




회원가입 & 로그인

SignUp & SignIn



2020. 05. 07
강의 : 이민주



00. 오늘의 목표

1. 회원가입 기능을 구현해보자!
2. 간단한 회원가입 form 커스터마이징을 진행해보자!
3. 로그인기능을 구현해보자!

00. 사전 체크



이전에 로그인하던 계정은 무엇인가요??

지금까지 로그인했던 계정은 일반 회원 계정이 아니라, 관리자(SuperUser) 계정입니다.
게시판을 이용하는 모든 사람들에게 관리자 권한을 줄 수는 없으니, 계정을 만들 수 있게 해 주어야겠죠?



그렇다면 회원가입 & 로그인을 위해 장고가 제공하는 기능들은 없나요?

물론 있습니다! 장고는 게시판에 특화된 프레임워크니까요.
장고에서 사전에 제공하는 기능들은 다음과 같습니다.

1. User Model
: 장고에서 기본 유저모델을 제공합니다. 물론 나중에 원하는대로 커스터마이징도 가능합니다!
2. Login/Logout 을 위한 view
: 회원가입을 위한 view까지 제공하지는 않지만, 로그인과 로그아웃을 위한 view는 장고에서 제공합니다.
3. Form
: Form은 장고에서 UserCreateForm으로 제공하고, 이를 활용하여 커스터마이징도 가능합니다.

위에 제공되는 기본 기능들을 제외한 기능들은 우리가 직접 만들어줘야 합니다!
그리고 커스터마이징같은 경우 워낙 다양한 사례들이 있어 지금 알려드릴 수는 없지만,
커스터마이징을 하실 거라면 무조건 프로젝트를 시작하자마자 다른것들 하기 전에 해주셔야 합니다!

> 01. 회원가입

1) Accounts App 세팅하기

계정 관리를 위한 accounts 라는 App을 생성해줍니다.

01 가상환경 켜기 -> ud_session 폴더로 들어가기(cd ud_session)

02 app 생성하기

`python manage.py startapp accounts`

맥의 경우 python3 manage.py ~~ 로 해주세요!

APP 이름

Settings.py

03 Settings.py 에도 잊지 말고 App을 추가해주세요!

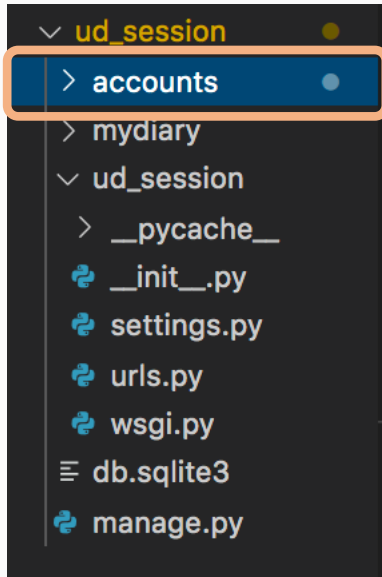
장고에 내장되어 있는 기능들을 활용하기 위해서
다음과 같이 App을 추가해줍니다.
내장 view를 사용할 것이기 때문에 다음과 같이 적어줄게요!

```
33 INSTALLED_APPS = [  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles',  
40     'mydiary',  
41     'accounts.apps.AccountsConfig',  
42 ]  
43  
44
```

01. 회원가입

2) Templates 만들어주기

회원가입 정보를 입력하고 가입을 진행할 수 있는 Templates 을 Account 앱 안에 만들어줍니다.



이번에는 기존에 사용하던 'mydiary' 앱이 아닌, 방금 새로 만들어준 'accounts' 앱 안에서 작업을 할 겁니다.

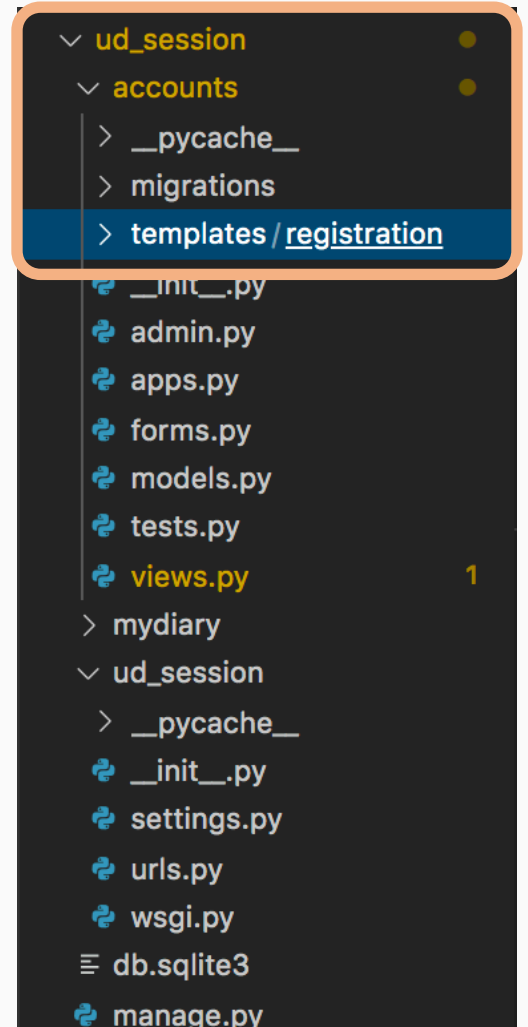
예전에 mydiary에서 했던 것과 같이, accounts 앱 안에 template 폴더를 만들어주세요! 그리고 그 안에 registration이라는 폴더를 만들어주고, 그 안에 register.html 이라는 html 파일을 만들어줍니다.

폴더구조 : accounts>templates>registration>register.html



잠깐! 왜 저번처럼 accounts>templates>accounts 가 아니라 accounts>templates>registration 로 설정하냐구요?

왜냐하면, 내장되어있는 로그인 view의 폴더주소가 기본적으로 registration으로 설정되어있기 때문입니다.





01. 회원가입

2) Templates 만들어주기

회원가입 정보를 입력하고 가입을 진행할 수 있는 Templates 을 Account 앱 안에 만들어줍니다.

register.html

```
ud_session > accounts > templates > accounts > <> register.html > form > form
1  <h1>회원가입</h1>
2  <form method="POST">
3      {% csrf_token %}
4      {{ form.as_p }}
5      <button type="submit">Sign Up</button>/
6  </form>
```

- 1 : 회원가입 글 태그
- 2 : URL로 데이터가 전송되지 않는 POST 방식으로 데이터를 전달해줍니다.
- 3 : csrf 공격을 막아주는, 보안을 위한 코드입니다.
form 태그 안에 적어주어야 form을 submit 할 때 함께 전송됩니다.
- 4 : form을 넣어주는 html 문법
- 5 : 제출 속성을 가진 버튼 태그(버튼 내용 : Sign Up)

그리고 mydiary 앱 안에 있는 home.html 로 가서, register.html 로 이동할 수 있는 링크를 만들어줍니다.

```
ud_session > mydiary > templates > mydiary > <> home.html > ...
1  <h1>아기사자의 다이어리입니다.</h1>
2
3  <a href="{% url 'new' %}">일기 작성하기</a><br><br>
4  <a href="{% url 'register' %}">회원가입</a><br><br>
5
6  {% for post in posts_list %}
7      <a href="{% url 'detail' index=post.pk %}">{{ post.title }}</a><br>
8      <hr>
9  {% endfor %}
10
```

home.html

01. 회원가입

3) View 작성하기

회원가입을 위한 view는 장고에서 제공해주지 않기 때문에, 직접 코딩해서 만들어줍니다!
역시 mydiary가 아닌 **accounts 앱 안에 있는 view에 입력해주세요!!!**

Views.py

```
ud_session > accounts > views.py > ...
1  from django.shortcuts import render, redirect
2  from django.utils import timezone
3  from django.contrib.auth.forms import UserCreationForm
4
5  # Create your views here.
6  def register(request):
7      if request.method == 'POST':
8          form = UserCreationForm(request.POST)
9          if form.is_valid():
10             form.save()
11             username = form.cleaned_data.get('username')
12             return redirect('home')
13      else:
14          form = UserCreationForm()
15      return render(request, 'registration/register.html', {'form': form})
16
```

회원가입 View 작동 원리

- 1 : 장고에 내재된(미리 만들어져 있는) render, redirect 패키지를 사용
- 2 : 장고에 내재된 (미리 만들어져 있는) timezone 패키지를 사용
- 3 : 장고에 내재된 (미리 만들어져 있는) 회원가입 Form을 사용(ID/PW 사용)
- 6 : register이라는 이름의 함수를 만듦(해당 함수는 는 요청을 받으면 작동됨)
- 7 : register.html에서 Submit 버튼을 누르면 POST 방식으로 접근
- 9 : form이 제대로 다 차 있는지 확인해보고 맞다면
- 10 : form에 들어온 정보를 DB에 저장
- 11 : Username 을 GET 방식으로 받아오고
- 12 : (제대로 저장 잘 되었으면) 'home' 템플릿으로 돌아간다.
- 13: 만일 요청받는 방식이 'POST' 방식이 아니라면 = GET 이면,
form에는 원래 기본 UserCreationForm을 담아서 보낸다. **인덴트 주의
- 15: (POST 방식으로 들어가면, 모두 redirect로 빠질 것이기 때문에
사실상 GET 방식으로 왔을 때 적용되는 루트)
'form'에 form을 담아 Register.html로 요청을 돌려준다. <key, value> 형태.

01. 회원가입

4) Url 연결하기

마지막으로 view와 Templates을 연결해줄 url을 작성해줍니다.

url은 ud_session의 하위 항목에 위치해 있습니다!!! 여기에 입력해주세요..

urls.py

```
17 from django.contrib import admin
18 from django.urls import path, include
19 import mydiary.views
20 import accounts.views
21
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('', mydiary.views.home, name="home"),
26     path('new/', mydiary.views.new, name="new"),
27     path('detail/<int:index>', mydiary.views.detail, name="detail"),
28     path('edit/<int:index>', mydiary.views.edit, name="edit"),
29     path('detail/<int:pk>/delete', mydiary.views.delete, name="delete"),
30     path('registration/register/', accounts.views.register, name="register"),
31 ]
```

127.0.0.1

회원가입

사용자 이름: 150자 이하 문자, 숫자 그리고 @/./+/_만 가능합니다.

비밀번호:

- 다른 개인정보와 비슷한 비밀번호는 사용할 수 없습니다.
- 비밀번호는 최소 8자 이상이어야 합니다.
- 비밀번호는 일상적으로 사용되는 비밀번호일 수 없습니다.
- 비밀번호는 전부 숫자로 할 수 없습니다.
- 비밀번호가 너무 짧습니다. 최소 8 문자를 포함해야 합니다.
- 비밀번호가 너무 일상적인 단어입니다.
- 비밀번호가 전부 숫자로 되어 있습니다.

비밀번호 확인: 확인을 위해 이전과 동일한 비밀번호를 입력하세요.

/

그럼 이제 서버를 돌려봅시다!

이제 서버 돌리는 방법은 다들 아시죠?

서버 돌리고 5분동안 짧게 디버깅 시간 가지고 넘어가도록 할게요 :)

➤ 02. 회원가입 심화 배우기 : 커스터마이징(form)

“그런데, 그러면 회원가입 기능을 구현할 때는 꼭 장고에서 만들어준 기능대로 ID 와 PW밖에 받을 수 없는건가요?”

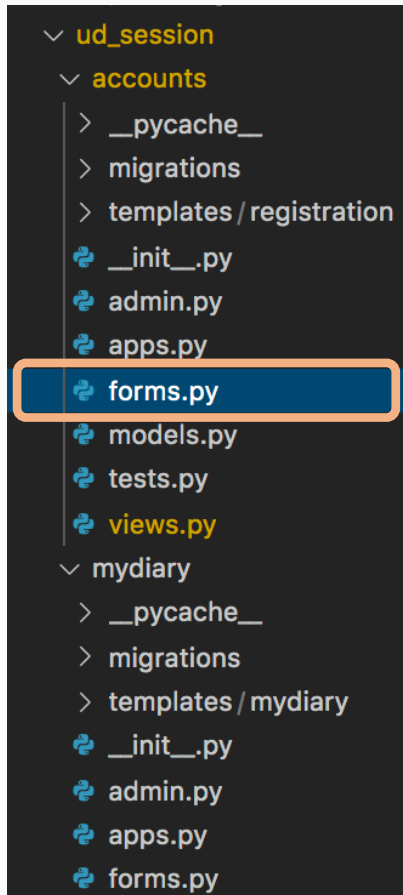
그렇지 않습니다!! 당연히 다양한 정보들을 입력받을 수 있도록 회원가입과 로그인 기능을 커스터마이징할 수 있습니다.

여기에는 다양한 방법이 있는데, 그 방법을 다 배우기에는 너무 많고 복잡하니 오늘은 간단하게 Form을 이용한 커스터마이징을 배워볼거예요!

02. 회원가입 심화 배우기 : 커스터마이징(form)

1) forms.py 만들기

지금까지는 장고에서 기본적으로 제공해주는 UserCreationForm을 사용해 보았습니다.
하지만 이제 커스터마이징을 할 거니까요, 회원가입 커스터마이징을 위한 forms.py 를 만들어줍니다.
역시 mydiary가 아닌 **accounts** 앱 안에 만들어주세요!!!



forms.py

```
ud_session > accounts > forms.py > ...
1  from django import forms
2  from django.contrib.auth.models import User
3  from django.contrib.auth.forms import UserCreationForm
4
5  class UserRegisterForm(UserCreationForm):
6      email = forms.EmailField()
7
8      class Meta:
9          model = User
10         fields = ['username', 'email', 'password1', 'password2',]
11
```

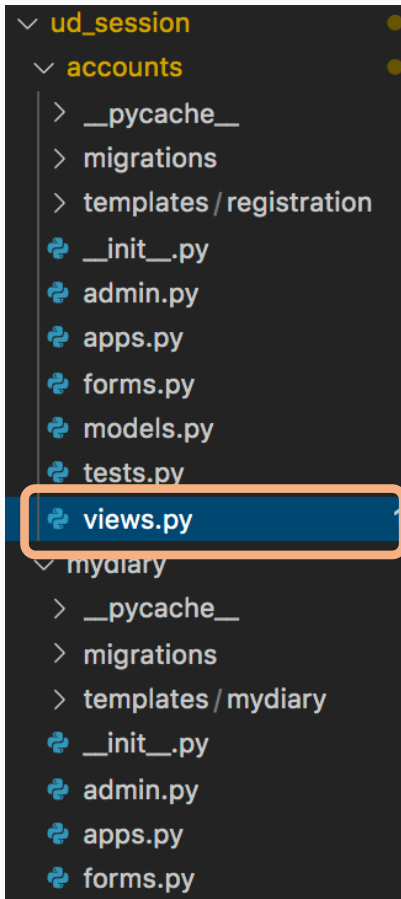
회원가입 form 작동 원리

- 1 : 장고에 내재된(미리 만들어져 있는) forms 패키지를 사용
- 2 : 장고에 내재된 (미리 만들어져 있는) User 모델을 사용
- 3 : 장고에 내재된 (미리 만들어져 있는) UserCreationForm을 사용
- 5 : UserRegisterForm 이라는 새로운 form을 만들어줍니다.
(이 새로운 Form은 UserCreationForm를 상속받습니다)
- 6 : email 필드를 추가합니다.
- 8 : (~ Meta는 Form을 쓸 사람을 위해 장고가 정해놓은 문법)
- 9 : 폼이 다루는 모델 = User
폼이 가진 필드(항목) = username(UserCreationForm이 제공),
Email, password1, password2

02. 회원가입 심화 배우기 : 커스터마이징(form)

2) Views.py 수정하기

Form을 새로 만들어주었으니, 이에 맞추어 views.py도 수정해줍니다.
역시 mydiary가 아닌 accounts 앱 안에 있는 views.py 입니다!!!



Views.py 수정하기

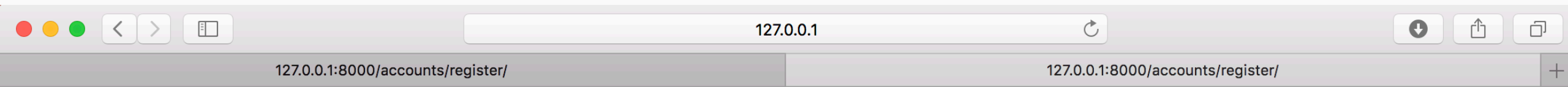
Views.py

```
ud_session > accounts > views.py > ...
1  from django.shortcuts import render, redirect
2  from django.utils import timezone
3  from django.contrib.auth.forms import UserCreationForm
4  from .forms import UserRegisterForm
5
6  # Create your views here.
7  def register(request):
8      if request.method == 'POST':
9          form = UserRegisterForm(request.POST)
10         if form.is_valid():
11             form.save()
12             username = form.cleaned_data.get('username')
13             return redirect('home')
14         else:
15             form = UserRegisterForm()
16         return render(request, 'registration/register.html', {'form':form})
17
```

네모칸 부분을 수정해줍니다!

➤ 02. 회원가입 심화 배우기 : 커스터마이징(form)

그럼 이제 다시한번 서버를 돌려봅시다!
새롭게 email 필드가 생기게 된 것을 확인하실 수 있을 거예요!
서버 돌리고 약 10분간 디버깅 시간 가지고 넘어가도록 할게요 :)



회원가입

사용자 이름: 150자 이하 문자, 숫자 그리고 @/./+/_만 가능합니다.

Email:

비밀번호:

- 다른 개인정보와 비슷한 비밀번호는 사용할 수 없습니다.
- 비밀번호는 최소 8자 이상이어야 합니다.
- 비밀번호는 일상적으로 사용되는 비밀번호일 수 없습니다.
- 비밀번호는 전부 숫자로 할 수 없습니다.

비밀번호 확인: 확인을 위해 이전과 동일한 비밀번호를 입력하세요.

Sign Up /

03. 로그인

이제 회원가입 기능을 모두 구현했으니, 이를 활용해 로그인할 수 있도록 만들어줘야겠죠?
로그인의 경우 View는 장고에서 제공해주지만, templates는 제공하지 않기 때문에 직접 적어줘야 합니다.

1) 로그인 templates 만들어주기

먼저 accounts>templates>registration 폴더 안에
login.html 파일을 만들어줍니다(이름은 정해져 있기 때문에 반드시 이 이름으로 해주셔야 합니다!).

그 후 login.html 파일을 다음과 같이 채워넣어줍니다!

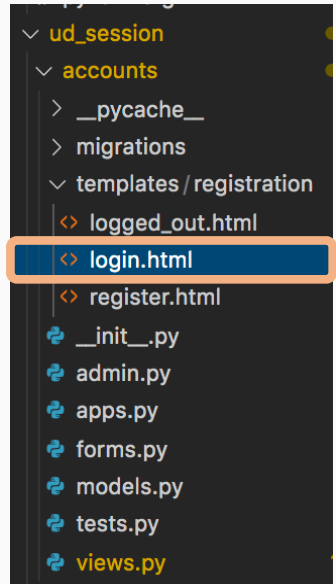
```
ud_session > accounts > templates > registration > <> login.html > ...
1  <h1>로그인</h1>
2  <form method = "POST">
3      {% csrf_token %}
4      {{ form.as_p }}
5      <button type="submit">Login</button><br><br>
6  </form>
7  <b>새로 계정을 만드시겠습니까?</b><br><br>
8  <a href="{% url 'register' %}">회원가입</a>
9
```

login.html

그리고 mydiary 앱 안에 있는 home.html 로 가서,
login.html 로 이동할 수 있는 링크를 만들어줍니다.

```
ud_session > mydiary > templates > mydiary > <> home.html > ...
1  <h1>아기사자의 다이어리입니다.</h1>
2
3  <a href="{% url 'register' %}">회원가입</a><br><br>
4  <a href="{% url 'login' %}">로그인</a><br><br>
5  <a href="{% url 'new' %}">일기 작성하기</a><br><br>
6
7  {% for post in posts_list %}
8      <a href="{% url 'detail' index=post.pk %}">{{ post.title }}</a><br>
9      <hr>
10
11  {% endfor %}
```

home.html



03. 로그인

다음으로 내장되어있는 로그인 Views와 방금 만든 templates를 연결해 줄 수 있는 url을 작성해줍니다.

2) Urls.py 입력하기

urls.py

```
16
17 from django.contrib import admin
18 from django.urls import path, include
19 import mydiary.views
20 import accounts.views
21 from django.contrib.auth import views as auth_views
22
23
24 urlpatterns = [
25     path('admin/', admin.site.urls),
26     path('', mydiary.views.home, name="home"),
27     path('new/', mydiary.views.new, name="new"),
28     path('detail/<int:index>', mydiary.views.detail, name="detail"),
29     path('edit/<int:index>', mydiary.views.edit, name="edit"),
30     path('detail/<int:pk>/delete', mydiary.views.delete, name="delete"),
31     path('registration/register/', accounts.views.register, name="register"),
32     path('registration/', include('django.contrib.auth.urls')),
33     path('registration/login/', auth_views.LoginView.as_view(), {'template_name': 'registration/login.html'}, name="login"),
34 ]
35
```

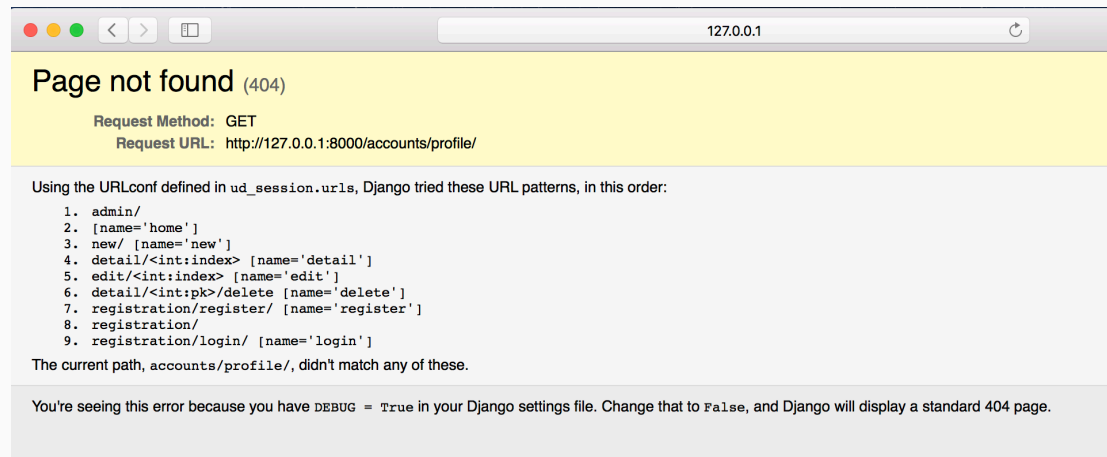
장고에 내장된 views를 Import

장고에 내장된 로그인 views와 templates 연결해주기

03. 로그인

다음으로 서버를 돌려주면!! 다음과 같은 결과화면이 나오게 됩니다!!

하지만 여기서 로그인 정보를 입력하고 로그인 버튼을 누르면!



에러가 발생합니다.

3) settings.py 입력하기

에러가 뜨는 것이 정상입니다.

장고에서 제공하는 내장 views에서 로그인 후에 profile이라는 templates로 연결되도록 설정해두었기 때문인데요.

우리는 profile.html 페이지를 만든 적이 없기 때문에,

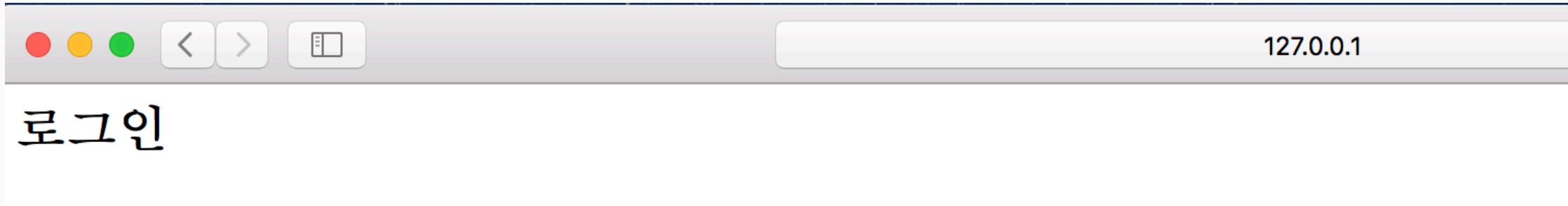
우리의 상황에 맞추어 로그인을 하면 home.html 로 돌아가도록 설정해 줄 것입니다.

```
123
124 LOGIN_REDIRECT_URL = '/'
125
```

Settings.py의 마지막 줄에 다음과 같은 코드를 추가해주세요!

03. 로그인

이제 다시 서버를 돌려주면!! 다음과 같은 결과화면이 나오게 됩니다!!



로그인

사용자 이름:

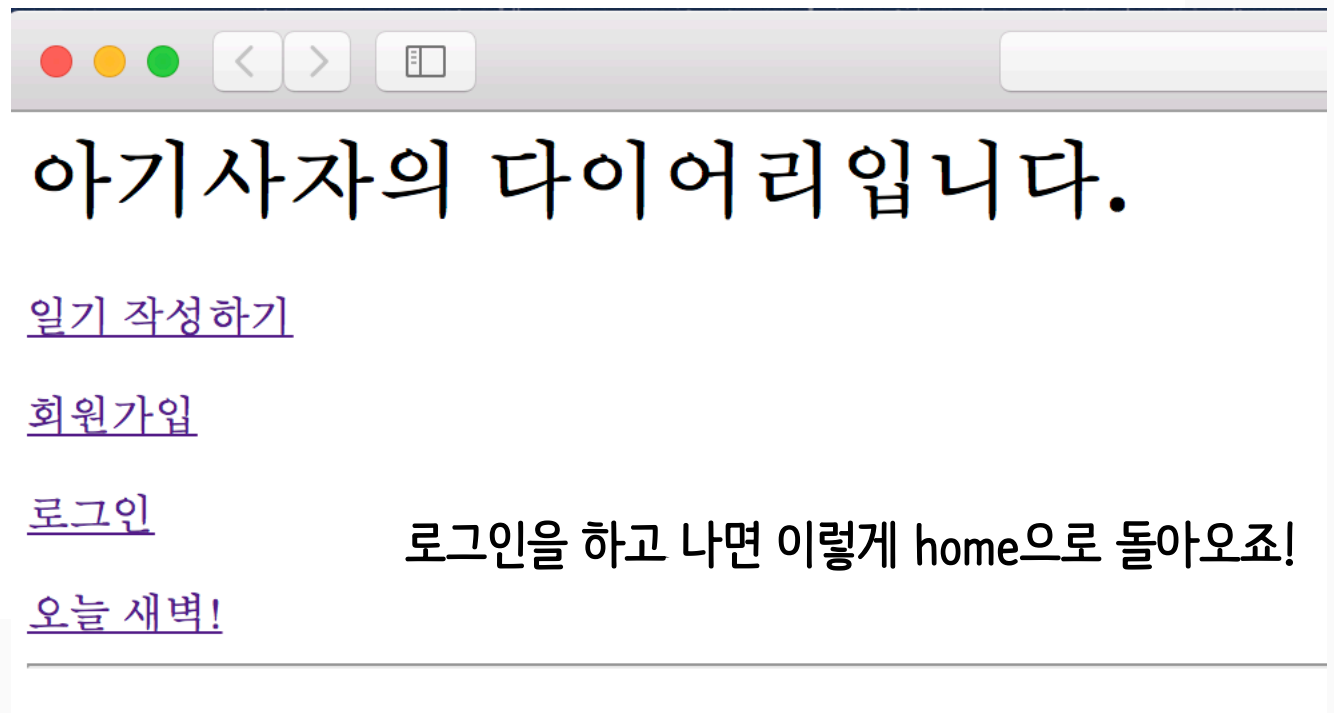
비밀번호:

Login

[회원가입](#)

새로 계정을 만드시겠습니까?

[회원가입](#)



아기사자의 다이어리입니다.

[일기 작성하기](#)

[회원가입](#)

[로그인](#)

[오늘 새벽!](#)

로그인을 하고 나면 이렇게 home으로 돌아오죠!

03. 로그인

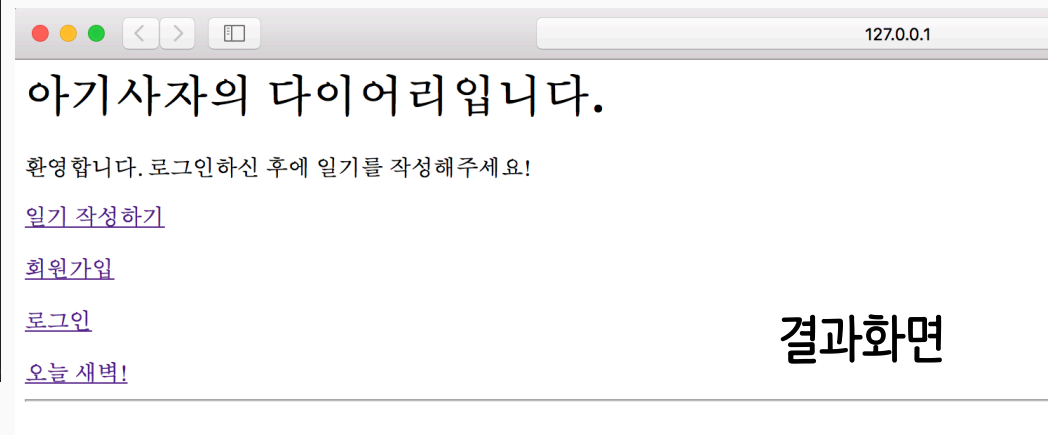
그런데 로그인을 하고 나서, 로그인이 제대로 되었는지 확인할 수가 있어야겠죠?
유저 아이디가 포함된 환영문구와, 로그아웃 버튼을 만들어줄게요.

4) home.html 에 로그인 여부를 확인할 수 있는 코드 입력

```
ud_session > mydiary > templates > mydiary > <> home.html > ...
1  <h1>아기사자의 다이어리입니다.</h1>
2
3  {% if user.is_authenticated %}
4  <p>환영합니다, {{ user.username }}님. 행복한 하루 보내세요!</p>
5  <a href="{% url 'logout' %}">로그아웃</a><br><br>
6  {% else %}
7  <p>환영합니다. 로그인하신 후에 일기를 작성해주세요!</p>
8  {% endif %}
9
10 <a href="{% url 'new' %}">일기 작성하기</a><br><br>
11 <a href="{% url 'register' %}">회원가입</a><br><br>
12 <a href="{% url 'login' %}">로그인</a><br><br>
13
14 {% for post in posts_list %}
15     <a href="{% url 'detail' index=post.pk %}">{{ post.title }}</a><br>
16     <hr>
17 {% endfor %}
18
```

home.html

- 3 : (파이썬 문법)만약 로그인이 되어있다면
- 4 : 가입한 아이디가 포함된 안내문구를 띄우고
- 5 : 로그아웃 버튼을 만든다.
- 6 : (파이썬 문법)만약 로그인이 되어있지 않다면
- 7 : 새 유저 환영문구
- 8 : if문 끝



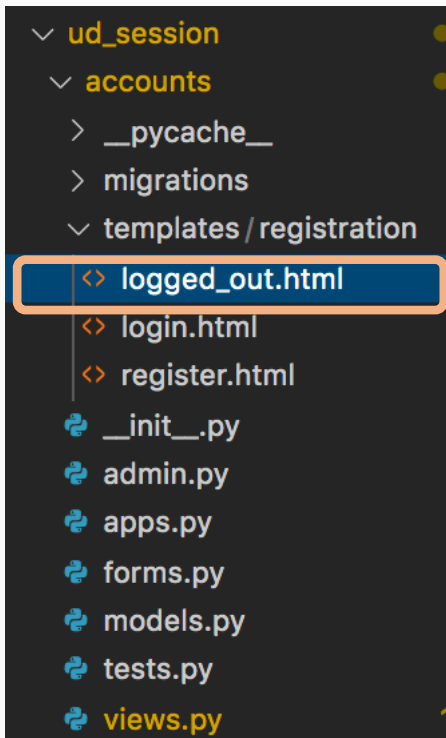
04. 로그아웃

이제 회원가입과 로그인은 모두 끝났습니다.

이제 마지막으로 로그아웃 기능을 구현해 보아요!

로그아웃 역시 View는 장고에서 제공해주지만, templates는 제공하지 않기 때문에 직접 적어줘야 합니다.

1) 로그아웃 templates 만들어주기



먼저 accounts>templates>registration 폴더 안에
logged_out.html 파일을 만들어줍니다(이름은 정해져 있기 때문에 반드시 이 이름으로 해주셔야 합니다!).

그 후 logged_out.html 파일을 다음과 같이 채워넣어줍니다!

```
ud_session > accounts > templates > registration > logged_out.html > ...
1  <h2> Good Bye. </h2>
2  <p><a href="{%url 'login'%}">
3      다시 로그인하기
4  </a></p>
5
6  <p><a href="{%url 'home'%}">
7      홈으로
8  </a></p>
9
```

logged_out.html

로그인 views와 templates 연결해주기

- 1 : 작별인사 글 태그
- 2 : 로그인 페이지로 이어지는 링크
- 3 : 재로그인 안내 문구
- 6 : 홈으로 이동하는 링크

04. 로그아웃

아까 Home.html에서 로그인 확인 문구를 띄울 때 로그아웃 버튼은 만들어줬었죠!
이제 내장된 로그아웃 views와 templates를 연결하는 Url을 적어 마무리해줍니다.

2) Urls.py 입력하기

urls.py

```
17 from django.contrib import admin
18 from django.urls import path, include
19 import mydiary.views
20 import accounts.views
21 from django.contrib.auth import views as auth_views
22
23
24 urlpatterns = [
25     path('admin/', admin.site.urls),
26     path('', mydiary.views.home, name="home"),
27     path('new/', mydiary.views.new, name="new"),
28     path('detail/<int:index>', mydiary.views.detail, name="detail"),
29     path('edit/<int:index>', mydiary.views.edit, name="edit"),
30     path('detail/<int:pk>/delete', mydiary.views.delete, name="delete"),
31     path('registration/register/', accounts.views.register, name="register"),
32     path('registration/', include('django.contrib.auth.urls')),
33     path('registration/login/', auth_views.LoginView.as_view(), {'template_name': 'registration/login.html'}, name="login"),
34     path('registration/logged_out/', auth_views.LogoutView.as_view(), {'template_name': 'registration/logged_out.html'}, name="logout"),
35 ]
36
```

장고에 내장된 로그아웃 views와 templates 연결해주기

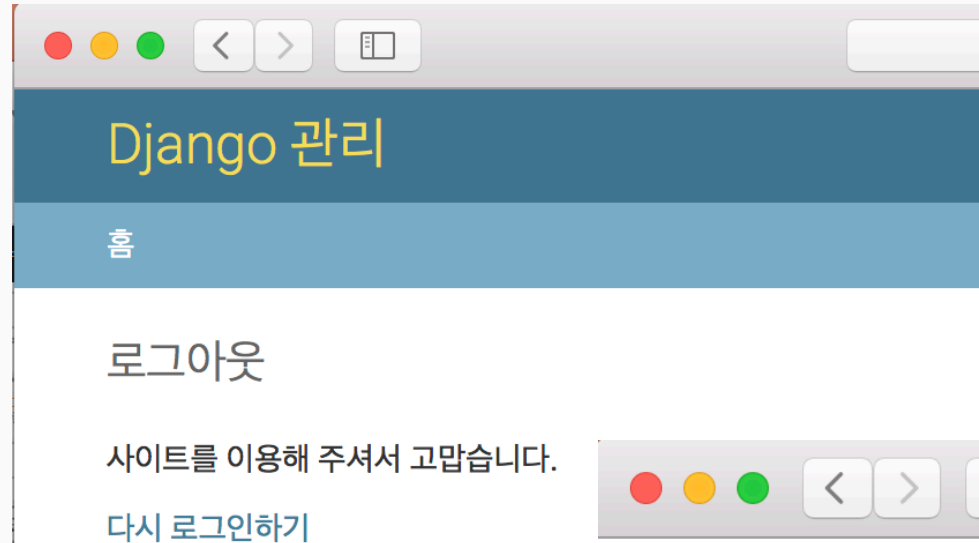
04. 로그아웃

이제 서버를 돌려보면..! 엉? 장고 admin 페이지에서 로그아웃이 됩니다.
우리가 애써 만든 logged_out.html은 어디로 갔을까요?
이러한 현상은 장고에서 app을 읽는 순서 때문에 발생하는데요.

3) Settings.py 조정하기 settings.py

```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'mydiary',
41     'accounts.apps.AccountsConfig',
42 ]
```

```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'mydiary',
35     'accounts.apps.AccountsConfig',
36     'django.contrib.admin',
37     'django.contrib.auth',
38     'django.contrib.contenttypes',
39     'django.contrib.sessions',
40     'django.contrib.messages',
41     'django.contrib.staticfiles',
42 ]
```



최종 결과 화면

이렇게 앱의 위치를 위쪽으로 옮겨서 장고가 auth보다 우리가 만들어준 App을 먼저 읽을 수 있도록 해줍니다!



만약 로그아웃 템플릿 없이 바로 로그아웃하고 싶으신 경우에는 템플릿을 따로 만들지 마시고, settings.py LOGIN_REDIRECTED_URL 밑에 다음과 같이 적어주시면 됩니다!

```
124 LOGIN_REDIRECT_URL = '/'
125 LOGOUT_REDIRECT_URL = '/'
```

05. HomeWork

여기까지 마치셨다면, 축하합니다!!!
아기사자 여러분은 이제 회원가입과 로그인 기능을 구현하실 수 있게 되었습니다.

회원가입과 로그인은 여러가지 방법으로 커스터마이징하실 수 있는데요,
자유롭게 추가 옵션을 선택하여 공부해보셔도 좋습니다!

다음 세션 과제 :

gitHub에 만들었던 나만의 사이트에
회원가입/로그인 기능을 추가해주세요!