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()
```

Built-in Fields

CharField(**kwargs)

Default widget: TextInput

Empty value: Whatever you've given as empty_value.

Normalizes to: A string.

Uses MaxLengthValidator and MinLengthValidator if max_length and min_length are provided. Otherwise, all inputs are valid.

Error message keys: required, max_length, min_length

It has three optional arguments for validation:

max_length and *min_length* - If provided, these arguments ensure that the string is at most or at least the given length.

strip - If True (default), the value will be stripped of leading and trailing whitespace.

empty_value - The value to use to represent "empty". Defaults to an empty string.

Example:- name=forms.CharField(min_length=5, max_length=50, error_messages={'required':'Enter Your Name'})

Built-in Fields

BooleanField(**kwargs)

Default widget: CheckboxInput

Empty value: False

Normalizes to: A Python True or False value.

Validates that the value is True (e.g. the check box is checked) if the field has required=True.

Error message keys: required

Built-in Fields

IntegerField(**kwargs)

Default widget: NumberInput when Field.localize is False, else TextInput.

Empty value: None

Normalizes to: A Python integer.

Validates that the given value is an integer. Uses MaxValueValidator and MinValueValidator if max_value and min_value are provided. Leading and trailing whitespace is allowed, as in Python's int() function.

Error message keys: required, invalid, max_value, min_value

The max_value and min_value error messages may contain %(limit_value)s, which will be substituted by the appropriate limit.

It takes two optional arguments for validation:

max_value and *min_value* - These control the range of values permitted in the field.

Built-in Fields

DecimalField(**kwargs)

Default widget: NumberInput when Field.localize is False, else TextInput.

Empty value: None

Normalizes to: A Python decimal.

Validates that the given value is a decimal. Uses MaxValueValidator and MinValueValidator if max_value and min_value are provided. Leading and trailing whitespace is ignored.

Error message keys: required, invalid, max_value, min_value, max_digits, max_decimal_places, max_whole_digits

The max_value and min_value error messages may contain %(limit_value)s, which will be substituted by the appropriate limit. Similarly, the max_digits, max_decimal_places and max_whole_digits error messages may contain %(max)s.

It takes four optional arguments:

max_value and *min_value* - These control the range of values permitted in the field, and should be given as decimal.Decimal values.

max_digits - The maximum number of digits (those before the decimal point plus those after the decimal point, with leading zeros stripped) permitted in the value.

decimal_places - The maximum number of decimal places permitted.

Built-in Fields

FloatField(**kwargs)

Default widget: NumberInput when Field.localize is False, else TextInput.

Empty value: None

Normalizes to: A Python float.

Validates that the given value is a float. Uses MaxValueValidator and MinValueValidator if max_value and min_value are provided. Leading and trailing whitespace is allowed, as in Python's float() function.

Error message keys: required, invalid, max_value, min_value

It takes two optional arguments for validation, *max_value* and *min_value*. These control the range of values permitted in the field.

Built-in Fields

SlugField(**kwargs)

Default widget: TextInput

Empty value: '' (an empty string)

Normalizes to: A string.

Uses validate_slug or validate_unicode_slug to validate that the given value contains only letters, numbers, underscores, and hyphens.

Error messages: required, invalid

This field is intended for use in representing a model SlugField in forms.

It takes an optional parameter:

allow_Unicode - A boolean instructing the field to accept Unicode letters in addition to ASCII letters. Defaults to False.

Built-in Fields

EmailField(**kwargs)

Default widget: EmailInput

Empty value: '' (an empty string)

Normalizes to: A string.

Uses EmailValidator to validate that the given value is a valid email address, using a moderately complex regular expression.

Error message keys: required, invalid

It has two optional arguments for validation, *max_length* and *min_length*. If provided, these arguments ensure that the string is at most or at least the given length.

Built-in Fields

URLField(**kwargs)

Default widget: URLInput

Empty value: '' (an empty string)

Normalizes to: A string.

Uses URLValidator to validate that the given value is a valid URL.

Error message keys: required, invalid

It takes the following optional arguments:

max_length and *min_length* - These are the same as CharField.max_length and CharField.min_length.

Built-in Fields

DateField(**kwargs)

Default widget: DateInput

Empty value: None

Normalizes to: A Python datetime.date object.

Validates that the given value is either a datetime.date, datetime.datetime or string formatted in a particular date format.

Error message keys: required, invalid

It takes one optional argument:

input_formats - A list of formats used to attempt to convert a string to a valid datetime.date object.

If no input_formats argument is provided, the default input formats are taken from DATE_INPUT_FORMATS if USE_L10N is False, or from the active locale format DATE_INPUT_FORMATS key if localization is enabled. See also format localization.

Built-in Fields

TimeField(**kwargs)

Default widget: TimeInput

Empty value: None

Normalizes to: A Python datetime.time object.

Validates that the given value is either a datetime.time or string formatted in a particular time format.

Error message keys: required, invalid

It takes one optional argument:

input_formats - A list of formats used to attempt to convert a string to a valid datetime.time object.

If no input_formats argument is provided, the default input formats are taken from TIME_INPUT_FORMATS if USE_L10N is False, or from the active locale format TIME_INPUT_FORMATS key if localization is enabled. See also format localization.

Built-in Fields

DateTimeField(**kwargs)

Default widget: DateTimeInput

Empty value: None

Normalizes to: A Python datetime.datetime object.

Validates that the given value is either a datetime.datetime, datetime.date or string formatted in a particular datetime format.

Error message keys: required, invalid

It takes one optional argument:

input_formats - A list of formats used to attempt to convert a string to a valid datetime.datetime object.

If no input_formats argument is provided, the default input formats are taken from DATETIME_INPUT_FORMATS if USE_L10N is False, or from the active locale format DATETIME_INPUT_FORMATS key if localization is enabled. See also format localization.

Built-in Fields

DurationField(**kwargs)

Default widget: TextInput

Empty value: None

Normalizes to: A Python timedelta.

Validates that the given value is a string which can be converted into a timedelta. The value must be between datetime.timedelta.min and datetime.timedelta.max.

Error message keys: required, invalid, overflow.

Accepts any format understood by parse_duration().

Built-in Fields

FileField(**kwargs)

Default widget: ClearableFileInput

Empty value: None

Normalizes to: An UploadedFile object that wraps the file content and file name into a single object.

Can validate that non-empty file data has been bound to the form.

Error message keys: required, invalid, missing, empty, max_length

It has two optional arguments for validation, *max_length* and *allow_empty_file*. If provided, these ensure that the file name is at most the given length, and that validation will succeed even if the file content is empty.

To learn more about the UploadedFile object.

When you use a FileField in a form, you must also remember to bind the file data to the form.

The max_length error refers to the length of the filename. In the error message for that key, %(max)d will be replaced with the maximum filename length and %(length)d will be replaced with the current filename length.

Built-in Fields

FilePathField(**kwargs)

Default widget: Select

Empty value: '' (an empty string)

Normalizes to: A string.

Validates that the selected choice exists in the list of choices.

Error message keys: required, invalid_choice

The field allows choosing from files inside a certain directory. It takes five extra arguments; only path is required:

path - The absolute path to the directory whose contents you want listed. This directory must exist.

recursive - If False (the default) only the direct contents of path will be offered as choices. If True, the directory will be descended into recursively and all descendants will be listed as choices.

match - A regular expression pattern; only files with names matching this expression will be allowed as choices.

allow_files - Optional. Either True or False. Default is True. Specifies whether files in the specified location should be included. Either this or allow_folders must be True.

allow_folders - Optional. Either True or False. Default is False. Specifies whether folders in the specified location should be included. Either this or allow_files must be True.

Built-in Fields

ImageField(**kwargs)

Default widget: ClearableFileInput

Empty value: None

Normalizes to: An UploadedFile object that wraps the file content and file name into a single object.

Validates that file data has been bound to the form. Also uses FileExtensionValidator to validate that the file extension is supported by Pillow.

Error message keys: required, invalid, missing, empty, invalid_image

Using an ImageField requires that Pillow is installed with support for the image formats you use.

Built-in Fields

ChoiceField(**kwargs)

Default widget: Select

Empty value: '' (an empty string)

Normalizes to: A string.

Validates that the given value exists in the list of choices.

Error message keys: required, invalid_choice

The invalid_choice error message may contain %(value)s, which will be replaced with the selected choice.

It takes one extra argument:

choices - Either an iterable of 2-tuples to use as choices for this field, or a callable that returns such an iterable. This argument accepts the same formats as the choices argument to a model field. See the model field reference documentation on choices for more details. If the argument is a callable, it is evaluated each time the field's form is initialized. Defaults to an empty list.

Built-in Fields

TypedChoiceField(**kwargs) - Just like a ChoiceField, except TypedChoiceField takes two extra arguments, coerce and empty_value.

Default widget: Select

Empty value: Whatever you've given as empty_value.

Normalizes to: A value of the type provided by the coerce argument.

Validates that the given value exists in the list of choices and can be coerced.

Error message keys: required, invalid_choice

It takes extra arguments:

coerce - A function that takes one argument and returns a coerced value. Examples include the built-in int, float, bool and other types. Defaults to an identity function. Note that coercion happens after input validation, so it is possible to coerce to a value not present in choices.

empty_value - The value to use to represent "empty." Defaults to the empty string; None is another common choice here. Note that this value will not be coerced by the function given in the coerce argument, so choose it accordingly.

Built-in Fields

UUIDField(**kwargs)

Default widget: TextInput

Empty value: '' (an empty string)

Normalizes to: A UUID object.

Error message keys: required, invalid

This field will accept any string format accepted as the hex argument to the UUID constructor.

Built-in Fields

ComboField(**kwargs)

Default widget: TextInput

Empty value: '' (an empty string)

Normalizes to: A string.

Validates the given value against each of the fields specified as an argument to the ComboField.

Error message keys: required, invalid

It takes one extra required argument:

fields - The list of fields that should be used to validate the field’s value (in the order in which they are provided).

Built-in Fields

GenericIPAddressField(**kwargs) - A field containing either an IPv4 or an IPv6 address.

Default widget: TextInput

Empty value: '' (an empty string)

Normalizes to: A string. IPv6 addresses are normalized as described below.

Validates that the given value is a valid IP address.

Error message keys: required, invalid

The IPv6 address normalization follows RFC 4291#section-2.2 section 2.2, including using the IPv4 format suggested in paragraph 3 of that section, like ::ffff:192.0.2.0. For example, 2001:0::0:01 would be normalized to 2001::1, and ::ffff:0a0a:0a0a to ::ffff:10.10.10.10. All characters are converted to lowercase.

It takes two optional arguments:

protocol - Limits valid inputs to the specified protocol. Accepted values are both (default), IPv4 or IPv6. Matching is case insensitive.

unpack_ipv4 - Unpacks IPv4 mapped addresses like ::ffff:192.0.2.1. If this option is enabled that address would be unpacked to 192.0.2.1. Default is disabled. Can only be used when protocol is set to 'both'.

Built-in Fields

MultipleChoiceField(**kwargs)

Default widget: SelectMultiple

Empty value: [] (an empty list)

Normalizes to: A list of strings.

Validates that every value in the given list of values exists in the list of choices.

Error message keys: required, invalid_choice, invalid_list

The invalid_choice error message may contain %(value)s, which will be replaced with the selected choice.

It takes one extra required argument, choices, as for ChoiceField.

Built-in Fields

TypedMultipleChoiceField(**kwargs)

Just like a MultipleChoiceField, except TypedMultipleChoiceField takes two extra arguments, coerce and empty_value.

Default widget: SelectMultiple

Empty value: Whatever you’ve given as empty_value

Normalizes to: A list of values of the type provided by the coerce argument.

Validates that the given values exists in the list of choices and can be coerced.

Error message keys: required, invalid_choice

The invalid_choice error message may contain %(value)s, which will be replaced with the selected choice.

It takes two extra arguments, coerce and empty_value, as for TypedChoiceField.

Built-in Fields

NullBooleanField(**kwargs)

Default widget: NullBooleanSelect

Empty value: None

Normalizes to: A Python True, False or None value.

Validates nothing (i.e., it never raises a ValidationError).

Built-in Fields

RegexField(**kwargs)

Default widget: TextInput

Empty value: '' (an empty string)

Normalizes to: A string.

Uses RegexValidator to validate that the given value matches a certain regular expression.

Error message keys: required, invalid

Takes one required argument:

regex - A regular expression specified either as a string or a compiled regular expression object.

Also takes max_length, min_length, and strip, which work just as they do for CharField.

strip - Defaults to False. If enabled, stripping will be applied before the regex validation.

Built-in Fields

MultiValueField(fields=(), **kwargs)

Default widget: TextInput

Empty value: '' (an empty string)

Normalizes to: the type returned by the compress method of the subclass.

Validates the given value against each of the fields specified as an argument to the MultiValueField.

Error message keys: required, invalid, incomplete

Aggregates the logic of multiple fields that together produce a single value.

This field is abstract and must be subclassed. In contrast with the single-value fields, subclasses of MultiValueField must not implement clean() but instead - implement compress().

Takes one extra required argument:

fields - A tuple of fields whose values are cleaned and subsequently combined into a single value. Each value of the field is cleaned by the corresponding field in fields – the first value is cleaned by the first field, the second value is cleaned by the second field, etc. Once all fields are cleaned, the list of clean values is combined into a single value by compress().

Built-in Fields

Also takes some optional arguments:

require_all_fields - Defaults to True, in which case a required validation error will be raised if no value is supplied for any field.

When set to False, the Field.required attribute can be set to False for individual fields to make them optional. If no value is supplied for a required field, an incomplete validation error will be raised.

A default incomplete error message can be defined on the MultiValueField subclass, or different messages can be defined on each individual field.

widget - Must be a subclass of django.forms.MultiWidget. Default value is TextInput, which probably is not very useful in this case.

compress(data_list) - Takes a list of valid values and returns a “compressed” version of those values – in a single value. For example, SplitDateTimeField is a subclass which combines a time field and a date field into a datetime object.

This method must be implemented in the subclasses.

Built-in Fields

SplitDateTimeField(**kwargs)

Default widget: SplitDateTimeWidget

Empty value: None

Normalizes to: A Python datetime.datetime object.

Validates that the given value is a datetime.datetime or string formatted in a particular datetime format.

Error message keys: required, invalid, invalid_date, invalid_time

Takes two optional arguments:

input_date_formats - A list of formats used to attempt to convert a string to a valid datetime.date object.

If no input_date_formats argument is provided, the default input formats for DateField are used.

input_time_formats - A list of formats used to attempt to convert a string to a valid datetime.time object.

If no input_time_formats argument is provided, the default input formats for TimeField are used.

Built-in Fields

ModelChoiceField(**kwargs)

Default widget: Select

Empty value: None

Normalizes to: A model instance.

Validates that the given id exists in the queryset.

Error message keys: required, invalid_choice

Allows the selection of a single model object, suitable for representing a foreign key. Note that the default widget for ModelChoiceField becomes impractical when the number of entries increases. You should avoid using it for more than 100 items.

A single argument is required:

queryset - A QuerySet of model objects from which the choices for the field are derived and which is used to validate the user’s selection. It’s evaluated when the form is rendered.

Built-in Fields

ModelChoiceField also takes two optional arguments:

empty_label - By default the <select> widget used by ModelChoiceField will have an empty choice at the top of the list. You can change the text of this label (which is "---------" by default) with the empty_label attribute, or you can disable the empty label entirely by setting empty_label to None:

Note that if a ModelChoiceField is required and has a default initial value, no empty choice is created (regardless of the value of empty_label).

to_field_name - This optional argument is used to specify the field to use as the value of the choices in the field's widget. Be sure it's a unique field for the model, otherwise the selected value could match more than one object. By default it is set to None, in which case the primary key of each object will be used.

The __str__() method of the model will be called to generate string representations of the objects for use in the field's choices. To provide customized representations, subclass ModelChoiceField and override label_from_instance. This method will receive a model object and should return a string suitable for representing it.

Built-in Fields

ModelMultipleChoiceField(**kwargs)

Default widget: SelectMultiple

Empty value: An empty QuerySet (self.queryset.none())

Normalizes to: A QuerySet of model instances.

Validates that every id in the given list of values exists in the queryset.

Error message keys: required, list, invalid_choice, invalid_pk_value

The invalid_choice message may contain %(value)s and the invalid_pk_value message may contain %(pk)s, which will be substituted by the appropriate values.

Allows the selection of one or more model objects, suitable for representing a many-to-many relation. As with ModelChoiceField, you can use label_from_instance to customize the object representations.

A single argument is required:

queryset - Same as ModelChoiceField.queryset.

Takes one optional argument:

to_field_name - Same as ModelChoiceField.to_field_name.