**Django UserCreationForm**

Django have vas verities of built-in template/form. Here we are going to see few of them.

For better understanding of usercreation/registration form better to know below built-in modules/class

* User Objects

<https://docs.djangoproject.com/en/3.1/topics/auth/default/>

* User models

<https://docs.djangoproject.com/en/3.1/ref/contrib/auth/#django.contrib.auth.models.User>

* UserCreationForm class from django.contrib.auth.form

<https://docs.djangoproject.com/en/3.1/topics/auth/default/>

* Attributes of boundfileds

<https://docs.djangoproject.com/en/3.1/ref/forms/api/>

* Meta class

**UserCreationForm**

This is ModelForm (django.contrib.auth.forms) for creating a new user.

It has three fields: 1.) username (from the user model), 2.) password1, and 3.) password2

It verifies that password1 and password2 match, validates the password using validate\_password(), and sets the user’s password using set\_password()

**Steps to creating form**

Import UserCreationForm in view.py file and render.

Simple way of rendering form

def sign\_up(request):

    return render(request,'enroll/signup.html',form)

Forms with capability of saving data to DB

def signup(request):

    if request.method=='POST':

        print('getting data from form')

        fm=UserCreationForm(request.POST)

        if fm.is\_valid():

            print('forms is valid')

            fm.save()

            form={'form':fm,}

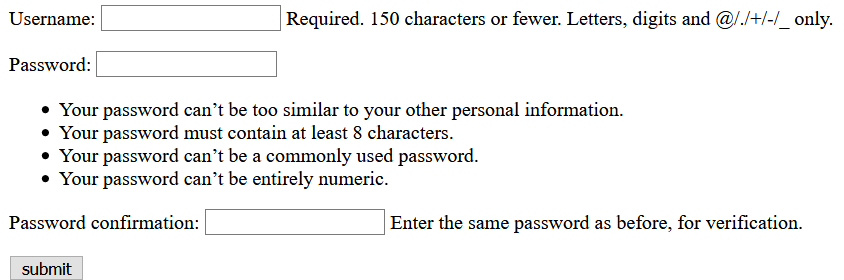
            return render(request,'login/signup.html',form)

        else:

            return HttpResponse('<h1>User was not  created</h1>')

If we create form in above way it’s going to shows some unrequired fileds which doesn’t looks good.

That we can see from below screenshot



**Customizing Django forms/Usercreationform**

Let say we don’t want waring or info message for each input.

We can achieve it by using the boundfields attributes , forms tags etc.

Let’s take a look on Django code of HTML.

<body>

    <form action="" method="POST"  novalidate>

        {% csrf\_token %}

        {% for fm in form %}

            {{ fm.label\_tag}}{{ fm }}{{ fm.errors|striptags }}

            <br>

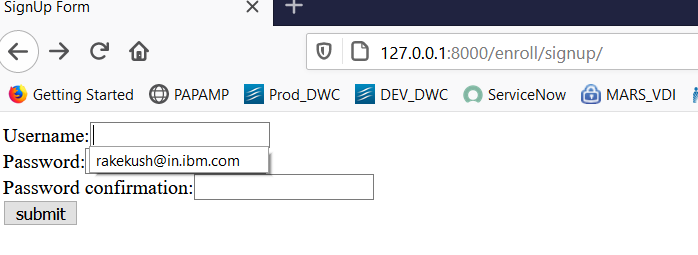
        {% endfor %}

        <input type="submit" value="submit">

    </form>

</body>

Now this will remove all warning/info files and forms will look more cleaner as:-



**Adding customfields**

Let say we want to add email-id, first name, last name etc in our registration form

We can achieve it using the help of User objects, Meta class and inheritance

**Steps**

Create a forms.py file in your app, create a class for your form and writer Meta class as inner class and update the files you want. It will look like below in form.py file

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

class SignUp(UserCreationForm):

    class Meta:

        model=User

        fields=['username','email','first\_name','last\_name']

We will not touch the html forms data, it’s same as previous

In view.py file views name for user creation will look like:-

from django.shortcuts import render

from django.contrib.auth.forms import UserCreationForm

from django.http import HttpResponse

**from .forms import SignUp**

def sign\_up(request):

    if request.method == "POST":

        return HttpResponse('<h3>Form data was saved</h3>')

    else:

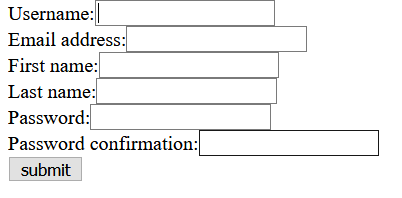
        fm=SignUp()

        form={'form':fm,}

        #return HttpResponse('<h3>Welcome signup paeg</h3>')

        return render(request,'enroll/signup.html',form)

Now our forms will look like-



Note:

We can change the label name for each field of field of form using ‘lable’ attribute from it.

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

class SignUp(UserCreationForm):

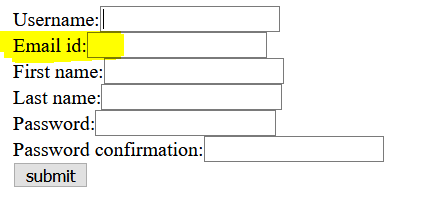
    class Meta:

        model=User

        fields=['username','email','first\_name','last\_name']

        labels={'email':'Email id'}

Now using the labels attribute we can change the default label of email field to ‘Email id’



**redirect\_to\_login(next, login\_url=None, redirect\_field\_name='next')**

This method redirect to login page and then back to URL

Useful when any user directly hits a URL but he is not logged

We can set login url using LOGIN\_URL in setting.py, more on below link

<https://docs.djangoproject.com/en/3.1/ref/settings/#std:setting-LOGIN_URL>

**Django user objects/Model**

django.contrib.auth.models

Full details can be checked on below links

<https://docs.djangoproject.com/en/3.1/topics/auth/default/>

<https://docs.djangoproject.com/en/3.1/ref/contrib/auth/#django.contrib.auth.models.User.set_password>

User objects are the core of the authentication system.

They typically represent the people interacting with your site and are used to enable things like restricting access, registering user profiles, associating content with creators etc.

1. **Attributes of user objects**

The primary attributes of the default user are

* username
* password
* email
* first\_name
* last\_name

There all attributes are case and character sensitive. To use check usercreationform doc.

1. **Creating user**

For creating user we can take help of **create\_user()** helper function.

*create\_user(username, email=None, password=None, \*\*extra\_fields)*

Creates, saves and returns a User

The username and password are set as given. The domain portion of email is automatically converted to lowercase, and the returned User object will have **is\_active** set to True.

1. **Changing passwords**

Django does not store raw (clear text) passwords on the user model, but only a hashed

Because of this, do not attempt to manipulate the password attribute of the user directly. This is why a helper function is used when creating a user.

We can change password using **set\_password()**function

user\_obj.set\_password(raw\_password)

This function takes raw/plain text password and store it to DB in hashed format.

from django.contrib.auth.models import User

u = User.objects.get(username='john') --- get the user with username

u.set\_password('new password') ---- change the password for that user

u.save()

**Authenticating users**

Use **authenticate**() to verify a set of credentials. It take 2 arguments 1) username 2) password

**Returns a User object if the credentials are valid for a backend.**

If the credentials aren’t valid for any backend or if a backend raises PermissionDenied, it returns None

User objects means django user object which contain all user object information(username,first name,full name) , for more details check Django user objects

from django.contrib.auth import authenticate

user = authenticate(username='john', password='secret')

if user is not None:

    # A backend authenticated the credentials

else:

    # No backend authenticated the credentials

**Checking django user authentication**

Django uses sessions and middleware to hook the authentication system into request objects(djnago request object)

These provide a **request.user** attribute on every request which represents the current user. If the current user has not logged in, this attribute will be set to an instance of **AnonymousUser**, otherwise it will be an instance of User.

**is\_authenticated:**

**-----**this is used to check if the given user id logged in or not. E.g:-

*If request.user.is\_authenticated:*

*#do something*

*Else:*

*#send the user to login page*

**is\_anonymous:**

----------------- Returns true if the user id is not authenticated

**How to login/logout a user**

First validate the user credetails

If correct then call **login(request,username)** ---- from django.contrib.auth import login

While logging out call **logout(request)** --------- from django.contrib.auth import logout

**Checking logout in django templates**

If we can to check if user is logged in or not then below will work for templates.

***{% if user.is\_authenticated %}*** ---------------- this checks if user is logged in or not in templates

It’s typical use is to add logout option in navbar

**Django login implementation**

For logging in we will use building login form/class in django **"AuthenticationForm"** from **"django.contrib.auth.forms"** module

* AuthenticationForm() ---- simple blank login form(username, password as input field)
* AuthenticationForm(request=request,data=request.POST) ---- this is generally used for getting data(old password, new password) of password change form

**Steps:**

1. First check the if request is coming for authentication using request.method or we have to render form for changing password
2. If need to render then render the authentication form
3. If request if coming then validate, authenticate and then redirect to required page

def user\_login(request):

    if request.method=="POST":

        print('in user\_login views name')

        fm=AuthenticationForm(request=request,data=request.POST)

        if fm.is\_valid():

            print('forms is valid')

            uname=fm.cleaned\_data['username']

            passwd=fm.cleaned\_data['password']

            user = authenticate(username=uname, password=passwd)

            if user is not None:

                login(request,user)

                print('user id authenticated')

                name={'name':uname,}

                return render(request,'login/profile.html',name)

        else:

            print('invalid user id or password')

            form={'form':fm,}

            return render(request,'login/login.html',form)

    else:

        print('in user\_login view')

        fm=AuthenticationForm()

        form={'form':fm,}

        return render(request,'login/login.html',form)

**Django logout implementation**

Call logout(request) in logout views

def user\_logout(request):

    logout(request)

    return HttpResponseRedirect('/login/')

**-:Changing password using django:-**

We will sue "**PasswordchangeForm**" or **"SetPassword"** class/form from django available in django.contrib.auth module.

This form have old pass, new pass, comfirm new pass filed

PasswordChangeForm(user=request.user,data=request.POST)—for getting data from form

PasswordChangeForm()----------- blank form for password change

Steps:

Major steps and concept are same as login form

def change\_password(request):

    if request.user.is\_authenticated: #check if user is logged in or not, if yes then

        if request.method=='POST': # proceed below else ask him to login first

            print('fetching detail for resetting the password')

            fm=PasswordChangeForm(user=request.user,data=request.POST)

            if fm.is\_valid(): #Check if form data are correct(old password)

                print('updating the password') *#new password and confirmed new pass*

                fm.save() #save will update the password

                logout(request)

                return HttpResponseRedirect('login/profile/')—redirect to login page

            else: #**form data not valid then redirect to same form**

                fm=PasswordChangeForm(user=request.user)

                form={"form":fm}

                return render(request,'login/changepassword.html',form) #

        else:

            fm=PasswordChangeForm(user=request.user)

            form={"form":fm}

            return render(request,'login/changepassword.html',form)

    else:

        return HttpResponseRedirect('/login/auth')--- rende

**NOTE:**

if request.user.is\_authenticated:-------- this must be implemented in each views method

**User Model or user class in django.contrib.auth.models**

<https://docs.djangoproject.com/en/3.1/ref/contrib/auth/#django.contrib.auth.models.UserManager.create_user>

It contains fields/attributes related to user, mainly for password change, get user attributes etc.

User objects have the following fields/attributes:

1. username
2. first\_name
3. last\_name
4. email
5. passowrd
6. groups
7. user\_permissions
8. is\_staff
9. is\_active
10. is\_super
11. last\_login
12. date\_joined
13. **Methods**

There are many methods few commona are:-

1. get\_username()--returns username
2. get\_full\_name()--returns full name
3. get\_short\_name()--returns fisrt\_name
4. set\_password(raw\_password)---change password
5. check\_password(raw\_password)--returns true is password is correct

**-:Adding profile picture in navbar:-**

For better understanding we should know something about MEDIA\_ROOT, MEDIA\_URL and upload\_to field in Imagefield filetype

For adding profile picture in navbar we need to create a function based model in models.py file and inherit the models.Model class

Add the image file in this class and make OneToOne relationship with building user model

class Profile(models.Model):

    user=models.OneToOneField(User,on\_delete=models.CASCADE)

    image=models.ImageField(default='profilepic.jpeg',upload\_to='profile\_picture')

    def \_\_str\_\_(self):

        return self.user.username

Now this will create a new table in database by name appName\_Profie in that one column will be image

Now when we run the migration based on the value of 'upload\_to' value and 'MEDIA\_ROOT', 'MEDIA\_URL'

Django will create a directory on the server to server the images

MEDIA\_ROOT=os.path.join(BASE\_DIR,'picture')-# means create adirectoey ‘picture’ for images

MEDIA\_URL='/picture/'

If MEDIA\_ROOT not given the it will upload/make ready images in our BASE\_DIR.

Now it will look like below after migration.

**How to get it**

<img src="{{ user.profile.image.url }}">

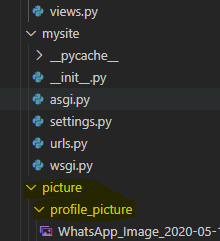
*{{get\_object\_of\_profile\_class\_or\_required\_class.image\_field\_name.url}}*

<img src="{{ user.profile.image.url }}">

User---- by default available attribute by django of user Model

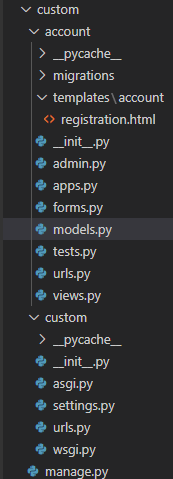
profile----Object of Profile model class

image---label or name given for ImageField



**Overriding User mode in django and creating signup form**s

Project skelton.



Models.py file:

from django.db import models

from django.contrib.auth.models import AbstractBaseUser, BaseUserManager

class MyAccountManager1(BaseUserManager):

    def create\_user(self,email,username,password=None):

        if not email:

            raise ValueError("email address can't be blank")

        if not username:

            raise ValueError("user name filed can't be blank")

        user=self.model(

            email=self.normalize\_email(email),

            username=username,

        )

        user.set\_password(password)

        user.save(using=self.\_db)

        def create\_superuser(self,email,username,passowrd):

            user=self.create\_user(

                email=self.normalize\_email(email),

                password=password,

                username=username,

            )

            user.is\_admin=True

            user.is\_staff=True

            user.is\_superuser=True

            user.save(using=self.\_db)

            return user

# Create your models here.

class Account(AbstractBaseUser):

    email=models.EmailField(max\_length=100,unique=True)

    username=models.CharField(max\_length=20, unique=True)

    date\_joind=models.DateField(verbose\_name='date\_joined',auto\_now\_add=True)

    last\_login=models.DateField(verbose\_name='last\_login',auto\_now=True)

    mobile\_number=models.IntegerField(default=91)

    is\_admin=models.BooleanField(default=False)

    is\_active=models.BooleanField(default=True)

    is\_staff=models.BooleanField(default=False)

    is\_superuser=models.BooleanField(default=False)

    USERNAME\_FIELD= 'email'

    REQUIRED\_FIELDS=['username']

    def \_\_str\_\_(self):

        return self.email

    def has\_perm(self,perm,obj=None):

        return self.is\_admin

    def has\_module\_perms(self,app\_label):

        return True

forms.py file:

from .models import Account

from django.contrib.auth.forms import UserCreationForm

class Register(UserCreationForm):

    class Meta:

        model=Account

        fields=['email','username','mobile\_number']

views.py file:

from django.shortcuts import render

from django.http import HttpResponse

from .forms import Register

# Create your views here.

def user\_register(request):

    if request.method=="POST":

        print('getting forms data')

        fm = Register(request.POST)

        if fm.is\_valid():

            fm.save()

            print('form is valid and submitting the form data to database')

            return HttpResponse('resistration completed')

        #return render(request,'first/registration.html')

        else:

            context={'form':fm,}

            return render(request,'account/registration.html',context)

    else:

        print('need to create form')

        reg=Register()

        context={'form':reg,}

        return render(request,'account/registration.html',context)

settings.py file:

AUTH\_USER\_MODEL='account.Account'

Add above line