태그로, 회원정보수정으로 향하는 링크를 달았다.

`accounts/views.py` 의 `update` 와, `change_password` 를 다음과 같이 작성한다.

```
from django.contrib.auth.forms import AuthenticationForm, PasswordChangeForm
from django.contrib.auth.decorators import login_required
from django.contrib.auth import update_session_auth_hash
from .forms import CustomUserCreationForm, CustomUserChangeForm

@login_required
@require_http_methods(['GET', 'POST'])
def update(request):
    if request.method == 'POST':
        form = CustomUserChangeForm(request.POST, instance=request.user)
        if form.is_valid():
            form.save()
            return redirect('movies:index')
    else:
        form = CustomUserChangeForm(instance=request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/update.html', context)


@login_required
@require_http_methods(['GET', 'POST'])
def change_password(request):
    if request.method == 'POST':
        form = PasswordChangeForm(request.user, request.POST)
        if form.is_valid():
            form.save()
            update_session_auth_hash(request, form.user)
            return redirect('movies:index')
    else:
        form = PasswordChangeForm(request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/change_password.html', context)
```

`update` 에서, 현재 접속한 유저의 정보를 그대로 폼에 적용하기 위해 `instance=request.user` 를 인자로 추가했다.

`change_password` 에선 Django 에서 제공하는 `PasswordChangeForm` 을 가져와 사용한다. 특히 비밀번호는 해시값 처리를 해야하기 때문에, `update_session_auth_hash` 를 사용해 해시처리 해줬다.

`accounts/update.html` 는 다음과 같다.

```
{% extends 'base.html' %}
```

```
{% block content %}
  <h1>회원정보수정</h1>
  <form action="{% url 'accounts:update' %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
  </form>
{% endblock content %}
```

`accounts/change_password.html` 은 다음과 같다.

```
{% extends 'base.html' %}

{% block content %}
  <h1>비밀번호 변경</h1>
  <form action="{% url 'accounts:change_password' %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
  </form>
{% endblock content %}
```

여기서 변경할 수 있는 정보는 오로지 이메일로 한정된다. 비밀번호를 변경하려면 “다만 이 폼을 사용하여~” 에 적힌 “폼” 이라는 링크를 클릭하면 되는데, Django 에서는 이 문구에서 “폼” 의 링크를 `password` 로 걸어두었고, 이것은 `accounts/urls.py` 에 의해, `change_password` 함수를 실행시키게 된다.

비밀번호 변경이 되는지 테스트해보자.