
WTForms-Components Documentation

Release 0.1

Konsta Vesterinen, Janne Vanhala, Vesa Uimonen

Oct 30, 2018

Contents

1	Fields	3
1.1	WTForms derived HTML5 Fields	3
1.2	SelectField & SelectMultipleField	4
1.3	PhoneNumberField	4
1.4	ColorField	4
1.5	NumberRangeField	4
1.6	PassiveHiddenField	5
1.7	TimeField	5
1.8	Read-only fields	5
2	Validators	7
2.1	DateRange validator	7
2.2	Email validator	7
2.3	If validator	8
2.4	Chain validator	8
2.5	Unique Validator	8

WTForms-Components provides various additional fields, validators and widgets for WTForms.

1.1 WTForms derived HTML5 Fields

WTForms-Components provides enhanced versions of WTForms HTML5 fields. These fields support HTML5 compatible min and max validators. WTForms-Components is smart enough to automatically attach HTML5 min and max validators based on field's NumberRange and DateRange validators.

Example:

```
from wtforms import Form
from wtforms_components import DateTimeField
from werkzeug.datastructures import MultiDict

class TestForm(Form):
    test_field = DateTimeField(
        'Date',
        validators=[DateRange(
            min=datetime(2000, 1, 1),
            max=datetime(2000, 10, 10)
        )]
    )

form = TestForm(MultiDict(test_field='2000-2-2'))
form.test_field

# <input id="test_field" max="2000-10-10 00:00:00" min="2000-01-01 00:00:00" name=
↪ "test_field" type="datetime" value="2000-2-2">
```

Same applies to IntegerField:

```
from wtforms import Form
from wtforms_components import IntegerField
```

(continues on next page)

(continued from previous page)

```
from werkzeug.datastructures import MultiDict

class TestForm(Form):
    test_field = IntegerField(
        'Date',
        validators=[NumberRange(
            min=1,
            max=4
        )]
    )

form = TestForm(MultiDict(test_field='3'))
form.test_field

# <input id="test_field" max="4" min="1" name="test_field" type="number" value="3">
```

1.2 SelectField & SelectMultipleField

WTForms-Components provides enhanced versions of WTForms SelectFields. Both WTForms-Components SelectField and SelectMultipleField support the following enhancements:

- Ability to generate `optgroup` elements.
- `choices` can be a callable, which allows for dynamic choices. With the plain version of WTForms this has to be added manually, after instantiation of the form.

1.3 PhoneNumberField

Older versions of WTForms-Components had a `PhoneNumberField`. As of version 0.10.0 this field has now been moved to [WTForms-Alchemy](#).

1.4 ColorField

`ColorField` is a string field representing a `Color` object from `colour` package.

Example:

```
from wtforms import Form
from wtforms_components import ColorField

class DocumentForm(Form):
    background_color = ColorField()
```

1.5 NumberRangeField

`NumberRangeField` is a string field representing a `NumberRange` object from [SQLAlchemy-Utils](#).

Example:

```
from wtforms import Form
from wtforms_components import NumberRangeField

class EventForm(Form):
    estimated_participants = NumberRangeField('Estimated participants')
```

1.6 PassiveHiddenField

PassiveHiddenField acts just like normal `wtforms.fields.HiddenField` except it doesn't populate object values with `populate_obj` function.

Example:

```
from wtforms import Form, TextField
from wtforms_components import PassiveHiddenField

class EventForm(Form):
    id = PassiveHiddenField()
    name = TextField('Name')
```

1.7 TimeField

TimeField is a string field which stores a *datetime.time* matching a format.

```
from wtforms import Form, DateField
from wtforms_components import TimeField

class EventForm(Form):
    start_date = DateField('Start date')
    start_time = TimeField('Start time')
```

1.8 Read-only fields

WTForms-Components provides a convenient function for making fields read-only.

In the following example we define a form where name field is defined as read-only.

```
from wtforms import Form, DateField, TextField
from wtforms_components import TimeField, read_only

class EventForm(Form):
    name = TextField('Name')
    start_date = DateField('Start date')
    start_time = TimeField('Start time')

    def __init__(self, *args, **kwargs):
        super(EventForm, self).__init__(*args, **kwargs)
        read_only(self.name)
```


2.1 DateRange validator

The DateRange validator is essentially the same as `wtforms.validators.NumberRange` validator but validates dates.

In the following example we define a `start_time` and a `start_date` field, which do not accept dates in the past.

```
from datetime import datetime, date
from wtforms import Form
from wtforms.fields import DateField
from wtforms_components import DateRange

class EventForm(Form):
    start_time = DateField(
        validators=[DateRange(min=datetime.now())]
    )
    start_date = DateField(
        validators=[DateRange(min=date.today())]
    )
```

2.2 Email validator

Validates an email address. This validator is based on **‘Django’s email validator’** and is stricter than the standard email validator included in WTForms.

Example:

```
from wtforms import Form
from wtforms.fields import TextField
from wtforms_components import Email

class UserForm(Form):
```

(continues on next page)

(continued from previous page)

```
email = TextField(
    validators=[Email()]
)
```

2.3 If validator

The If validator provides means for having conditional validations. In the following example we only validate field email if field user_id is provided.

```
from wtforms import Form
from wtforms.fields import IntegerField, TextField
from wtforms_components import If

class SomeForm(Form):
    user_id = IntegerField()
    email = TextField(validators=[
        If(lambda form, field: form.user_id.data, Email())
    ])
```

2.4 Chain validator

Chain validator chains validators together. Chain validator can be combined with If validator to provide nested conditional validations.

```
from wtforms import Form
from wtforms.fields import IntegerField, TextField
from wtforms_components import If

class SomeForm(Form):
    user_id = IntegerField()
    email = TextField(validators=[
        If(
            lambda form, field: form.user_id.data,
            Chain(DataRequired(), Email())
        )
    ])
```

2.5 Unique Validator

Unique validator provides convenient way for checking the unicity of given field in database.

As of WTForms-Components version 0.10.0 the Unique validator has been moved to WTForms-Alchemy due to its SQLAlchemy dependency.