Django pony forms Documentation

Release 0.3

Marco Braak

Aug 05, 2020

Contents

Django pony forms is an extension to the Django form. It helps you to produce better html for your forms, and to configure the html output.

The source is on github: https://github.com/mbraak/django_pony_forms

CHAPTER 1

What can *Django pony forms* do for you?

Better form html by default Just add *PonyFormMixin* to your form class.

Form templates *Django pony forms* allows you to write form templates. You can write a template for all your forms, or a specific template for one particular form.

Contents:

1.1 Installation

Install the package:

```
$ pip install django_pony_forms
```

Add `django_pony_forms` to your installed apps in `settings.py`.

```
INSTALLED_APPS = (
    ..
    'django_pony_forms',
)
```

1.2 Requirements

The package is tested with Django 2.2 - 3.1 and Python 3.6-3.8.

1.3 Tutorial

1.3.1 Use the mixin

Let's create our first Django pony form. Just mix in the *PonyFormMixin*.

forms.py:

```
from django import forms
from django_pony_forms.pony_forms import PonyFormMixin

class ExampleForm(PonyFormMixin, forms.Form):
    name = forms.CharField()
```

Let's render the form in a template:

index.html:

```
<form method="post">
    {% csrf_token %}
    {{ form }}
    <input type="submit" value="Save" />
</form>
```

The html for the `{{ form }}` looks like this:

```
<div class="form-row required" id="row-name">
    <label for="id_name">Name</label>
    <input id="id_name" type="text" name="name" />
</div>
```

1.3.2 Write a form template

You can define the following form templates:

- Form template; variable *form_template*
- Row template; variable *row_template*
- Errorlist template; variabe *errorlist_template*

Let's change the errorlist template:

```
class ExampleForm(PonyFormMixin, forms.Form):
    errorlist_template = 'my_errorlist.html'

    name = forms.CharField()
```

my_errorlist.html:

```
{% for error in errors %}
    <span class="help-inline">{{ error }}</span>
{% endfor %}
```

1.3.3 Use pony forms in a view template

You can also ignore the form template and define the form in your view template.

This example uses the **rows** property of the pony form:

```
class ExampleForm(PonyFormMixin, forms.Form):
    name = forms.CharField()
    start_date = forms.DateField()
```

```
<form method="post">
    {% csrf_token %}
    {{ form.top_errors }}
    {{ form.hidden_fields }}

    {{ form.rows.name }}
    {{ form.rows.start_date }}
    <input type="submit" value="Save" />
</form>
```

1.3.4 Use fieldsets

This example defines the fieldsets *first* and *second*:

```
class ExampleForm(PonyFormMixin, forms.Form):
    fieldset_definitions = dict(
        first='field1', 'field3'],
        second=['field2', 'field4']
    )

    field1 = forms.CharField()
    field2 = forms.CharField()
    field3 = forms.CharField()
    field4 = forms.CharField()
```

Let's use the fieldsets in the view template:

```
<form method="post">
    {% csrf_token %}
    {{ form.top_errors }}
    {{ form.hidden_fields }}

    <div class="first">
        {{ form.fieldsets.first }}
    </div>

    <div class="second">
        {{ form.fieldsets.second }}
    </div>

    <input type="submit" value="Save" />
</form>
```

1.4 Api

1.4.1 PonyFormMixin

PonyFormMixin has the following properties:

form_template

This sets the form template. The default is 'django_pony_forms/base_form.html'.

```
class ExampleForm(PonyFormMixin, forms.Form):
    form_template = 'my_form.html'
```

row_template

This sets the row template. The default is 'django_pony_forms/row.html'.

```
class ExampleForm(PonyFormMixin, forms.Form):
    row_template = 'my_row.html'
```

errorlist_template

This sets the errorlist template. The default is 'django_pony_forms/errorlist.html'.

```
class ExampleForm(PonyFormMixin, forms.Form):
    errorlist_template = 'my_errorlist.html'
```

fieldset_definitions

This defines the fieldsets that you can use in your view template. It is a dictionary of lists of fieldnames:

```
class ExampleForm(PonyFormMixin, forms.Form):
    fieldset_definitions = dict(
        first=['field_a', 'field_b']
        another=['field_x', 'field_y']
    )
```

The referenced fields must exist in the form.

View template:

```
{{ form.fieldsets.first }}
```

1.4.2 Context for form template

In your form template you can use the following variables.

hidden_fields

Renders the hidden fields. You can also access individual hidden fields.

To render all hidden fields:

```
{{ hidden_fields }}
```

To render a single hidden field:

```
{{ hidden_field.my_hidden_field }}
```

fields

This is a dictionary of all the bound fields. It renders a widget.

```
{{ field.my_name }}
```

rows

Renders all the rows. You can also access individual rows.

To render all rows:

```
{{ rows }}
```

To render a single row. This renders the full row including label and widget.

```
{{ row.my_name }}
```

top_errors

This renders all top errors. You can also iterate over the errors.

```
{{ top_errors }}
```

To iterate over the errors:

```
{% for error in top_errors %}
    <p>{{ errors }}</p>
{% endfor %}
```

fieldsets

You can use this to render a specific fieldset.

```
{{ form.fieldsets.my_first_fieldset }}
```

1.4.3 Context for row template

In a row template you can use the following variables:

label

This renders the label tag:

```
{{ label }}
```

field

This renders the widget:

```
{{ field }}
```

name

This is the name of the field:

```
{{ name }}
```

css_classes

This contains the css classes that are defined by Django. For example, the 'required' class.

```
<div{% if css_classes %} class="{{ css_classes }}"{% endif %}>
</div>
```

help_text

This defines the help text.

```
{{ help_text }}
```

errors

This renders the errors. You can also iterate over the errors.

```
{{ errors }}
```

To iterate over the errors:

```
{% for error in errors %}
    {{ error }}
{% endfor %}
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search