wtforms-webwidgets Documentation

Release 0.1

Nick Whyte

November 05, 2015

Contents

1 Common Library **1**

2 Bootstrap **3**
- 2.1 Core . . . 3
- 2.2 HTML5 . . . 3
- 2.3 Extras . . . 3
- 2.4 Util . . . 3
- 2.5 Abstract Widgets . . . 3

3 Extending **5**
- 3.1 Making Widgets For Your Favourite Framework . . . 5
- 3.2 Contibuting . . . 5

4 Usage In Flask (And Others) **7**
- 4.1 Example . . . 7

CHAPTER 1

Common Library

CHAPTER 2

Bootstrap

default_widgets
A dictionary defining the default widget type for each kind of WTForms Field.

Very useful when using FieldRenderer to automatically render your fields without declaring `widget=MyWidget()`.

2.1 Core

2.2 HTML5

2.3 Extras

2.4 Util

2.5 Abstract Widgets

CHAPTER 3

Extending

3.1 Making Widgets For Your Favourite Framework

Create a submodule within this module named the title of the web framework you wish to bring functionality to.

When creating widgets from scratch, be sure to apply the `wtforms_webwidgets.common.CustomWidgetMixin` mixin to your class.

If you are extending an existing wtforms.widgets class, decorate it with `wtforms_webwidgets.common.custom_widget_wrapper`. This allows our `FieldRenderer` know this is a custom widget, and not to check the lookup dictionary to render a field which has this widget.

3.2 Contibuting

To contribute your improvements to this library, please fork the repository, add functionality and submit a pull request.

CHAPTER 4

Usage In Flask (And Others)

When declaring fields as part of a form using WTForms, if you wished to set a custom widget for a field, you would need to set *widget=MyWidget()*.

I have found when overriding all widgets with some skinned widgets, it's very tedious and prone to errors to set this value every time. Instead, it is more intuitive to set a dictionary of defaults, and look up field types and get their apropriate widget.

To use this method we need:

1. A way of identifying when a widget has been set from the Field kwargs.
2. A way of overriding a default widget when it is not provided.

I have found that it is most aproriatly done within a `render_field` templating macro.

Provided within the common submodule of this framework is the class `FieldRenderer`. This class provides an interface for setting a lookup table for default renderers and a method to render a given field.

4.1 Example

An example from Flask/Jinja.

```
from wtforms_webwidgets import FieldRenderer
from wtforms_webwidgets.bootstrap import default_widgets

renderer = FieldRenderer(lookup_dict=default_widgets)

# Alternatively, you can declare your own lookup dictionary:
import wtforms_webwidgets.bootstrap as wt_bs
renderer = FieldRenderer(lookup_dict={
    'TextField': wt_bs.TextInput(),
})

# Example for injecting into Jinja within Flask
app.jinja_env.globals['render_field'] = renderer
```

Now, within your templates you can do the following:

```
{{ render_field(form.my_field) }}
```

If the widget was not declared with a custom widget, it will be renderered accordingly to the FieldRender's lookup dictionary.

D

default_widgets, 3