190527

인증

장고는 기본적으로 user모델을 가지고 있음 -> 갖다 쓰기만 하면 됨

회원가입, 로그인 등의 기능을 만들어보자!

1. app 만들기

1. `$ django-admin startapp accounts`
2. settings.py - 앱 등록
3. urls.py

```
app_name='accounts'

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

2. 회원가입

user를 생성하는 로직.

UserCreationForm : 장고가 가지고 있는 유저 생성 폼

[views.py]

```
...
from django.contrib.auth.forms import UserCreationForm
...

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

[signup.html]

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

{% block body %}
{% load crispy_forms_tags %}

    <h1>회원가입</h1>
```

```
    <form action='' method='POST'>
        {% csrf_token %}
        {{ form|crispy }}

        <input type="submit" value="Submit"/>
    </form>

{% endblock %}
```

3. 로그인

session을 create 하는 행위.

AuthenticationForm : 장고가 지원하는 인증 폼

[views.py]

```
...
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth import login as auth_login
...

def login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST) # 얘는 요청 관련 정보도 알아야함 ->
request 도 넣어줌
        if form.is_valid() :
            # form.get_user : 입력된 user 정보 (id, pw 등)
            auth_login(request, form.get_user())
            return redirect('boards:index')
    else:
        form = AuthenticationForm()
    return render(request, 'accounts/login.html', {'form': form})
```

[login.html]

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

{% block body %}
{% load crispy_forms_tags %}

    <h1>로그인</h1>
    <form action='' method='POST'>
        {% csrf_token %}
        {{ form|crispy }}

        <input type="submit" value="Submit"/>
    </form>

{% endblock %}
```

[base.html]

```
...
    <div class="container">
        <h1>안녕, {{ user.username }}</h1> # 현재 로그인한 유저 보여줌
        <hr>
        {% block body %}
        {% endblock %}
    </div>
...
```

3. 로그아웃

session을 delete 하는 행위.

[views.py]

```
...
from django.contrib.auth import logout as auth_logout
...

def logout(request):
    if request.method=='POST':
        auth_logout(request)
        return redirect('boards:index')
    redirect('boards:index')
```

로그인/로그아웃 버튼 만들기

[base.html]

```
    <div class="container">
        {% if user.is_authenticated %}
            <h1>안녕, {{ user.username }}</h1>
            <form action="{% url 'accounts:logout' %}" method='POST'>
                {% csrf_token %}
                <input type="submit" value="로그아웃"/>
            </form>
        {% else %}
            <h1>
                <a href="{% url 'accounts:signup' %}">[회원가입]</a>
                <a href="{% url 'accounts:login' %}">[로그인]</a>
            </h1>
        {% endif %}
        <hr>
        {% block body %}
        {% endblock %}
    </div>
```

4. 로그인 여부에 따라 권한 다르게 부여

[boards/index.html]

```
...
    {% if user.is_authenticated %}
        <a href="{% url 'boards:new' %}">[새 글 작성]</a>
    {% else %}
        <a href="{% url 'accounts:login' %}">새 글을 작성하려면 로그인을 해주세요.</a>
    {% endif %}
...
```

5. 회원가입과 동시에 로그인

[views.py]

```
def signup(request):
    ...
        if form.is_valid():
            user = form.save() # save 된 user를 받음
            auth_login(request, user) # request와 유저정보 넣기
            return redirect('boards:index') # 다른 app으로 redirect 가능
    ...
```

6. 로그인 한 유저 회원가입/로그인 막기

[views.py]

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

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

        ...
```

7. 회원탈퇴

유저정보를 db에서 삭제

[views.py]

```
def delete(request):
    if request.method=='POST':
        request.user.delete()
    return redirect('boards:index')
```

[base.html]

```
{% if user.is_authenticated %}
<h1>안녕, {{ user.username }}</h1>
<form action="{% url 'accounts:delete' %}" method='POST' style='display:inline'>
    {% csrf_token %}
    <input type="submit" value="회원탈퇴"/>
</form>
...
```

8. 회원정보 수정

UserChangeForm : 장고에서 지원하는 유저 수정 폼

[views.py]

```
def edit(request):
    if request.method=='POST':
        form = UserChangeForm(request.POST, instance=request.user) # param1: 입력된 정보,
params2: 기존 정보
        if form.is_valid():
            form.save()
            return redirect('boards:index')
    else:
        form = UserChangeForm(instance=request.user)
    return render(request, 'accounts/edit.html', {'form': form})
```

[edit.html]

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

{% block body %}
{% load crispy_forms_tags %}

    <h1>회원정보 수정</h1>
    <form action='' method='POST'>
        {% csrf_token %}
        {{ form|crispy }}

        <input type="submit" value="Submit"/>
    </form>

{% endblock %}
```

[base.html]

```
...
{% if user.is_authenticated %}
<h1>안녕, {{ user.username }}</h1>
<a href="{% url 'accounts:edit' %}">회원정보 수정</a>
...
```

9. 커스텀 폼 만들기

기존 UserChangeForm이 너무 TMI 많이 알려줘서 깔끔하게 커스텀해서 쓸것임

[forms.py]

```
from django.contrib.auth.forms import UserChangeForm
from django.contrib.auth import get_user_model

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

커스텀 적용

[views.py]

```
def edit(request):
    if request.method=='POST':
        form = UserCustomChangeForm(request.POST, instance=request.user) # form 바꿔줌!
        if form.is_valid():
            form.save()
            return redirect('boards:index')
    else:
        form = UserCustomChangeForm(instance=request.user) # form 바꿔줌!
    return render(request, 'accounts/edit.html', {'form': form})
```

10. 비밀번호 변경

PasswordChangeForm : 장고가 가지고 있는 패스워드 변경 폼

[views.py]

```
def change_password(request):
    if request.method=='POST':
        form = PasswordChangeForm(request.user, request.POST) # param1: 요청 유저, param2: 입력
된 정보
        if form.is_valid():
            form.save()
            return redirect('boards:index')
    else:
        form = PasswordChangeForm(request.user)
    return render(request, 'accounts/change_password.html', {'form' : form})
```

[change_password.html]

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

{% block body %}
{% load crispy_forms_tags %}

    <h1>비밀번호 변경</h1>
    <form action='' method='POST'>
        {% csrf_token %}
        {{ form|crispy }}

        <input type="submit" value="Submit"/>
    </form>

{% endblock %}
```

[base.html]

```
...
<div class="container">
    {% if user.is_authenticated %}
    <h1>안녕, {{ user.username }}</h1>
    <a href="{% url 'accounts:edit' %}">회원정보 수정</a>
    <a href="{% url 'accounts:change_password' %}">비밀번호 변경</a>
...
```

11. 세션 유지

비밀번호 변경 시, 세션이 변경되면서 로그인이 풀려버림

-> 세션을 업데이트 해서 유지해주자!

[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) # param1: 요청 유저, param2: 입력
된 정보
        if form.is_valid():
```

```
            user = form.save()
            update_session_auth_hash(request, user)
            return redirect('boards:index')
    else:
        form = PasswordChangeForm(request.user)
    return render(request, 'accounts/change_password.html', {'form' : form})
```

12. 로그인 유저만 C,U,D 권한 주기

login_require 데코레이터를 활용. (데코레이터: 그 함수가 실행되기 전에 실행되는 함수)

[boards/views.py]

```
...
from django.contrib.auth.decorators import login_required
...

@login_required
def new(request):
    ...

@login_required
def update(request, pk):
    ...
```

13. 템플릿 하나로 합치기

지금까지 만든 login, signup, edit, change_password 는 h1만 다르고 전부 동일

-> 템플릿 합쳐버리자!

[auth_form.html]

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

{% block body %}
{% load crispy_forms_tags %}

{% if request.resolver_match.url_name == 'signup' %}
    <h1>회원가입</h1>
{% elif request.resolver_match.url_name == 'edit' %}
    <h1>회원정보 수정</h1>
{% elif request.resolver_match.url_name == 'change_password' %}
    <h1>비밀번호 변경</h1>
{% else %}
    <h1>로그인</h1>
{% endif %}

    <form action='' method='POST'>
        {% csrf_token %}
        {{ form|crispy }}

        <input type="submit" value="Submit"/>
    </form>
```

```
{% endblock %}
```

[views.py]

로그인, 회원가입 등의 렌더를 auth_form.html로 바꿔주기.