Render Django Form Fields Manually

Last Updated : 22 Jul, 2021

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). We have already covered on How to create and use a form in Django?. 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

These render the form automatically but if you want to create a beautiful form with some CSS effects, you need to render the form fields manually. This article revolves around how to render the form fields manually.

Rendering Form fields manually

Illustration of **Rendering Django Forms Manually** using an Example. Consider a project named geeksforgeeks having an app named geeks.

> *Refer to the following articles to check how to create a project and an app in Django.*
>
> - *How to Create a Basic Project using MVT in Django?*
> - *How to Create an App in Django ?*

In your geeks app make a new file called forms.py where you would be making all your forms. To create a Django form you need to use Django Form Class. Let's demonstrate how,
In your forms.py Enter the following,

Python3

```
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())
```

Let's explain what exactly is happening, left side denotes the name of the field and to right of it, you define various functionalities of an input field correspondingly. A field's syntax is denoted as
Syntax :

```
Field_name = forms.FieldType(attributes)
```

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

Python3

```
from django.shortcuts import render
from .forms import InputForm
```

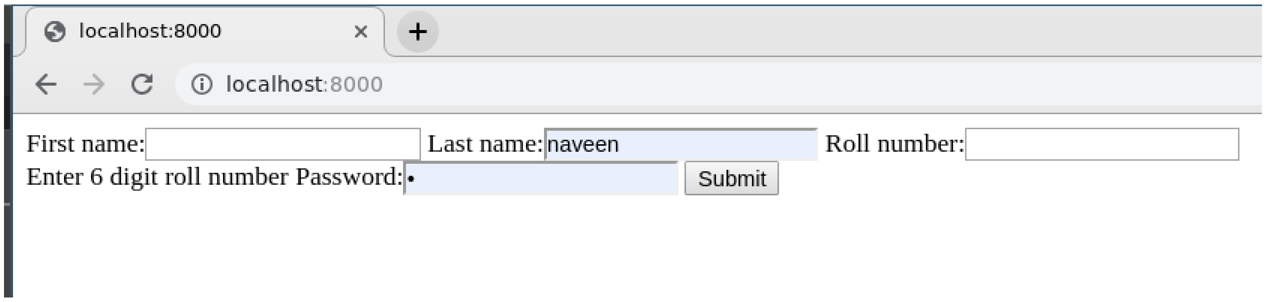
```
# Create your views here.
def home_view(request):
    context ={}
    context['form']= InputForm()
    return render(request, "home.html", context)
```

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

html

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

All set to check if form is working or not let's visit http://localhost:8000/



First name: Last name: naveen Roll number:
Enter 6 digit roll number Password: •

.

Form is working properly but visuals are disappointing, We can render these fields manually to improve some visual stuff. Each field is available as an attribute of the form using {{ form.name_of_field }}, and in a Django template, will be rendered appropriately. For example:

```
{{ form.non_field_errors }}
<div class="fieldWrapper">
    {{ form.subject.errors }}
    <label for="{{ form.subject.id_for_label }}">Email subject:</label>
    {{ form.subject }}
</div>
```

.

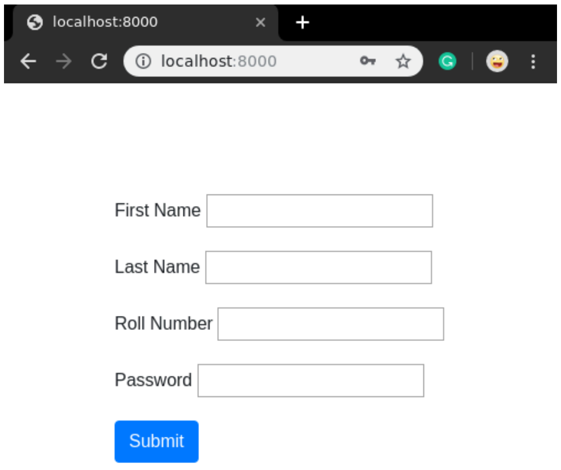
Let’s modify our form to look pretty impressive,

html

```
<html>

<head>
    <link
     rel="stylesheet"
     href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.cs
    <style>
        .i-am-centered {
            margin: auto;
            max-width: 300px;
            padding-top: 20%;
        }
    </style>
</head>

<body>
    <div class="i-am-centered">
        <form method="POST">
            {% csrf_token %}
            <div class="form-group">
                <label>First Name </label>
                {{ form.first_name }}
            </div>
            <div class="form-group">
                <label>Last Name </label>
                {{ form.last_name }}
            </div>
            <div class="form-group">
                <label>Roll Number</label>
                {{ form.roll_number }}
            </div>
            <div class="form-group">
```

```
                <label>Password</label>
                {{ form.password }}
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
</body>

</html>
```

Now visit http://localhost:8000/ and check modified form.



First Name

Last Name

Roll Number

Password

Submit

These were just some basic modifications using Bootstrap. One can customize it to an advanced level using various CSS tricks and methods.

{{ field }} attributes

- **{{ field.label }}**
  The label of the field, e.g. Email address.
- **{{ field.label_tag }}**
  The field’s label wrapped in the appropriate HTML tag. This includes the form’s label_suffix. For example, the default label_suffix is a colon:

```
<label for="id_email">Email address:</label>
```

- **{{ field.id_for_label }}**
  The ID that will be used for this field (id_email in the example above). If you are constructing the label manually, you may want to use this in place of label_tag. It's also useful, for example, if you have some inline JavaScript and want to avoid hardcoding the field's ID.
- **{{ field.value }}**
  The value of the field. e.g someone@example.com.
- **{{ field.html_name }}**
  The name of the field that will be used in the input element's name field. This takes the form prefix into account, if it has been set.
- **{{ field.help_text }}**
  Any help text that has been associated with the field.
- **{{ field.errors }}**
  Outputs a <ul class="errorlist"> containing any validation errors corresponding to this field. You can customize the presentation of the errors with a {% for error in field.errors %} loop. In this case, each object in the loop is a string containing the error message.
- **{{ field.is_hidden }}**
  This attribute is True if the form field is a hidden field and False otherwise. It's not particularly useful as a template variable, but could be useful in conditional tests such as:

```
{% if field.is_hidden %}
   {# Do something special #}
{% endif %}
```

- **{{ field.field }}**
  The Field instance from the form class that this BoundField wraps. You can use it to access Field attributes, e.g. {{ char_field.field.max_length }}

N Nave... 11

Previous Article

Python | Form validation using django

Next Article

Django URL patterns | Python

Similar Reads

{{ form.as_p }} - Render Django Forms as paragraph

Django forms are an advanced set of HTML forms that can be created using python and support all features of HTML forms in a pythonic way. Rendering...

2 min read

{{ form.as_table }} - Render Django Forms as table

Django forms are an advanced set of HTML forms that can be created using python and support all features of HTML forms in a pythonic way. Rendering...

2 min read

{{ form.as_ul }} - Render Django Forms as list

Django forms are an advanced set of HTML forms that can be created using python and support all features of HTML forms in a pythonic way. Rendering...

2 min read

Django Form | Data Types and Fields

When gathering user information to store in a database, we employ Django forms. Django offers a variety of model field forms for different uses, and these...

6 min read

Django Form | Build in Fields Argument

We utilize Django Forms to collect user data to put in a database. For various purposes, Django provides a range of model field forms with various field...

3 min read

Render Model in Django Admin Interface

Rendering model in admin refers to adding the model to the admin interface so that data can be manipulated easily using admin interface. Django's ORM...

3 min read

Render a HTML Template as Response - Django Views

A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response. This article revolves around how to render...

3 min read

Render HTML Forms (GET & POST) in Django

Django is often called "Batteries Included Framework" because it has a default setting for everything and has features that can help anyone develop a website...

5 min read

Intermediate fields in Django | Python

Prerequisite: Django models, Relational fields in DjangoIn Django, a many-to-many relationship exists between two models A and B, when one instance of A ...

2 min read

Python | Relational fields in Django models

Prerequisite: Django models Django models represent real-world entities, and it is rarely the case that real-world entities are entirely independent of each other....

4 min read

Article Tags : Python Django-forms Python Django

Practice Tags : python

Engineering Maths
Software Development
Software Testing

Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects