**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 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 sign\_up(request):

    if request.method == "POST":

        fm=UserCreationForm(request.POST)

        #validate the form data

#save data to database

#if you want redirect to required page

    else:

        fm=UserCreationForm()

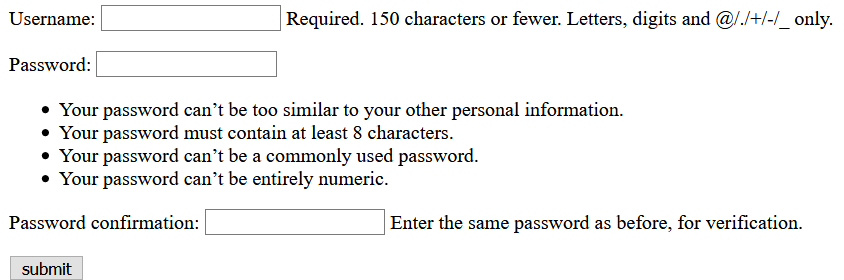
        form={'form':fm,}

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

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

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>

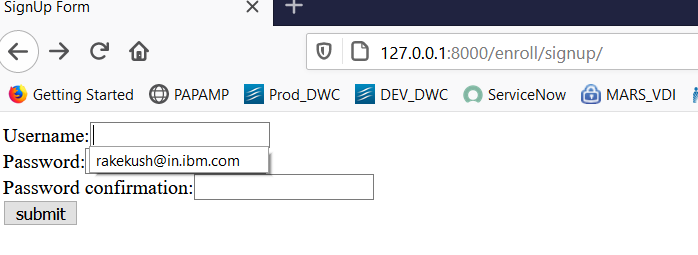
        {% endfor %}

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

    </form>

</body>

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



**Adding custom fields**

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, 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:

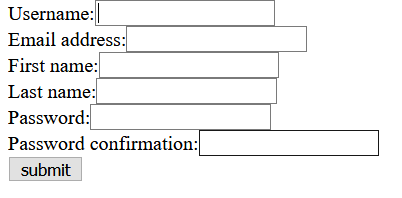
        fm=SignUp()

        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):

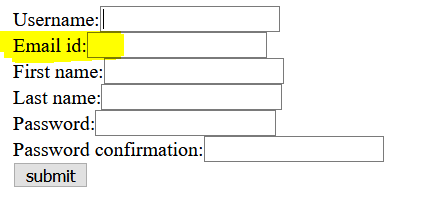
    class Meta:

        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**

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

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

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

**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:

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