Day9 Django Auth2 (회원가입,탈퇴,정보수정,비밀번호변경)

자료	Django
구분	Django
과목	

test33_skeleton.zip

accounts

accounts 앱을 통해서 회원들의 회원가입 및 탈퇴 / 로그인 로그아웃 / 회원정보 수정까지 먼저 구현 해 보자.

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'),
]
```

회원가입

`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` 은 회원가입시 사용자 입력을 받는 form인데 Django에서 제공해 주는 built in form 이다.
하지만 우리는 `CustomUserCreationForm` 을 사용해서 사용자 회원 가입 시 필요한 추가 정보를 수집할 것이다.
즉, 회원가입시 유저가 작성 할 form을 커스터마이징 할 것이다.
이때, `UserCreationForm` 과 `UserChangeForm` 을 각각 `CustomUserCreationForm` , `ustomUserChangeForm`
으로 상속받아 오버라이딩 한다.

`from django.contrib.auth import get_user_model`
에 대해서 살펴보자.
우리는 Django에서 제공해주는 User 모델을 커스터마이즈 해서 사용할 것이라고 했다. 그래서
accouts > models.py 에 커스터마이즈를 한 User 모델을 사용한다고 정의 했었다.

```
from django.db import models
from django.contrib.auth.models import AbstractUser
# Create your models here.

class User(AbstractUser):
    pass
```

따라서 회원 가입시에도 커스터마이즈 된 User 모델에 맞춰 `get_user_model()` 을 사용해서
커스터마이즈 된 폼으로 유저 생성을 도울 것이다.

Django 공식문서에 따르면,

프로젝트에서 커스터마이즈된 사용자 모델을 사용할 경우

유저 생성시 `CustomUserCreationForm` 을 사용한다면 get_user_model() 을 사용하는 것이

코드의 유연성과 유지보수에 유리함으로 `get_user_model()` 사용을 권장한다.

아래 코드를 다시 살펴보자.

```
# 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',)
```

유저 생성에서는 `username` 과 `email` 만 가져다가 사용하겠다.

유저 수정은 `email` 만 바꾸도록 허용 한다.

비밀번호 변경 폼은 Django 에서 기본적으로 제공됨으로 우리는 신경 쓰지 않아도 좋다.

회원가입 기능을 구현 해 줄

`signup` 함수를 `views.py` 에 추가로 작성해보자.

```
from django.views.decorators.http import require_http_methods
from .forms import CustomUserCreationForm

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

    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            auth_login(request, user)
            return redirect('todos:index')
    else:
```

```
        form = CustomUserCreationForm()
    context = {
        'form': form,
    }
    return render(request, 'accounts/signup.html', context)



# 우리는 `urls.py` 에 URL들을 미리 작성했기에, 에러 방지를 위해
# views.py에 아래와 같이 함수들도 미리 정의를 해 두고 pass를 넣어 두었다.

def delete(request):
    pass

def update(request):
    pass

def change_password(request):
    pass
```

Signup 함수에 작성된 코드를 살펴보자.

- if request.user.is_authenticated:

  return redirect('todos:index')

  사용자가 이미 로그인한 상태라면

  회원가입 페이지에 들어오지 못하고 `todos:index` 로 향하도록 했다.

로그인을 하지 않은 상태라면 POST 일 경우와 GET 일 경우로 달라진다.

POST 는 `signup.html` 으로 부터 데이터가 넘어 올 것이다.

- 회원가입 페이지에서 사용자가 회원가입에 필요한 폼을 모두 작성 후에 `submit` 버튼을 누르면 동작하는 부분이다.
- 사용자가 입력한 폼이 양식에 맞는지(`is_valid()`) 판단한 후, 맞으면 회원가입 후 로그인하고, `todos: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 링크 또는 버튼을 클릭했을 때 동작하는데, 코드 상에선 `else` 부분이다.

- 회원가입 폼을 `context` 로 만든 다음 사용자에게 보여주게 된다.

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

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Don't think, Just do it</title>
 </head>
 <body>

  <nav>
   {% if user.is_authenticated %}
   <a href="{% url 'todos:index' %}">[Main]</a>
   <a href="{% url 'todos:create_todo' %}">[Create]</a>
   <a href="{% url 'accounts:logout' %}">[Logout]</a>

   <h3>Hello, {{ user.username }}</h3>
   {% else %}
    <a href="{% url 'accounts:signup' %}">[Signup]</a>
    <a href="{% url 'accounts:login' %}">[Login]</a>
   {% endif %}
  </nav>
  <hr>
  {% block content %}

  {% endblock %}
 </body>
</html>
```

`<nav>` 태그 안에, 만약 유저가 로그인한 상태라면 유저의 이름이 출력 되고, [Main] [Create] [Logout] 링크가 보일 것이다.

만약 사용자가 로그인 상태가 아니라면 [Login] 또는 [Signup] 링크를 클릭할 수 있도록 만들었다.

다음, accounts 앱에서 `/templates/accounts/signup.html` 을 생성후 아래와 같이 작성한다.

```
{% 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 잊지말고 꼭 넣어주자.

서버를 다시 켜서 http://127.0.0.1:8000/todos/ 를 확인해 보자.

이제 사용자를 로그아웃 시킬 수 있도록 로그아웃 기능을 제작해보자.

이미 로그아웃 기능을 구현을 했지만 코드를 조금 업그레이드 시켜 보겠다.

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

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Don't think, Just do it</title>
  </head>
  <body>

    <nav>
      {% if user.is_authenticated %}
      <a href="{% url 'todos:index' %}">[Main]</a>
      <a href="{% url 'todos:create_todo' %}">[Create]</a>

      {% comment %} <a href="{% url 'accounts:logout' %}">[Logout]</a> {% endcomment %}

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

        <h3>Hello, {{ user.username }}</h3>

      {% else %}
        <a href="{% url 'accounts:signup' %}">[Signup]</a>
        <a href="{% url 'accounts:login' %}">[Login]</a>
      {% endif %}
    </nav>
    <hr>
    {% block content %}

    {% endblock %}
  </body>
</html>
```

a 태그로 링크형식으로 만들어 놓았던 logout 을 버튼 형식으로 바꾸었다.

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

```
from django.views.decorators.http import require_http_methods, require_POST
# 기존 코드에서 , require_POST 만 추가하자.

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

만약 사용자가 로그인된 상태라면 (권한이 있는 사용자라면)

로그아웃 진행하고 `todos:index` 로 리다이렉트 할 것이며

혹시 로그인을 하지 않은 상태라면 그냥 `todos:index` 로 리다이렉트 할 것이다.

- `@require_POST` 는 `@require_http_methods(['POST'])` 의 단축버전 이다.
- 오직 `POST` 요청만 허용하며, 다른 HTTP 메서드(GET, PUT, DELETE 등)는 **405 Method Not Allowed**를 반환하겠다는 뜻이다.

회원탈퇴

다음, 회원 탈퇴를 만들어보자.

이번 회원탈퇴도 링크가 아닌 버튼으로도 만들어 보겠다.

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

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Don't think, Just do it</title>
  </head>
  <body>

    <nav>
      {% if user.is_authenticated %}

          ...

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

      <h3>Hello, {{ user.username }}</h3>
```

```
    ...
```

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

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

만약 로그인된 상태라면, 유저를 삭제하고 로그아웃한다.

여기서 주의를 할 점이 있다.

코드를 구현할 때 순서가 중요하다.

데이터베이스에 데이터로 남아있는 유저 삭제한 후에 로그아웃 해야 한다. 로그아웃을 먼저 해 버리면

어떠한 유저를 회원탈퇴를 해야 할지 모를 것이기 때문이다.

구현하면서 실수를 많이 하는 부분이라 다시 강조하자면, `request.user.delete()`

를 통해서 먼저 회원삭제를 한 후에 `auth_logout` 을 통해서 로그아웃을 진행을 한다.

탈퇴를 한 후에는 `todos:index` 로 리다이렉트 한다.

지금까지 회원가입 > 로그아웃 > 회원탈퇴 순으로 구현해 보았다.

이제 로그인을 만들어보자. 우리는 앞서 로그인 기능도 이미 구현을 다 했었다.

그러나 살짝 업그레이드 시켜보자. `accounts/views.py` 에서 `login` 함수를 작성해보자.

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

    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        if form.is_valid():
            auth_login(request, form.get_user())
            return redirect('todos:index')
    else:
        form = AuthenticationForm()
    context = {
        'form': form,
```

```
    }
    return render(request, 'accounts/login.html', context)
```

기존에 있는 코드에서 http method 데코레이터를 추가하고

이리 로그인이 된 유저가 로그인 요청을 보냈을 경우 index 페이지로 redirect 하는 코드만 추가 했다.

`accounts/login.html` 에 base.html 상속받는 코드로 수정 할 것이 있는지 보자.

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

{% block content %}
  <h1>로그인 페이지</h1>
  <form action="{% url 'accounts:login' %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="login" />
  </form>
{% endblock %}
```

기존 코드랑 다른점이 없다.!!

이렇게 로그인 로그아웃

서버 켜서 확인해 보자.

회원가입 > 로그아웃 > 회원탈퇴 > 회원가입 기능을 다 구현 했으니 서버를 켜서 확인해보자.

새로 회원가입도 해보고 로그아웃도 해보자. 그리고

다시 로그인 해보고 게시글도 작성해보고 회원탈퇴 버튼도 눌러보자.

이상 없이 잘 되었다면 회원정보를 수정 할 수 있도록 코드를 구현해 보자.

회원정보 수정

`base.html` 로 가자. “정보수정” form하나 추가하자.

```
    ...

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

  <form action="{% url 'accounts:update' %}" method="GET">
    {% csrf_token %}
    <input type="submit" value="정보수정">
```

```
    </form>


    <h3>Hello, {{ user.username }}</h3>
    ...
```

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

```
from django.contrib.auth.decorators import login_required
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('todos:index')
    else:
        form = CustomUserChangeForm(instance=request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/update.html', context)
```

새롭게 추가 된 내용은 import login_required 데코레이터 이다.

그리고 회원정보 수정을 하기 위한 form을 제공받기 위해서 CustomUserChangeForm 을 import 했다.

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

`accounts/update.html` 파일을 새로 생성후 아래와 같이 작성해보자.

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

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

```
  </form>
{% endblock content %}
```

비밀번호 수정

마지막으로 비밀번호 수정 해보자.

프로젝트폴더 (todo_list_pjt) 의 `urls.py` 로 이동하자.

그리고 경로와 import 문을 추가하자.

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from accounts import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('todos/', include('todos.urls')),
    path('accounts/', include('accounts.urls')),
    path('<int:user_pk>/password/', views.change_password, name='change_password')
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

`accounts/views.py` 의 `change_password` 함수를 다음과 같이 작성한다.

```
from django.contrib.auth.forms import AuthenticationForm, PasswordChangeForm
from django.contrib.auth import update_session_auth_hash

@login_required
@require_http_methods(['GET', 'POST'])
def change_password(request,user_pk):
    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('todos:index')
    else:
        form = PasswordChangeForm(request.user)
    context = {
        'form': form,
    }
    return render(request, 'accounts/change_password.html', context)
```

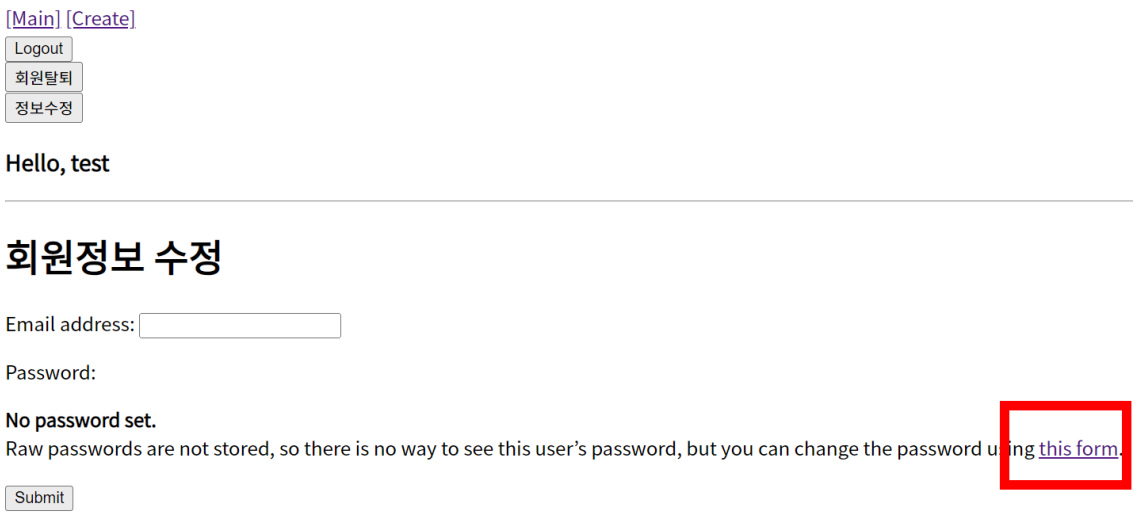
`change_password` 함수에서는 Django 에서 제공하는 `PasswordChangeForm` 을 import 해서 사용한다.

특히 비밀번호는 해시값 처리를 해야 하기 때문에, `update_session_auth_hash` 를 사용해 해시처리 해 두었다. 또한 암호 변경시 로그인 상태가 풀리는 세션 무효화를 막아주는 역할도 한다.

마지막으로, `accounts/change_password.html` 을 생성하자. 코드는 다음과 같다.

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

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



빨간 네모 박스를 클릭해서 비밀번호를 변경해보자.

<끝>

test33_completed.zip