**Customizing Django Form Error Fields – Complete Guide**

**Code file 33**

**Video file 39**

**🔹 1. Default Django Form Error Behavior**

By default, when you use Django forms, validation errors are displayed in **<ul class="errorlist">** with <li> items.

Example:

**<**ul class**=**"errorlist"**><**li**>**This field **is** required**.</**li**></**ul**>**

This happens when you render {{field.errors}} or {{form.as\_p}}.

**🔹 2. Customizing Error Messages in forms.py**

You can **override error messages** using the error\_messages dictionary in each field.

📂 **forms.py**

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

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

*# Here we override the default "This field is required." message*

    name **=** forms**.**CharField(**error\_messages=**{'required'**:** 'Name is required'})

    email **=** forms**.**EmailField(**error\_messages=**{'required'**:** 'Email is required'})

    roll **=** forms**.**IntegerField(**error\_messages=**{'required'**:** 'Roll is required'})

    passward **=** forms**.**CharField(

**error\_messages=**{'required'**:** 'Password is required'}**,**

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

    )

👉 **What happens here?**

* If the user leaves name empty → "Name is required" instead of "This field is required."

**🔹 3. Displaying Errors in Template**

📂 **register.html**

**<**form action**=**"" method**=**"POST" novalidate**>**

    {**%** csrf\_token **%**}

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

**<**div**>**

            {{field**.**label\_tag}} {{field}}

            {{field**.**errors}} **<**!-- Default shows **<**ul**><**li**>**Error**</**li**></**ul**>** --**>**

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

    {**%** endfor **%**}

**<**button**>**Submit**</**button**>**

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

👉 **Problem:** {{field.errors}} wraps errors in <ul><li>.  
👉 **Solution:** Use |striptags to remove HTML tags.

**🔹 4. Removing <ul><li> with striptags**

📂 **register.html**

**<**style**>**

**.**custom**-**error**-**css {

        color**:** rebeccapurple;

    }

**</**style**>**

**<**form action**=**"" method**=**"POST" novalidate**>**

    {**%** csrf\_token **%**}

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

**<**div**>**

            {{field**.**label\_tag}} {{field}}

**<**!-- Errors shown **as** plain text --**>**

**<**span class**=**"custom-error-css"**>**{{field**.**errors**|**striptags}}**</**span**>**

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

    {**%** endfor **%**}

**<**button**>**Submit**</**button**>**

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

👉 **Now:** Errors appear as plain text → "Name is required" (not inside <ul><li>).  
👉 This makes it easier to apply CSS.

**🔹 5. Handling Multiple Errors Per Field**

Sometimes you want **multiple validations** (e.g., min\_length, max\_length).

📂 **forms.py**

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

    name **=** forms**.**CharField(**error\_messages=**{'required'**:** 'Name is required'})

    email **=** forms**.**EmailField(

**error\_messages=**{'required'**:** 'Email is required'}**,**

**min\_length=**5**,** **max\_length=**50

    )

    roll **=** forms**.**IntegerField(**error\_messages=**{'required'**:** 'Roll is required'})

    passward **=** forms**.**CharField(

**error\_messages=**{'required'**:** 'Password is required'}**,**

**widget=**forms**.**PasswordInput()**,**

**min\_length=**5**,** **max\_length=**50

    )

📂 **register.html**

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

**<**div**>**

        {{field**.**label\_tag}} {{field}}

**<**!-- Loop through each error --**>**

        {**%** **for** error **in** field**.**errors **%**}

**<**span class**=**"custom-error-css"**>**{{error}}**</**span**>**

        {**%** endfor **%**}

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

{**%** endfor **%**}

👉 **Why?** Because if email is empty **and** too short, multiple errors appear:

* "Email is required"
* "Ensure this value has at least 5 characters"

**🔹 6. Non-field Errors (form.non\_field\_errors)**

Some validations are **not tied to a single field**, e.g., "Password and Confirm Password don’t match".

📂 **register.html**

**<**form action**=**"" method**=**"POST" novalidate**>**

    {**%** csrf\_token **%**}

**<**!-- Show non**-**field errors at the top --**>**

    {{form**.**non\_field\_errors}}

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

**<**div**>**

            {{field**.**label\_tag}} {{field}}

            {**%** **for** error **in** field**.**errors **%**}

**<**span class**=**"custom-error-css"**>**{{error}}**</**span**>**

            {**%** endfor **%**}

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

    {**%** endfor **%**}

**<**button**>**Submit**</**button**>**

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

👉 Use this for **cross-field validation** (done in clean() method in forms).

**🔹 7. Changing Default Errorlist Class**

By default, Django wraps errors in <ul class="errorlist">.

You can **customize this class** with CSS:

📂 **register.html**

**<**style**>**

**.**errorlist {

        color**:** green;

        font**-**weight**:** bold;

    }

**</**style**>**

{{form**.**as\_p}}

👉 Now <ul class="errorlist"> will be styled in green.

**🔹 8. Using error\_css\_class & required\_css\_class**

In Django forms, you can **add CSS classes** automatically to error fields and required fields.

📂 **forms.py**

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

    error\_css\_class **=** 'my-error'      *# Adds this class to fields with errors*

    required\_css\_class **=** 'required'   *# Adds this class to required fields*

    name **=** forms**.**CharField(**error\_messages=**{'required'**:**'Name is required'})

    email **=** forms**.**EmailField(

**error\_messages=**{'required'**:**'Email is required'}**,**

**min\_length=**5**,** **max\_length=**50

    )

    roll **=** forms**.**IntegerField(**error\_messages=**{'required'**:**'Roll is required'})

    passward **=** forms**.**CharField(

**error\_messages=**{'required'**:**'Password is required'}**,**

**widget=**forms**.**PasswordInput()**,**

**min\_length=**5**,** **max\_length=**50

    )

📂 **register.html**

**<**style**>**

**.**my**-**error {

        border**:** 2px solid red;  **/\*** Highlight input box **\*/**

    }

**.**required {

        font**-**weight**:** bold;

        color**:** blue;

    }

**</**style**>**

{{form**.**as\_p}}

👉 **Page source looks like:**

**<**ul class**=**"errorlist" id**=**"id\_email\_error"**><**li**>**Email **is** required**</**li**></**ul**>**

**<**p class**=**"my-error required"**>**

**<**label **for**="id\_email" **class**="required"**>**Email**:</**label**>**

**<**input *type*="email" name="email" required aria**-**invalid="true"**>**

**</**p**>**

* Fields with errors get .my-error
* Required fields get .required

**🔹 9. Views – Handling Form Submission**

📂 **views.py**

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

**from** django**.**http **import** HttpResponseRedirect

**from** student**.**forms **import** Profile

**def** Register(**request**)**:**

**if** request**.**method **==** 'POST'**:**

        form **=** Profile(request**.**POST)

**if** form**.**is\_valid()**:**

*# Access cleaned data*

            name **=** form**.**cleaned\_data['name']

            email **=** form**.**cleaned\_data['email']

            roll **=** form**.**cleaned\_data['roll']

            passward **=** form**.**cleaned\_data['passward']

*# Print to console (or save to DB)*

            print("Name:"**,** name)

            print("Email:"**,** email)

            print("Roll:"**,** roll)

            print("Password:"**,** passward)

**return** HttpResponseRedirect('/student/register/')

**else:**

        form **=** Profile()

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

**🔹 10. Extra Notes (Things You Missed)**

✅ You can **customize error messages globally** in settings.py using translations.  
✅ You can use **Bootstrap classes** with forms.Form by overriding widgets:

name **=** forms**.**CharField(

**error\_messages=**{'required'**:** 'Name is required'}**,**

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

)

✅ For **cross-field validation**, override clean():

**def** clean(*self*)**:**

    cleaned\_data **=** super()**.**clean()

    email **=** cleaned\_data**.**get("email")

    roll **=** cleaned\_data**.**get("roll")

**if** roll **==** 123 **and** email**.**endswith("@test.com")**:**

**raise** forms**.**ValidationError("Invalid combination of roll and email")

✅ Use **as\_p, as\_table, as\_ul** for different HTML rendering formats.

**🎯 Conclusion**

In Django, customizing form error fields involves:

1. Using error\_messages in fields.
2. Controlling error display with {{field.errors}}, |striptags, and loops.
3. Handling non-field errors with {{form.non\_field\_errors}}.
4. Styling with .errorlist class.
5. Using error\_css\_class & required\_css\_class for automatic highlighting.
6. Optionally overriding widgets for full CSS framework integration.