Form Field

A form's fields are themselves classes; they manage form data and perform validation when a form is submitted.

Syntax:- FieldType(**kwargs)

Example:-

IntegerField()

CharField(required)

CharField(required, widget=forms.PasswordInput)

```
from django import forms
class StudentRegistration(forms.Form):
 name = forms.CharField()
```

Field Arguments

required - It take True or False value. By default, each Field class assumes the required value is True.

label - The label argument lets you specify the “human-friendly” label for the field. This is used when the Field is displayed in a Form.

label_suffix - The label_suffix argument lets you override the form’s label_suffix on a per-field basis.

initial - The initial argument lets you specify the initial value to use when rendering this Field in an unbound Form.

disabled - The disabled boolean argument, when set to True, disables a form field using the disabled HTML attribute so that it won’t be editable by users. Even if a user tampers with the field’s value submitted to the server, it will be ignored in favor of the value from the form’s initial data.

Field Arguments

help_text - The help_text argument lets you specify descriptive text for this Field. If you provide help_text, it will be displayed next to the Field when the Field is rendered.

error_messages - The error_messages argument lets you override the default messages that the field will raise. Pass in a dictionary with keys matching the error messages you want to override.

Example:- name=forms.CharField(error_messages={'required':'Enter Your Name’})

validators - The validators argument lets you provide a list of validation functions for the field.

localize - The localize argument enables the localization of form data input, as well as the rendered output.

widget - The widget argument lets you specify a Widget class to use when rendering the Field.

Widget

A widget is Django's representation of an HTML input element.

The widget handles the rendering of the HTML, and the extraction of data from a GET/POST dictionary that corresponds to the widget.

The HTML generated by the built-in widgets uses HTML5 syntax, targeting <!DOCTYPE html>.

Whenever you specify a field on a form, Django will use a default widget that is appropriate to the type of data that is to be displayed.

Each field type has an appropriate default Widget class, but these can be overridden as required.

Form fields deal with the logic of input validation and are used directly in templates.

Widgets deal with rendering of HTML form input elements on the web page and extraction of raw submitted data.

Example:-

TextInput

Textarea

Widget

attrs - A dictionary containing HTML attributes to be set on the rendered widget.

If you assign a value of True or False to an attribute, it will be rendered as an HTML5 boolean attribute.

Example:-

feedback=forms.CharField(widget=forms.TextInput(attrs={'class':'somecss1 somecss2', 'id':'uniqueid'}))

Built-in Widgets

TextInput - It renders as: <input type="text" ...>

Example:- name = forms.CharField(widget=forms.TextInput)

NumberInput - It renders as: <input type="number" ...>

EmailInput - It renders as: <input type="email" ...>

URLInput – It renders as: <input type="url" ...>

PasswordInput - It renders as: <input type="password" ...>

It take one optional argument *render_value* which determines whether the widget will have a value filled in when the form is re-displayed after a validation error (default is False).

Built-in Widgets

HiddenInput - It renders as: <input type="hidden" ...>

DateInput - It renders as: <input type="text" ...>

It take one optional argument *format*. The format in which this field's initial value will be displayed.

If no *format* argument is provided, the default format is the first format found in DATE_INPUT_FORMATS.

DateTimeInput - It renders as: <input type="text" ...>

It take one optional argument *format*. The format in which this field's initial value will be displayed.

If no *format* argument is provided, the default format is the first format found in DATETIME_INPUT_FORMATS.

By default, the microseconds part of the time value is always set to 0. If microseconds are required, use a subclass with the supports_microseconds attribute set to True.

Built-in Widgets

TimeInput - It renders as: <input type="text" ...>

It take one optional argument *format*. The format in which this field's initial value will be displayed.

If no *format* argument is provided, the default format is the first format found in TIME_INPUT_FORMATS

Textarea - It renders as: <textarea>...</textarea>

CheckboxInput - It renders as: <input type="checkbox" ...>

It takes one optional argument *check_test* which is a callable that takes the value of the CheckboxInput and returns True if the checkbox should be checked for that value.

Built-in Widgets

Select - It renders as: <select><option ...>...</select>

choices attribute is optional when the form field does not have a choices attribute. If it does, it will override anything you set here when the attribute is updated on the Field.

NullBooleanSelect - Select widget with options ‘Unknown’, ‘Yes’ and ‘No’

SelectMultiple - Similar to Select, but allows multiple selection: <select multiple>...</select>

RadioSelect - Similar to Select, but rendered as a list of radio buttons within <li> tags:

```
<ul>
  <li><input type="radio" name="..."></li>
  ...
</ul>
```

Built-in Widgets

CheckboxSelectMultiple - Similar to SelectMultiple, but rendered as a list of checkboxes:

```
<ul>
  <li><input type="checkbox" name="..." ></li>
  ...
</ul>
```

FileInput - It renders as: <input type="file" ...>

ClearableFileInput - It renders as: <input type="file" ...> with an additional checkbox input to clear the field’s value, if the field is not required and has initial data.

Built-in Widgets

MultipleHiddenInput – It renders as: multiple <input type="hidden" ...> tags

A widget that handles multiple hidden widgets for fields that have a list of values.

choices - This attribute is optional when the form field does not have a choices attribute. If it does, it will override anything you set here when the attribute is updated on the Field.

SplitDateTimeWidget - Wrapper (using MultiWidget) around two widgets: DateInput for the date, and TimeInput for the time. Must be used with SplitDateTimeField rather than DateTimeField.

SplitDateTimeWidget has several optional arguments:

date_format Similar to DateInput.format

time_format Similar to TimeInput.format

date_attrs and *time_attrs* Similar to Widget.attrs. A dictionary containing HTML attributes to be set on the rendered DateInput and TimeInput widgets, respectively. If these attributes aren't set, Widget.attrs is used instead.

Built-in Widgets

SplitHiddenDateTimeWidget - Similar to SplitDateTimeWidget, but uses HiddenInput for both date and time.

SelectDateWidget - Wrapper around three Select widgets: one each for month, day, and year.

It takes several optional arguments:

years - An optional list/tuple of years to use in the “year” select box. The default is a list containing the current year and the next 9 years.

months - An optional dict of months to use in the “months” select box.

empty_label - If the DateField is not required, SelectDateWidget will have an empty choice at the top of the list (default). You can change the text of this label with the *empty_label* attribute.

empty_label can be a string, list, or tuple. When a string is used, all select boxes will each have an empty choice with this label. If empty_label is a list or tuple of 3 string elements, the select boxes will have their own custom label. The labels should be in this order ('year_label', 'month_label', 'day_label').