Wexdi SoftSkills

Django Forms and User Input

Creating and rendering forms in Django.

Wexdi SoftSkills

- Creating a form in Django is completely similar to creating a model, one needs to specify what fields would exist in the form and of what type. For example, to input, a registration form one might need First Name (CharField), Roll Number (IntegerField), and so on.

Syntax:

```
from django import forms
class FormName(forms.Form):
        # each field would be mapped as an input field in HTML
    field_name = forms.Field(**options)
```

Creating and rendering forms in Django.

Wexdi SoftSkills

```
# import the standard Django Forms
# from built-in library
from django import forms
  # creating a form
class InputForm(forms.Form):
     first_name = forms.CharField(max_length = 200)
    last_name = forms.CharField(max_length = 200)
    roll_number = forms.IntegerField(
                    help_text = "Enter 6 digit roll number"
                    )
    password = forms.CharField(widget = forms.PasswordInput())
```

Creating and rendering forms in Django.

Wexdi SoftSkills

Django form fields have several built-in methods to ease the work of the developer but sometimes one needs to implement things manually for customizing User Interface(UI). A form comes with 3 in-built methods that can be used to render Django form fields.

{{ form.as_table }} will render them as table cells wrapped in <tr> tags

{{ form.as_p }} will render them wrapped in <p> tags

{{ form.as_ul }} will render them wrapped in <li> tags

To render this form into a view, move to views.py and create a home_view as below.

Creating and rendering forms in Django.

Wexdi SoftSkills

```
from django.shortcuts import render

from .forms import InputForm


# Create your views here.

def home_view(request):

    context ={}

    context['form']= InputForm()

    return render(request, "home.html", context)
```

Creating and rendering forms in Django.

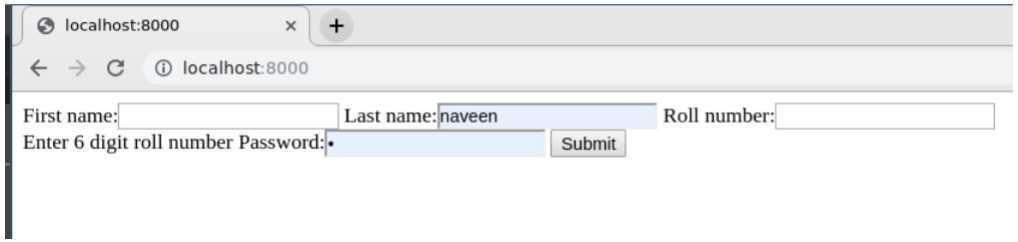
Wexdi SoftSkills

In view, one needs to just create an instance of the form class created above in forms.py.

Now let’s edit templates > home.html

```
<form action = "" method = "post">
    {% csrf_token %}
    {{form }}
    <input type="submit" value=Submit">
</form>
```

Now, visit http://localhost:8000/



Handling form submissions (GET vs POST).

Wexdi SoftSkills

- GET Method
  - Data in URL: GET sends form data as part of the URL and Information is visible in the browser's address bar.
  - Bookmarking and Caching: Form submissions with GET can be bookmarked and cached easily and it is useful for sharing links but not suitable for sensitive data.
  - Limit on Data Size: Using GET Limited data size for submission (typically up to 2048 characters) and this is ideal for small amounts of non-sensitive data.

    Syntax

```
<form action="/submit" method="GET">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username">
  <input type="submit" value="Submit">
</form>
```

Handling form submissions (GET vs POST).

Wexdi SoftSkills

- POST Method
  - Data in Request Body: POST sends form data in the body of the HTTP request and Information is not visible in the URL.
  - Security and Sensitivity: It is suitable for sensitive data like passwords and it is more secure as data is not exposed in the URL.
  - No Data Size Limit: No strict size limit for data submission and it is suitable for large amounts of information.

    Syntax

```
<form action="/submit" method="POST">
  <label for="password">Password:</label>
  <input type="password" id="password" name="password">
  <input type="submit" value="Submit">
</form>
```

Validation of form data.


```
def emp(request):
    if request.method == "POST":
        form = EmployeeForm(request.POST)
        if form.is_valid():
            try:
                return redirect('/')
            except:
                pass
        else:
            form = EmployeeForm()
    return render(request,'index.html',{'form':form})
```

Saving form data to the database.

Wexdi SoftSkills

```
if form.is_valid():
    name = form.cleaned_data['name']
    number = form.cleaned_data['phone_number']
    p = Person(name=name, phone_number=number,
date_subscribed=datetime.now(), messages_received=0)
    p.save()
...
```

The form.save() step is unnecessary.

```
')
```

Saving form data to the database.

Wexdi SoftSkills

You could also try using a ModelForm and some defaults or a custom save method, like

```
class Person(models.Model):
    name = models.CharField(max_length=96)
    phone_number = models.CharField(max_length=12)
    date_subscribed = models.DateTimeField(default=datetime.now())
    messages_received = models.IntegerField(default=0)

from django.forms import ModelForm

class SubscribeForm(ModelForm):
    class Meta:
        model = Person
        exclude = ('date_subscribed','messages_received')
```