# Django

## Authentication System 02

### *회원 가입*

- User 객체를 Create 하는 과정
- UserCreationForm() ; 회원 가입시 사용자 입력 데이터를 받는 built-in ModelForm

## 회원 가입 페이지 작성

```python
# accounts/urls.py

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

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

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

```python
# 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)
```

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

<a href="{% url 'accputns:signup' %}">회원가입</a>
```

- settings.py의 국가 코드를 변경하여 언어 변경 가능

## 회원 가입 로직

```
# 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)
```

- 에러
  - Manager isn't available; 'auth.User' has been swapped for 'accounts.User'
  - 회원 가입에 사용하는 UserCreationForm이 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 Form을 다시 작성
  - UserCreationForm UserChangeForm 두 Form 모두 class Meta: model = User가 작성된 Form이기 때문에 재작성 필요

## UserCreationFrom과 UserChangeForm 커스텀

- Custom User Model을 사용할 수 있도록 상속 후 일부분만 재작성

```python
# accounts/forms.py

from django.contrib.auth import get_user_model
form 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 사용자모델(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

## 회원 가입 로직 완성

- CustomUserCreationForm으로 변경

```python
# accounts/views.py

from .forms import CustomUserCreationForm, CustomUserChangeForm

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)
```

# 회원 탈퇴

- User 객체를 Delete 하는 과정

## 회원 탈퇴 로직

```
# 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')
```

# 회원정보 수정

- User 객체를 Update 하는 과정
- UserChangeForm() ; 회원정보 수정 시 사용자 입력 데이터를 받는 built-in ModelForm

```
# accounts/urls.py

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

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

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

```python
# accounts/views.py

from .forms import CustomUserCreationForm, 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)
```

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

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

## UserChangeForm 사용시 문제점

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

## CustomUser ChangeForm 출력 필드 재정의

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

```python
# accounts/forms.py

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

## 수정 로직 완성

```python
# accounts/views.py

from .forms import CustomUserCreationForm, CustomUserChangeForm

def update(request):
    if request.method=='POST':
        form = CustomUserChangeForm(requst.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)
```

# *비밀번호 변경*

- 인증된 사용자의 Session 데이터를 Update하는 과정
- PasswordChangeForm() ; 비밀번호 변경 시 사용자 입력 데이터를 받는 built-in Form

## 변경 페이지 작성

- django는 비밀번호 변경 페이지를 회원정보 수정 form 하단에서 별도 주소로 안내
  - /user_pk/password/

```python
# crud/urls.py

from accounts import views

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

```html
<!-- 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>
```

```python
# accounts/views.py
```

```python

from django.contrib.auth.forms import PasswordChangeForm

def change_password(request, user_pk):
    if request.method=='POST':
        form = PasswordChangeForm(request.user, 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
  - 적용

```python
# accounts/views.py

from django.contrib.auth.forms import PasswordChangeForm
from django.contrib.auth import update_session_auth_hash

def change_password(request, user_pk):
    if request.method=='POST':
        form = PasswordChangeForm(request.user, 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)
```

# *로그인 사용자에 대한 접근 제한*

## is_authenticated 속성

- 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

- 로그인 비로그인 상태에서 화면에 출력되는 링크를 다르게 설정하기

```
<!-- articles/index.html -->
{% if request.user.is_autenticated %}
<h3>Hello, {{ user.name }}</h3>
<a href="{% url 'user.username' %}">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:update' %}">Login</a>
<a href="{% url 'accounts:update' %}">Signup</a>
{% endif %}
```

- 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

- 인증된 사용자에 대해서만 view 함수를 실행시키는 데코레이터
- 비인증 사용자의 경우 /accounts/login/ 주소로 redirect 시킴

### login_required 적용

- 인증된 사용자만 게시글을 작성/수정/삭제할 수 있도록 수정

```python
# articles/views.py

from django.contrib.auth.decorators import login_reqired

@login_required
def create(request):
    pass

@login_required
def delete(request, article_pk):
    pass

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

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

```python
# accounts/views.py

from django.contrib.auth.decorators import login_reqired

@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
```

# 참고

## *is_authenticated 코드*

- 메서드가 아닌 속성값임을 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*

- 회원 가입 성공한 user 객체를 활용해 login 진행

```
# accounts/views.py

from .forms import CustomUserCreationForm, CustomUserChangeForm

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)
```

```
# UserCreationForm의 save 메서드

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

# 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

- PasswordChangeForm이 다른 Form과 달리 user 객체를 첫번째 인자로 받는 이유
  - 부모 클래스인 SetPasswordForm의 생성자 함수 구성을 따르기 때문
  - https://github.com/django/django/blob/4.2/django/contrib/auth/forms.py#L378

# Auth built-in form 코드

- UserCreationForm()
- UserChangeForm() PasswordChangeForm()