Registeration Form

Created	@October 14, 2023 11:34 AM

-The quick steps to create a registeration form:

1) Views

2) Template

3) Forms.py

4) Install Crispy Forms

but .. we'll go through them in details to make applying them easier, so we'll divide them to 6 smaller steps like this :

6 Detailed Steps to create a registeration form :

-After creating your new app : User ,

1) In Views, import the UserModelForm from django

2) Make your HTML Template

3) (Optional) : Add url of User app in project's urls.py

4) Update your Views to get and save data

4) Create your desired form fields in forms.py

5) Apply the changes to your views.py

6) Install Crispy forms for styling

and in the end we'll summarize them again as the quick 4 steps ..

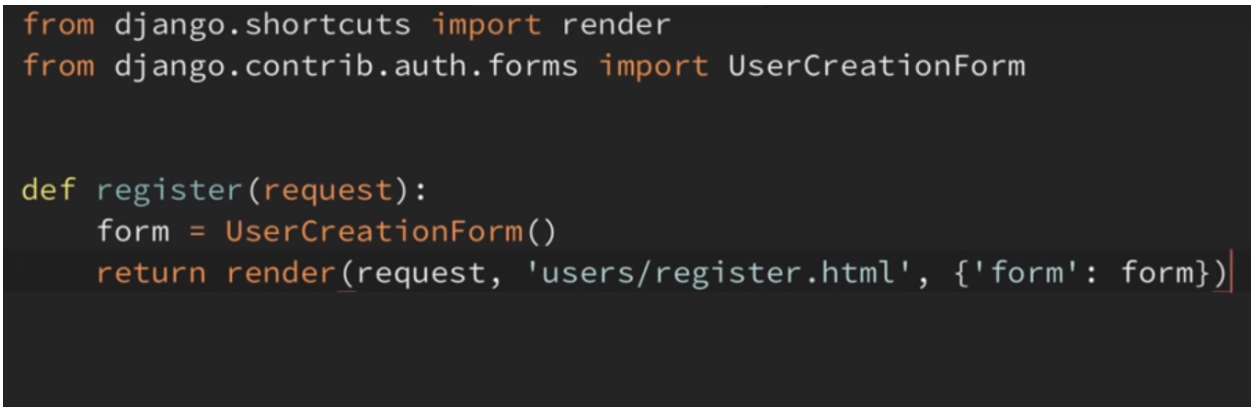
---

1) views.py :

- **import UserCreationForm for the built in django forms in the package : django.contrib.auth.form**

  this package makes you a built in form input so you don't have to manually create it .. all you have to do is just render it in your html templates, and just create the forms method tags & csrf token & the buttons you like (submit, login,…) as well as give them the styles

- **create an instance from that form**
- **render it in your html template using a context name "form" from that form**



```
from django.shortcuts import render
from django.contrib.auth.forms import UserCreationForm


def register(request):
    form = UserCreationForm()
    return render(request, 'users/register.html', {'form': form})
```

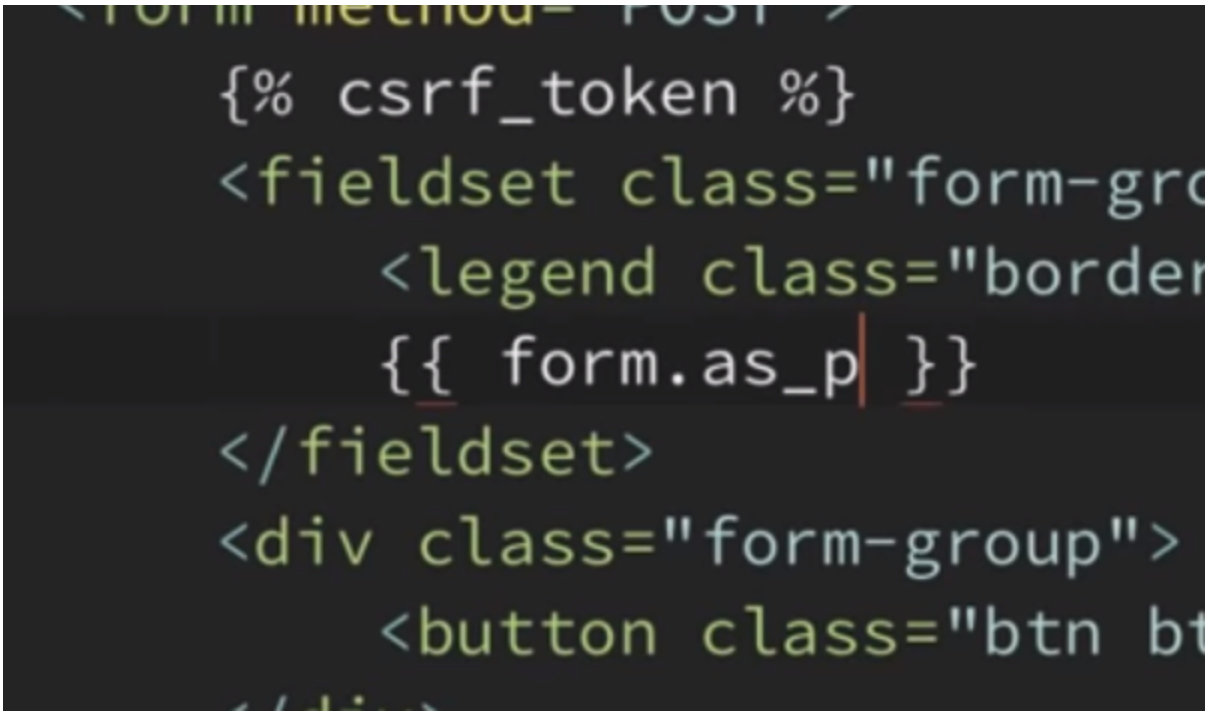
2) Your register.html template :

-here, you write the form method tags along with the csrf token, and you just render the built in django forms like this : {{form}} , and maybe give them bootstrap styles

-In case you forgot, the csrf token is a certificate given for the security of your POST method data so no mulware manipulates your post body data and your data goes safely to the server

```
{% extends "blog/base.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST">
            {% csrf_token %}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Join Today</legend>
                {{ form }}
            </fieldset>
            <div class="form-group">
                <button class="btn btn-outline-info" type="submit">Sign Up</
            </div>
        </form>
        <div class="border-top pt-3">
            <small class="text-muted">
                Already Have An Account? <a class="ml-2" href="#">Sign In</a
            </small>
        </div>
    </div>
{% endblock content %}
```

for better styling, you can do this : {{ form.as_p }} —> this “ as_p “ will display your form inputs as paragraph elements with spaces between each input



3) (Optional) : Add url of User app in project's urls.py

An important note :

Instead of adding your app url in the app urls.py then connecting it to the whole project in the project's urls.py,, you can do this only once by adding the URL of the app only once in the urls.py of the whole project by importing views there like this :

```python
from django.contrib import admin
from django.urls import path, include
from users import views as user_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('register/', user_views.register, name='register'),
    path('', include('blog.urls')),
]
```

Update your views.py to get and save data

Now, if you try to write your info in the form on the server and press submit, nothing will happen and you'll be redirected to the same page because you haven't wrote what to do with that information in the form,

so now in the views again :

- **you'll first tell the server to check whether the method is post or get**
- **if it's post , then …. if it's get then leave it as it is redirecting you to the same page**
- **if it's post, then**

  … make an instance of the form who's request method is post (form = …… (request.post))

  …. validate that your correct data are in the form

  ….if validation is true, then : - save the form using formsave() …. -get the username … -display a flash msg (msg that appears only once .. you import it from django.contrib and there 4 types : messages.success / messages.info / messages.warning / warning.error) …. don't forget to apply it in your base.html page so it is displyed in any other page

…. then redirect the user to a new page when logging in (home page for ex)

```python
from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm
from django.contrib import messages


def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            messages.success(request, f'Account created for {username}!')
            return redirect('blog-home')
    else:
        form = UserCreationForm()
    return render(request, 'users/register.html', {'form': form})
```

AND DON'T FORGET THE FORM.SAVE()

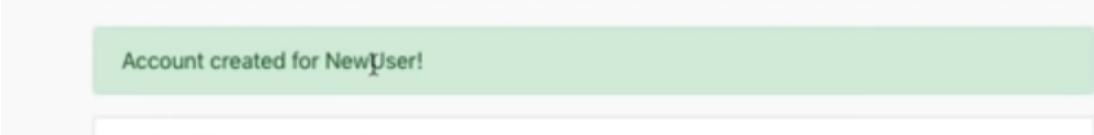
```python
from django.contrib import messages


def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Account created for {username}!')
            return redirect('blog-home')
    else:
        form = UserCreationForm()
    return render(request, 'users/register.html', {'form': form})
```

and display the message.success here in the base.html

```
    </nav>
  </header>
  <main role="main" class="container">
    <div class="row">
      <div class="col-md-8">
        {% if messages %}
          {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
              {{ message }}
            </div>
          {% endfor %}
        {% endif %}
        {% block content %}{% endblock %}
```

now, if you try signing up in the form, you'll get this message.success :



4) Create your desired form fields in forms.py

-Now what if you wanna add more fields to your forms ??

-Here in the template you render the whole django form {{ form }} instead of specific fields

-You'll have to create a new file in your User app that's called : forms.py

-There, you'll inherit from your User Model that django created from you , and you'll be able to add all the fields you want .. for EX : email field here ..

-class Meta : is a class of data about the data (aka forms) ..this is where you write the name of the model you inherit from , and the name of the fields you wanna display

In forms.py :

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


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

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

5) Apply the changes to your views.py

-NOW since you inherited from the User model, apply these changes (replace UserCreationForm with UserRegisterForm) to your Views.py :

In views.py :

```python
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import UserRegisterForm


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'Account created for {username}!')
            return redirect('blog-home')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})
```

6) Intsall Crispy Forms

This gives your django forms all the great styles without u manually writing them on your own.

1) pip install django-crispy-forms

2) Add it in the settings installed apps :

in your settings .py :

```
INSTALLED_APPS = [
    'blog.apps.BlogConfig',
    'users.apps.UsersConfig',
    'crispy_forms',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

3) **set the styling template that crispy forms will use : for ex : bootstrap 5 or 4**

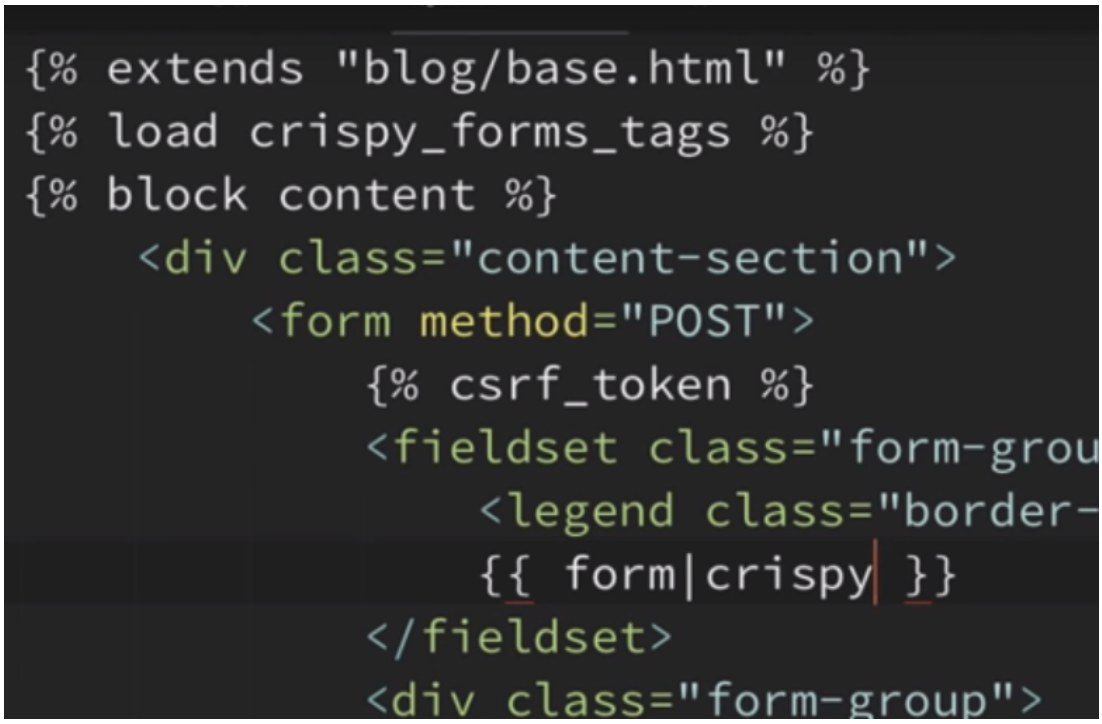
```
STATIC_URL = '/static/'

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

4) load the crispy forms template in your register.html **{% load crispy_forms_tags %}**

& replace it with the old **.as_p** **method so it becoms** **{{ form|crispy }}** **instead**

in your register.html :



NOTE : If this didn’t work with you, you can try the following because I had a problem with it and this fixed it :

33

I also ran to this problem but crispy-form is already supporting boostrap 5. In their git was instructed as so

```
$ pip install django-crispy-forms

$ pip install crispy-bootstrap5
```

And in settings.py

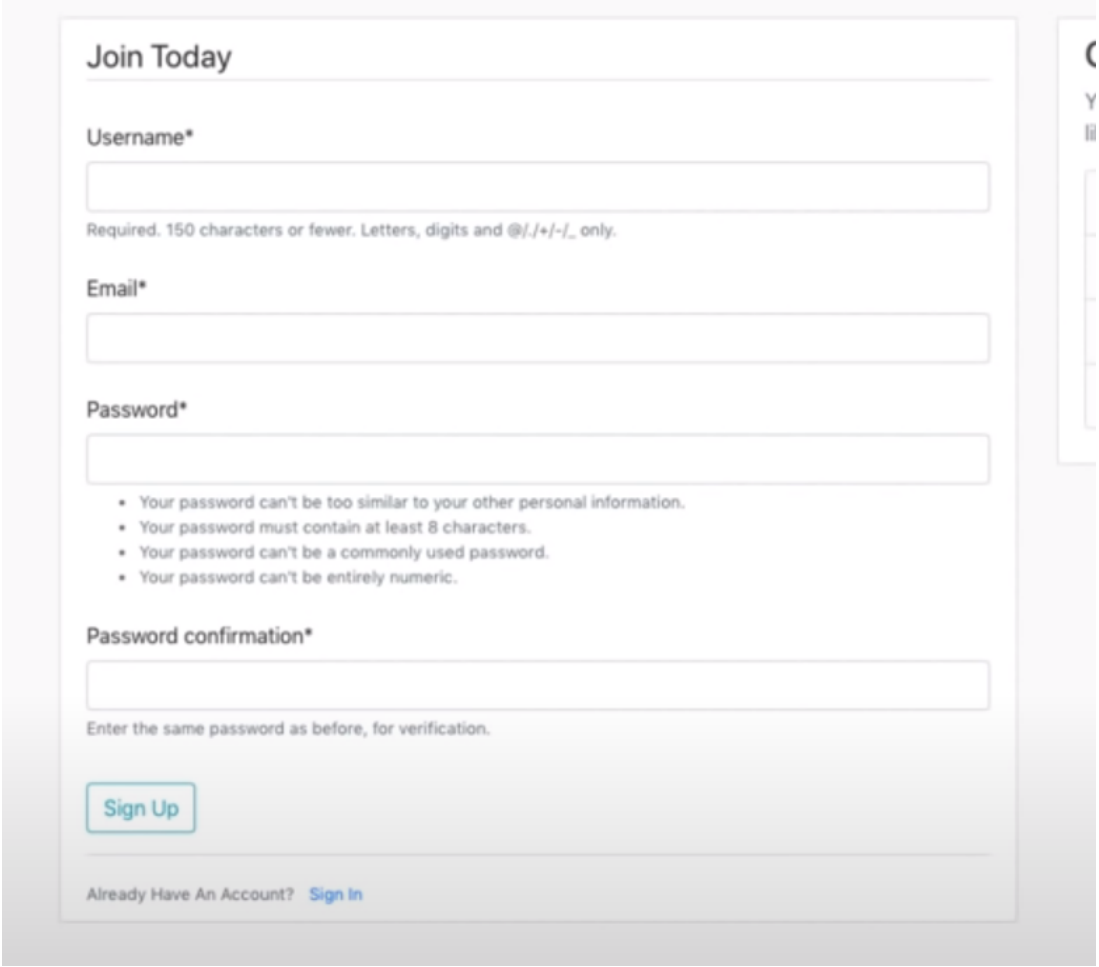
```
INSTALLED_APPS = [
 ...,
'crispy_forms',
'crispy_bootstrap5',  # Forgetting this was probably your error
 ]
```

And then at the bottom of the page of settings.py

```
CRISPY_ALLOWED_TEMPLATE_PACKS = "bootstrap5"
CRISPY_TEMPLATE_PACK = "bootstrap5"
```

This worked for me solving the TemplateDoesNotExist error. No need to downgrade bootstrap4

and now …. this is what your form will look like ：



SO … Summarizing them into 4 quick steps :

1) Forms.py

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


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

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

2) Views.py

```python
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import UserRegisterForm


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'Account created for {username}!')
            return redirect('blog-home')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})
```

3) Templates :

first .. register.html :

```
{% extends "blog/base.html" %}
{% load crispy_forms_tags %}
{% block content %}
    <div class="content-section">
        <form method="POST">
            {% csrf_token %}
            <fieldset class="form-group">
                <legend class="border-bottom mb-4">Join Today</legend>
                {{ form|crispy }}
            </fieldset>
            <div class="form-group">
                <button class="btn btn-outline-info" type="submit">Sign Up</
            </div>
        </form>
        <div class="border-top pt-3">
            <small class="text-muted">
                Already Have An Account? <a class="ml-2" href="#">Sign In</a
            </small>
        </div>
    </div>
{% endblock content %}
```

second .. base.html :

```
    </nav>
  </header>
  <main role="main" class="container">
    <div class="row">
      <div class="col-md-8">
        {% if messages %}
          {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
              {{ message }}
            </div>
          {% endfor %}
        {% endif %}
        {% block content %}{% endblock %}
```

Don’t forget to add the Project’s urls.py

```python
from django.contrib import admin
from django.urls import path, include
from users import views as user_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('register/', user_views.register, name='register'),
    path('', include('blog.urls')),
]
```

4) Intsall Crispy Forms

pip django-crispy-forms

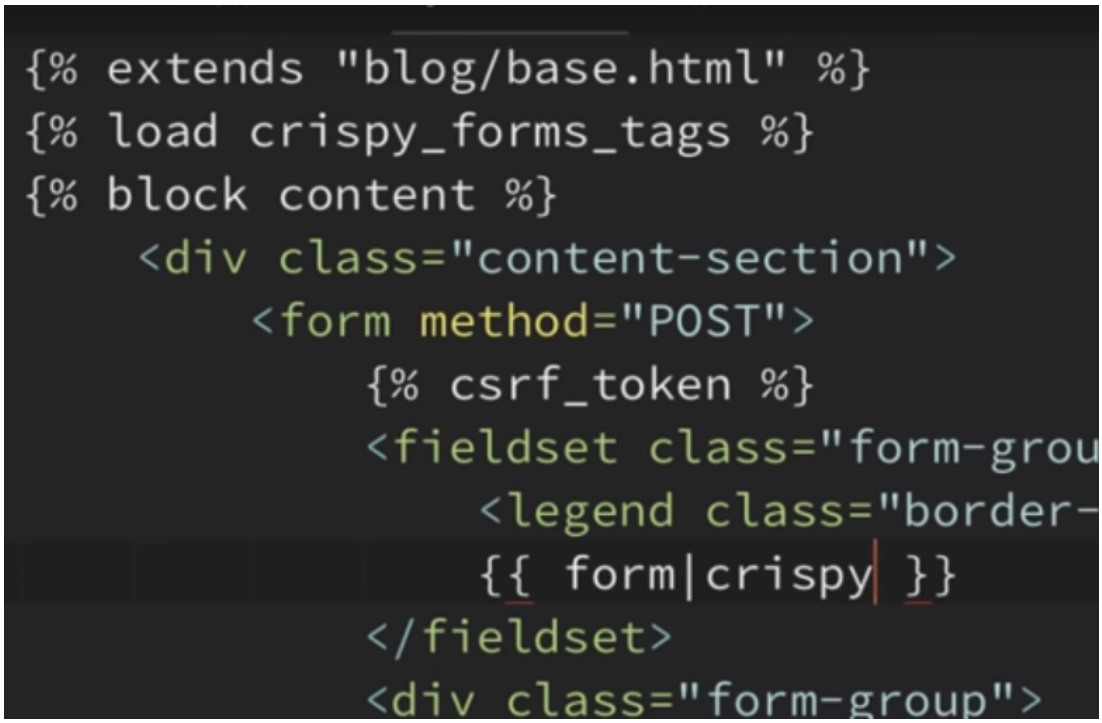
add User app & crispy app & crispy template

```
INSTALLED_APPS = [
    'blog.apps.BlogConfig',
    'users.apps.UsersConfig',
    'crispy_forms',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

```
STATIC_URL = '/static/'

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

Apply crispy forms tags to your templates


{% extends "blog/base.html" %}
{% load crispy_forms_tags %}
{% block content %}

<form method="POST">
{% csrf_token %}
<fieldset class="form-grou
<legend class="border-
{{ form|crispy }}
</fieldset>