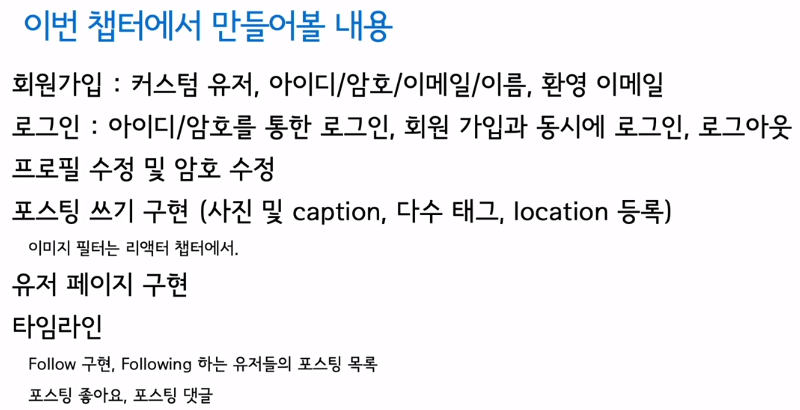
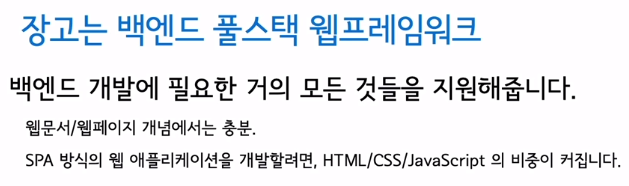
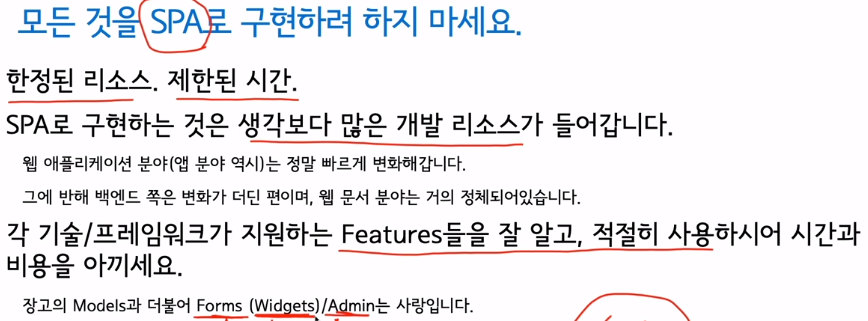
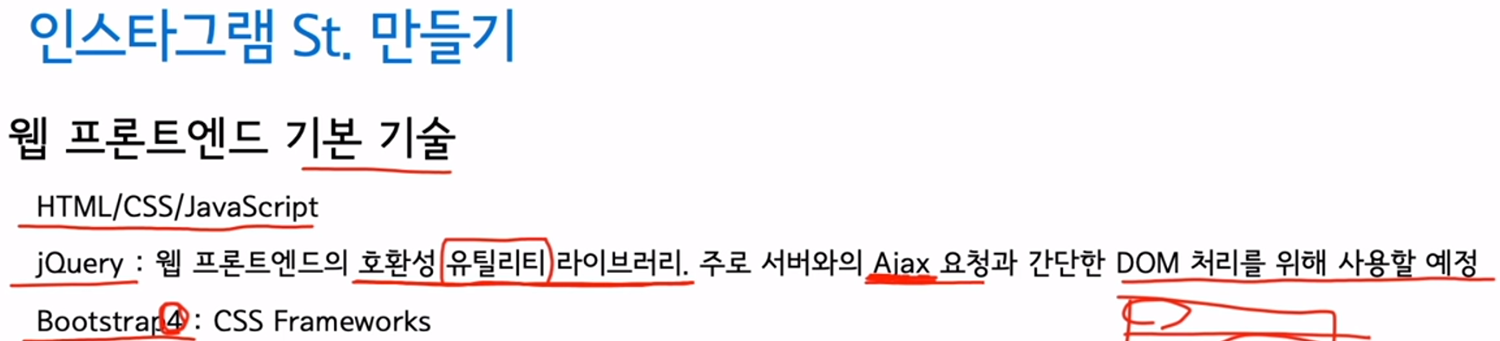
# 챕터7 – 비SPA 방식으로 장고 Forms/Views를 적극 활용한 인스타그램 St 만들기

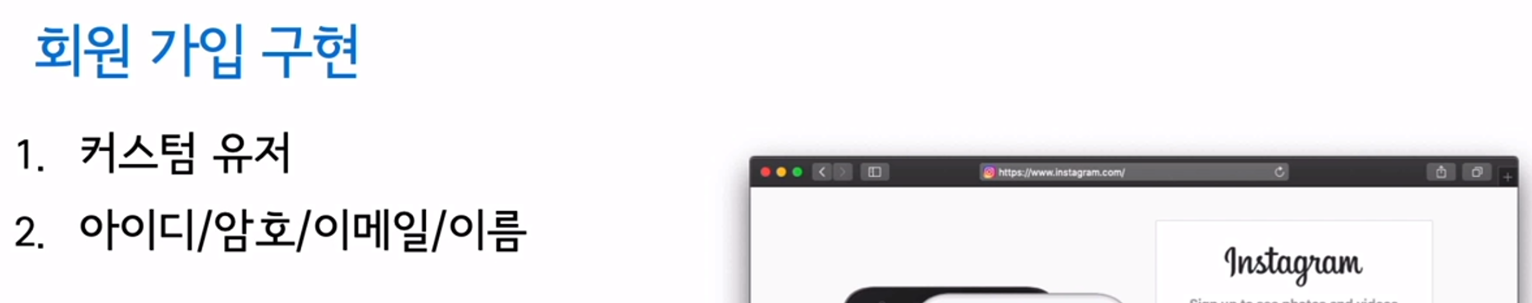
## 01 - Overview







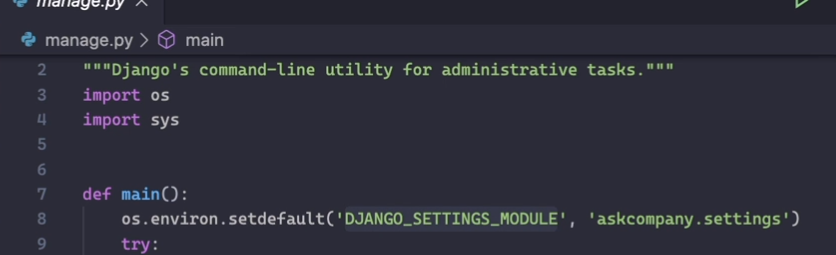




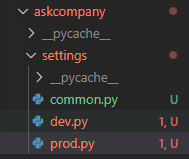
## 02 – 프로젝트 생성 및 초기 프로젝트 환경 설정

로컬에서 장고 프로젝트의 진입점은 manage.py 이고, 서비스 구동할 때는 wsgi.py와 asgi.py가 서버 프로그램의 진입점이다.

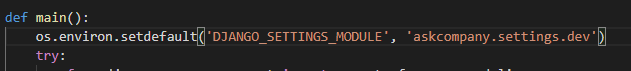
Manage.py에서



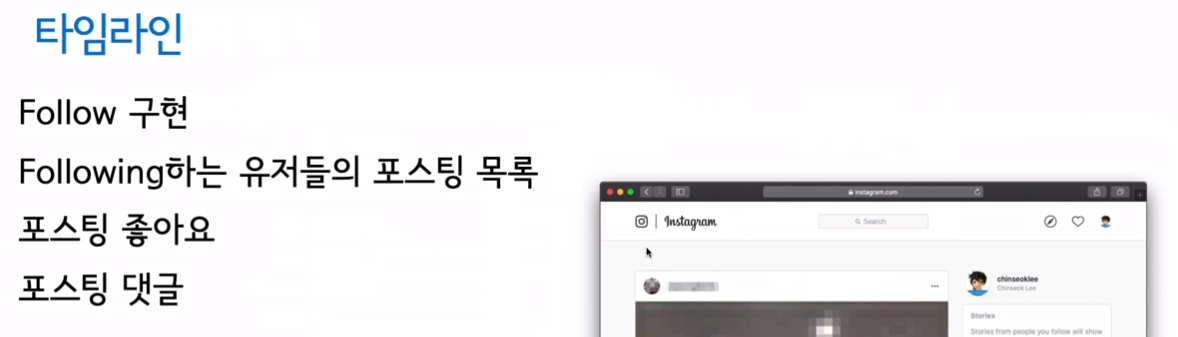
프로그램이 시작했을 때 환경변수 목록에서 ‘DJANGO\_SETTINGS\_MODULE’이 없다면 ‘askcompany.settings’ 값을 대신 환경변수 값으로 지정하겠다는 의미이다.



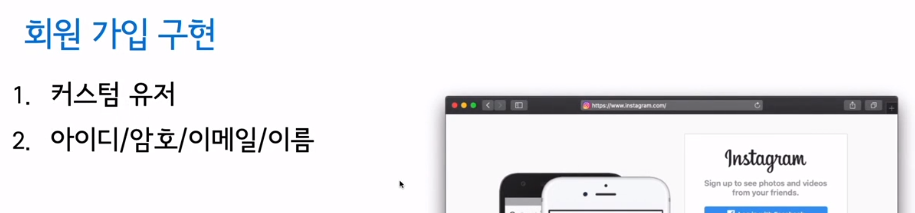
처럼 기존 settings.py 파일을 settings 폴더를 생성하여 하위로 옮겼으므로 manage.py, wsgi.py, asgi.py 에서 settings 파일 경로를 각각 개발,운영 환경에 맞는 설정으로 바꿔야 한다.

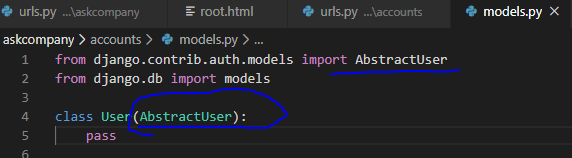


03 Bootstrap4를 활용한 기본 레이아웃 구현



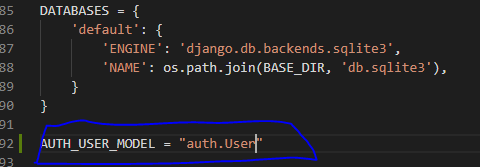
04 커스텀 유저 지정 및 회원가입 구현





장고에서 User model이 꼭 가져야될 필요 스펙들이 있는데, 그런 내용들은 AbstractUser 라는 추상화 모델을 통해서 지원하고 있다.

그래서 User 모델을 만들 때 필히 AbstractUser 모델을 상속받아 사용하는 것이 좋다.



기본적으로 AUTH\_USER\_MODEL 이라는 값은 “auth.User”으로 설정되어 있다.

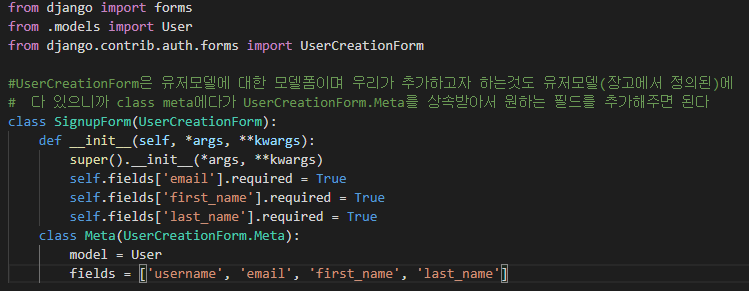
auth라는 앱에 있는 User 모델을 현재 프로젝트에 대한 기본 장고 모델로 쓰겠다는 의미이다.

따라서 장고 기본 auth.User 모델을 쓰는 것이 아니라 다른 custom user 모델을 만들어 쓸 경우 미리 위 설정을 끝마치고 db migrate를 해야 한다.

새로운 custom user model 생성 설정을 안한상태에서 db migrate를 하면 기본 User Model에 대한 테이블이 바뀌기 때문이다.

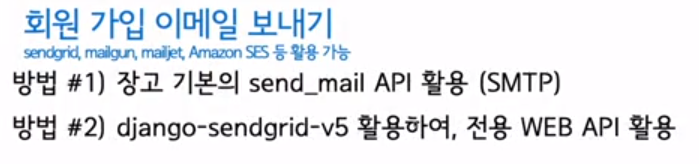


회원가입할 때 사용할 수 있는 UserCreationForm을 장고에서 이미 잘 만들어 놓은게 있으니 활용하면 좋다.



또한 위처럼 기존 정의된 함수를 오버라이딩해서 원하는 기능을 추가할 수 있다.

## 05-회원 가입 이메일 보내기



본 강의에서는 sendgrid를 통하여 메일 전송 기능 구현하는것으로 강의되었지만, 나는 sendgird 회원가입이 여의치 않아서 gmail을 통한 메일 전송 기능으로 구현 하였다.

Google smtp 설정법 블로그

<https://charactermail.tistory.com/237>

Settings.py에 추가 해야함

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_PORT = 587

EMAIL\_HOST\_USER = ‘gmailid’@gmail.com'

EMAIL\_HOST\_PASSWORD = 'password'

EMAIL\_USE\_TLS = True

장고 send\_mail() api doc 링크

<https://docs.djangoproject.com/en/3.0/topics/email/>

mail 보내는 api 사용법

In [1]: from django.core.mail import EmailMessage

In [2]: mail = EmailMessage('title','content',to=['TO@naver.com'])

In [3]: mail.send()

Out[3]: 1

In [4]: from django.core.mail import send\_mail

In [6]: send\_mail("hello title", "hello content", "FROM@gmail.com", [TO@naver.com'])

Out[6]: 1

### 올바른 유저 model 가져오는법

Settings의

AUTH\_USER\_MODEL = "accounts.User"

을 참조하여 활성화된 user 모델을 가져오게 된다.

**In [1]: from django.contrib.auth import get\_user\_model**

**In [2]: User = get\_user\_model()**

In [3]: user = User.objects.first()

In [4]: user.send\_welcome\_email()

### Txt 파일의 string을 읽어서 템플릿 렌더링된 문자열 만드는 부분

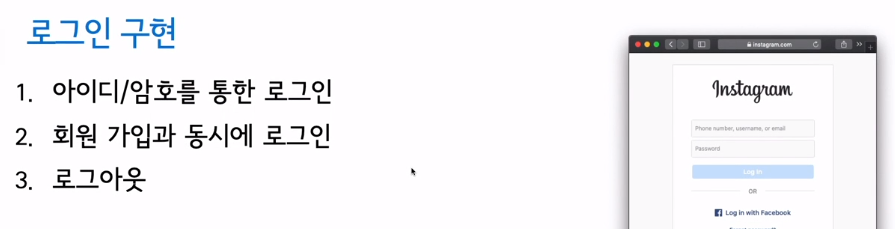
    def send\_welcome\_email(self):

        subject = render\_to\_string("accounts/welcome\_email\_subject.txt", {

            "user": self,

        })

## 07 로그인-로그아웃 구현 그리고 회원 가입과 동시에 로그인



from django.contrib.auth.views import LoginView

login = LoginView.as\_view(template\_name="accounts/login\_form.html")

login 기능은 장고에서 이미 만들어져 있는 LoginView 클래스를 사용해서 구현한다.

LoginView 클래스에서는 django.contrib.auth.forms.py 에서 정의되어 있는 class AuthenticationForm 클래스를 사용하고 있다.

django/contrib/auth/views.py 에서

logout\_then\_login( ) 함수를 이용하여 로그아웃 기능 구현

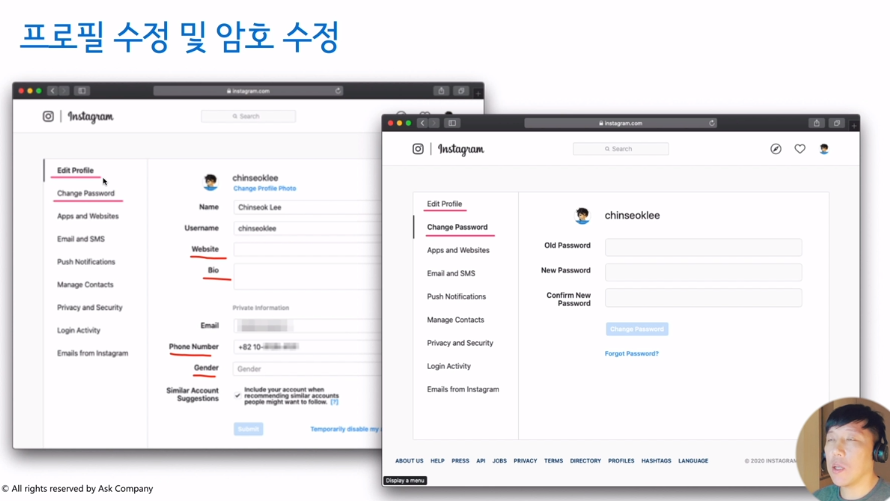
from django.contrib.auth import login as auth\_login

auth\_login(request, signed\_user)

으로 회원가입시 자동 로그인 구현

## 08 – django-pydenticon을 활용하여 프로필 디폴트 이미지 구현

## 09 – 프로필 수정 구현과 장고3에서 추가된 TextChoices 활용



Bio, phone\_number 필드 추가 하였음.

Python manage.py migrate accounts 0002 – 같은 명령어 사용하여 migration 취소할 수 있음.

취소된 생성된 migration 파일은 삭제하면 된다.

그 외 이미지 업로드 할 수 있는 기능 추가 구현하였음

## 10 auth앱의 PasswordChangeForm 커스텀을 통한 암호 수정 구현

Django auth 상속받아 암호 변경 하는 방법 구현

추가로

from django.contrib.auth.forms import PasswordChangeForm as AuthPasswordChangeForm

class PasswordChangeForm(AuthPasswordChangeForm):

    # 변경 후 암호가 변경 전 암호와 동일하면 예외 발생 시키는 코드

    def clean\_new\_password2(self):

        old\_password = self.cleaned\_data.get('old\_password')

        new\_password2 = super().clean\_new\_password2()

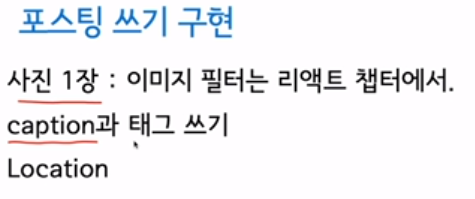
        if old\_password == new\_password2:

            raise forms.ValidationError("새로운 암호는 기존 암호와 다르게 입력해주세요.")

        return new\_password2

부분으로 변경 할 암호는 기존 암호를 사용할 수 없도록 기능 구현 하였음.

## 11 – Instagram 앱 생성과 포스팅 쓰기 구현



Forms.ModelForm 에서는 widgets을 통해서 특정 필드에 대해서 widget을 변경할 수 있다.

class PostForm(forms.ModelForm):

    class Meta:

        model = Post

        fields = ["photo", "caption", "location"]

        widgets = {

            "caption" : forms.Textarea,

        }

            # 필수 필드에 대한 값 지정 없이 commit=True 옵션 으로 form.save()가 호출 되면

            # 에러가 발생한다. 그래서 commit=False 옵션 줘서 바로 db에 저장되지 않도록 한다.

            post = form.save(commit=False)

Post, Tag model 클래스 생성

### Tag 클래스의 값을 Post 클래스의 caption 값에서 추출해서 저장을 하던대 어떻게 이렇게 기능이 동작을 하는건지 잘 모르겠음. 알아볼 것