# Rendering Forms Manually in Django 5

# Code 28

# Video 34

By default, Django forms can be rendered as:

* {{ form }} → whole form in one line
* {{ form.as\_p }}, {{ form.as\_table }}, {{ form.as\_ul }} → built-in render styles
* But in real-world apps, we **render fields manually** for **control over styling and JS**.

## 1. **Rendering Form Manually – Step by Step**

### ✅ registration.html (first approach)

**<**!-- Render form fields **with** labels manually --**>**

**<**form action**=**""**>**

**<**!-- Using Django's id\_for\_label -->

**<**label **for**="{{form.first\_name.id\_for\_label}}"**>**{{form**.**first\_name**.**label}}**</**label**>**

  {{form**.**first\_name}}

**<**label **for**="{{form.last\_name.id\_for\_label}}"**>**{{form**.**last\_name**.**label}}**</**label**>**

  {{form**.**last\_name}}

**</**form**>**

🔎 **Explanation:**

* form.first\_name.label → Text of label (First name)
* form.first\_name.id\_for\_label → Generates correct id dynamically
* {{form.first\_name}} → The <input> field

👉 Using id\_for\_label ensures **labels always connect to inputs**, even if we later change auto\_id in view.

## 2. **Copying Input From "View Source"**

When you render {{form.first\_name}}, Django outputs:

**<**input *type***=**"text" name**=**"first\_name" required id**=**"id\_first\_name"**>**

So instead of {{form.first\_name}}, we can **copy & customize** it manually:

**<**form action**=**""**>**

**<**label **for**="{{form.first\_name.id\_for\_label}}"**>**{{form**.**first\_name**.**label}}**</**label**>**

**<**input

*type*="text"

    name="{{form.first\_name.name}}"

    id="{{form.first\_name.id\_for\_label}}"

    value="{{form.first\_name.value|default:'e.g. Talha'}}"

    required

**>**

**<**label **for**="{{form.last\_name.id\_for\_label}}"**>**{{form**.**last\_name**.**label}}**</**label**>**

**<**input

*type*="text"

    name="{{form.last\_name.name}}"

    id="{{form.last\_name.id\_for\_label}}"

    value="{{form.last\_name.value}}"

    required

**>**

**</**form**>**

🔎 **Why better?**

* We can add value, placeholder, class="..." (Tailwind, Bootstrap).
* Allows full control over **HTML structure**.

## 3. **Ways to Provide Default Values**

Django gives **3 ways** to set initial values:

### forms.py

**from** django **import** forms

class **Registration**(*forms***.***Form*)**:**

    first\_name **=** forms**.**CharField()

    last\_name **=** forms**.**CharField(**initial=**"Ali Alli")   *# Method 2: inside form*

    email **=** forms**.**EmailField()

    city **=** forms**.**CharField(**help\_text=**"Enter your current city") *# help\_text*

### views.py

**from** django**.**shortcuts **import** render

**from** student**.**forms **import** Registration

**def** registration(**request**)**:**

*# Method 3: from views (dynamic initial values)*

    fm **=** Registration(**auto\_id=**True**,** **initial=**{'email'**:** 'xyz@example.com'})

**return** render(request**,** 'student/registration.html'**,** {'form'**:** fm})

### registration.html

**<**form action**=**""**>**

**<**!-- Method 1**:** default **with** template filter --**>**

**<**input *type***=**"text" name**=**"{{form.first\_name.name}}"

         id**=**"{{form.first\_name.id\_for\_label}}"

         value**=**"{{form.first\_name.value|default:'e.g. Talha'}}"**>**

**<**!-- Method 2**:** **from** form**.**py initial --**>**

**<**input *type***=**"text" name**=**"{{form.last\_name.name}}"

         id**=**"{{form.last\_name.id\_for\_label}}"

         value**=**"{{form.last\_name.value}}"**>**

**<**!-- Method 3**:** **from** views**.**py initial --**>**

**<**input *type***=**"email" name**=**"{{form.email.name}}"

         id**=**"{{form.email.id\_for\_label}}"

         value**=**"{{form.email.value}}"**>**

**</**form**>**

## 4. **Useful Attributes for Each Field**

**<**div**>**

  {{form**.**first\_name**.**label}}          **<**!-- Label text ("First name") --**>**

  {{form**.**first\_name**.**value}}          **<**!-- Current value --**>**

  {{form**.**first\_name**.**html\_name}}      **<**!-- HTML name ("first\_name") --**>**

  {{form**.**first\_name**.**field**.**required}} **<**!-- True**/**False --**>**

  {{form**.**first\_name**.**help\_text}}      **<**!-- Help text --**>**

**</**div**>**

These are powerful because they let you **build fields dynamically**.

## 5. **Dynamic Rendering with Loops(easy way through loop)**

Reduce retendency

Instead of writing each input manually, we can loop:

### forms.py

class **Address**(*forms***.***Form*)**:**

    name **=** forms**.**CharField()

    city **=** forms**.**CharField()

    state **=** forms**.**CharField()

    pin\_code **=** forms**.**IntegerField()

views**.**py

**def** address(**request**)**:**

    fm **=** Address(**auto\_id=**True)

**return** render(request**,** 'student/address.html'**,** {'form'**:** fm})

address**.**html

**<**form action**=**""**>**

  {**%** **for** field **in** form **%**}

**<**label **for**="{{field.id\_for\_label}}"**>**{{field**.**label}}**</**label**>**

**<**input

*type*="{{ field.field.widget.input\_type }}"

      name="{{ field.name }}"

      id="{{ field.id\_for\_label }}"

      value="{{ field.value|default\_if\_none:'' }}"

      placeholder="{{ field.field.widget.attrs.placeholder|default:'' }}"

      {**%** **if** field**.**field**.**required **%**}required{**%** endif **%**}

**>**

  {**%** endfor **%**}

**</**form**>**

🔎 **Benefit** → Avoids code repetition and auto-handles new fields.

## 6. **Handling Hidden Fields**

Sometimes forms have **hidden fields** (e.g., CSRF tokens, keys).

### forms.py

class **Login**(*forms***.***Form*)**:**

    name **=** forms**.**CharField()

    password **=** forms**.**CharField(**widget=**forms**.**PasswordInput())

    key **=** forms**.**CharField(**widget=**forms**.**HiddenInput())

### views.py

**def** login(**request**)**:**

    fm **=** Login()

**return** render(request**,** 'student/login.html'**,** {'form'**:** fm})

### login.html

**<**form action**=**""**>**

**<**!-- Visible fields --**>**

  {**%** **for** field **in** form**.**visible\_fields **%**}

    {{field**.**label\_tag}}

    {{field}}

  {**%** endfor **%**}

**<**!-- Hidden fields --**>**

  {**%** **for** hidden **in** form**.**hidden\_fields **%**}

    {{hidden}}

  {**%** endfor **%**}

**</**form**>**

🔎 **Why?**

* visible\_fields → name, password
* hidden\_fields → key (keeps security logic intact)

## 7. **Why Manual Rendering?**

* Full control over:
  + HTML (custom <div>, <span>, etc.)
  + Tailwind / Bootstrap classes
  + JavaScript interactions (onchange, onclick, validation)
* Handle hidden\_fields separately
* Avoid dependency on Django’s default form rendering

## 8. **Extra Things You Might Miss**

1. **Widgets**: You can customize field input type via forms.CharField(widget=...).
2. **Errors**: Show form errors with {{field.errors}}.
3. **CSS Classes**: Add via widget attrs:

name **=** forms**.**CharField(**widget=**forms**.**TextInput(**attrs=**{'class'**:** 'form-input'}))

1. **Field Order**: By default fields are in the same order as declared in Form class.
2. **Fieldsets**: Can group fields manually using templates.

✅ **Summary (One-Liners for Revision):**

•   {{form}} → whole form

•   {{form**.**field}} → single field

•   field**.**id\_for\_label → id **for** **<**label**>**

•   field**.**value → current value

•   field**.**html\_name → HTML input name

•   field**.**help\_text → help text defined **in** form

•   form**.**visible\_fields **/** form**.**hidden\_fields → split handling

•   Looping through form → dynamic field rendering