장고 시험 대비 4/1일 수업 내용

Authentication system 2

회원 가입
User 객체를 Create하는 과정

UserCreationForm( )
회원가입시 사용자 입력 데이터를 받는 ModelForm (built-in)

회원가입 views 함수

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

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)
```

커스텀 유저 모델을 사용하기 위해 작성해야 하는 Form
UserCreationForm, UserChangeForm
두 Form 모두 class Meta: model = User가 작성된 Form

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

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 사용자 모델(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 때문이다.
User 모델이 이름이 바뀌거나, 어떤 사유로 바뀌는 경우 User를 직접 참조하고 있는 위치마다 다 바꿔줘야 한다.
그런데 아래 get_user_modle을 사용하면 바꿔주지 않아도 알아서 활성화된 User 객체를 찾아서 반환해준다.
-> 직접 바꿔줄 수고를 없애준다.

django에서도 위 방법을 지양하고 아래 방법을 사용하는 것을 권장하고 있다.

```
커스텀 form 적용 회원가입 로직
from .forms import CustomUserCreationForm

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 Delete하는 과정

```
def delete(request):
    request.user.delete()
    return redirect('articles:index')
```

회원정보 수정
User 객체를 Update하는 과정

UserChangeForm( )
회원정보 수정시 사용자 입력 데이터를 받는 ModelForm(built-in)

UserChangeForm 사용시 문제점
User 모델의 모든 정보들(fields)까지 모두 출력되어 수정이 가능하기 때문에 일반 사용자들이 접근해서 안되는 정보는 출력하지 않도록 해야 함
-> CustomUserChangeForm에서 접근 가능한 필드를 다시 조정

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

회원정보 수정 views 함수

```
def update(request):
    if request.method == 'POST':
        form = CustomUserChangeForm(request.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 Form (built-in)

암호변경시 세션 무효화
비밀번호 변경하면 기존 세션과 회원 인증 정보가 불일치해 로그인 상태가 유지되지 못하고 로그아웃 처리됨
-> 기존세션 != 회원 인증 정보가 되버리기 때문

update_session_auth_hash(request, user)
암호변경 시 세션 무효화를 막아주는 함수
-> 새로운 password의 Session Data로 기존 session을 자동으로 갱신

비밀번호 변경 views 함수

```
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)
```

인증된 사용자에 대한 접근 제한

1. is_authenticated : 속성 (attribute)
   사용자가 인증되었는지 여부를 알 수 있는 User model의 속성
   모든 User 인스턴스에 대해 항상 True인 읽기 전용 속성, 비인증 사용자에 대해서는 항상 False

2. login_required : 데코레이터 (decorator)
   인증된 사용자에 대해서만 view 함수를 실행시키는 데코레이터
   비인증 사용자의 경우 /accounts/login 주소로 redirect 시킴
   -> 앱 이름을 accounts로 사용해야하는 이유

※ **2024.04.01 장고 09-02 장고 실습파일 참고하기**