USER PROFILE

In this article, we'll create a user profile that extends the built-in Django User model.

The users can also update their information and add a new profile picture.

Create User Profile

1. Add class Profile in users/models.py

```
from django.db import models
from django.contrib.auth.models import User


class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE) # Delete profile
when user is deleted
    image = models.ImageField(default='default.jpg', upload_to='profile_pics')

    def __str__(self):
        return f'{self.user.username} Profile' #show how we want it to be
displayed
```

2. Install Pillow

Pillow is a library working with images in Python.

```
python -m pip install Pillow
```

3. Create migration:

```
python manage.py makemigrations
```

4. Run migration

```
python manage.py migrate
```

5. Register Profile model

We need to register the Profile model in users/admin.py so we can access Profile in our Admin panel.

```
from django.contrib import admin
from .models import Profile

# Register your models here.
admin.site.register(Profile)
```

6. Change location of profile images

Open project mysite/settings.py file, under `STATIC_URL = '/static/'` , add this:

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media') # Directory where uploaded media is
saved.
MEDIA_URL = '/media/' # Public URL at the browser
```

Explanation

- `MEDIA_ROOT = os.path.join(BASE_DIR, 'media')` means the media root will be located in our project directory. When we upload an image, the image will be saved in the media directory.
- `MEDIA_URL = '/media/'` is how we can access URL at the browser

7. Create a Profile

a) Using Python shell

We can add Profile for our existing users in the Python shell.

- Run command: `$ python manage.py shell`
- Run

```
>>> from django.contrib.auth.models import User
>>> from users.models import Profile
>>> user = User.objects.get(username='<admin_user_name>')
>>> profile = Profile(user=user)
>>> profile.save()
```

b) Using Admin Panel

We can also add new users to Admin Panel.

- Run `python manage.py runserver` .
- Login: localhost:8000/admin/.
- Access: localhost:8000/admin/users/profile/add/.
- Add z new user and upload a profile picture.

```
.
├── media
│   ├── default.jpg
│   └── profile_pics
│       └── pic.jpg
```

8. Add default.jpg in the media

Now we'll see a `media` directory appear in our root directory thanks to the setting in step 6.

Now add a `default.jpg` in the media directory.

9. Update users/profile.html template

In our previous lesson, we created a basic template.

```
{% extends "blog/base.html" %} {% load crispy_forms_tags %} {% block content %}
<h1>{{ user.username }}</h1>
{% endblock content %}
```

Now we update it to show username, email, and image:

```
{% extends "blog/base.html" %} {% load crispy_forms_tags %} {% block content %}
<div class="content-section">
  <div class="media">
    <img
      class="rounded-circle account-img"
      src="{{ user.profile.image.url }}"
    />
    <div class="media-body">
      <h2 class="account-heading">{{ user.username }}</h2>
      <p class="text-secondary">{{ user.email }}</p>
    </div>
  </div>
  <!-- FORM HERE -->
</div>
{% endblock content %}
```

10. Update mysite/urls.py file

According to Django static file documentation, we can serve files uploaded by a user during development by adding this snippet to urls.py.

```
from django.conf import settings
from django.conf.urls.static import static
```

```
urlpatterns = [
    # ... the rest of your URLconf goes here ...
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

So now, open mysite/urls.py and update:

```
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from users import views as user_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('register/', user_views.register, name='register'),
    path('profile/', user_views.profile, name='profile'),
    path('login/',
auth_views.LoginView.as_view(template_name='users/login.html'), name='login'),
    path('logout/',
auth_views.LogoutView.as_view(template_name='users/logout.html'),
name='logout'),
    path('', include('blog.urls')),
]

# Only add this when we are in debug mode.
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

11. Add Django signals

We need to add Django signals, so every new user has a default profile picture as well.

- In our user's directory, create signals.py.

```
from django.db.models.signals import post_save #Import a post_save signal when a
user is created
from django.contrib.auth.models import User # Import the built-in User model,
which is a sender
```

```python
from django.dispatch import receiver # Import the receiver
from .models import Profile


@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)


@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()
```

12. Update users/apps.py

```python
from django.apps import AppConfig


class UsersConfig(AppConfig):
    name = 'users'

    def ready(self):
        import users.signals
```

Update User Profile

We can create forms to allow users to update their username, email, and a profile picture.

1. Update users/forms.py

We create UserUpdateForm and ProfileUpdateForm in users/forms.py

```python
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from .models import Profile
```

```
class UserRegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

# Create a UserUpdateForm to update a username and email
class UserUpdateForm(forms.ModelForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email']

# Create a ProfileUpdateForm to update image.
class ProfileUpdateForm(forms.ModelForm):
    class Meta:
        model = Profile
        fields = ['image']
```

2. Update users/views.py

```
from django.shortcuts import render, redirect
from django.contrib import messages
from django.contrib.auth.decorators import login_required
# Import User UpdateForm, ProfileUpdatForm
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created! You are
now able to log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})

# Update it here
@login_required
```

```python
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=request.user)
        p_form = ProfileUpdateForm(request.POST,
                                   request.FILES,
                                   instance=request.user.profile)
        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, f'Your account has been updated!')
            return redirect('profile') # Redirect back to profile page

    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form
    }

    return render(request, 'users/profile.html', context)
```

- instance=request.user: User form will have the current information.
- request.FILES: means that it will have files.

3. Add form in profile.html

```html
{% extends "blog/base.html" %} {% load crispy_forms_tags %} {% block content %}
<div class="content-section">
  <div class="media">
    <img
      class="rounded-circle account-img"
      src="{{ user.profile.image.url }}"
    />
    <div class="media-body">
      <h2 class="account-heading">{{ user.username }}</h2>
      <p class="text-secondary">{{ user.email }}</p>
    </div>
  </div>
  <!-- FORM HERE -->
  <form method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    <fieldset class="form-group">
```

```
      <legend class="border-bottom mb-4">Profile Info</legend>
      {{ u_form|crispy }}
      <!-- User form -->
      {{ p_form|crispy }}
      <!-- Profile form -->
    </fieldset>
    <div class="form-group">
      <button class="btn btn-outline-info" type="submit">Update</button>
    </div>
  </form>
</div>
{% endblock content %}
```

Explanation

- `enctype="multipart/form-data"` helps saving image data.

4. Resize the image when a user upload

Edit users/models.py:

```
from django.db import models
from django.contrib.auth.models import User
from PIL import Image

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg', upload_to='profile_pics')

    def __str__(self):
        return f'{self.user.username} Profile'

    # Override the save method of the model
    def save(self):
        super().save()

        img = Image.open(self.image.path) # Open image

        # resize image
        if img.height > 300 or img.width > 300:
            output_size = (300, 300)
            img.thumbnail(output_size) # Resize image
            img.save(self.image.path) # Save it again and override the larger
image
```

5. Display an image of the author beside each post on Homepage.

- Open blog/index.html
- After the article tag, add: `<img class="rounded-circle article-img" src="{{ post.author.profile.image.url }}">`

```
{% extends "blog/base.html" %} {% block content %} {% for post in posts %}
<article class="media content-section">
  <img
    class="rounded-circle article-img"
    src="{{ post.author.profile.image.url }}"
  />
  <div class="media-body">
    <div class="article-metadata">
      <a class="mr-2" href="#">{{ post.author }}</a>
      <small class="text-muted">{{ post.date|date:"F d, Y" }}</small>
    </div>
    <h2><a class="article-title" href="#">{{ post.title }}</a></h2>
    <p class="article-content">{{ post.content }}</p>
  </div>
</article>
{% endfor %} {% endblock content %}
```