django-gollum Documentation

Release 1.0.0

Luke Sneeringer

Sep 27, 2017

Contents

This is **django-gollum**, a better way to mess with styling Django forms. gollum provides a better way to specify just the HTML Attributes and CSS classes you need on a Form subclass.

CHAPTER 1

Installation

To install django-gollum, just use:

```
pip install django-gollum
```

CHAPTER 2

Dependencies

- **Python version**
  - django-gollum is tested on Python 2.7 and Python 3.3.
  - It is probable that it will run on Python 2.6, but Python 2.6 is not explicitly tested.
- **Django version**
  - django-gollum is tested against Django 1.4+ on Python 2.7, and Django 1.5+ on Python 3.3.
- **Other dependencies**
  - dict.sorted
  - six

All dependencies are handled for you if you install using pip.

CHAPTER 3

Getting Started

Adding HTML or CSS to a Django Form with gollum is easy:

1. Subclass *gollum.forms.Form* or *gollum.forms.ModelForm*.
2. Add a Attrs or CSS inner class specifying fields with extra HTML attributes or CSS, respectively.

Here's an example:

```
from gollum import forms

class MyForm(forms.Form):
    foo = models.CharField(max_length=50)
    bar = models.IntegerField()

    class Attrs:
        bar = { 'placeholder': 25 }

    class CSS:
        foo = 'green'
        bar = ['purple', 'translucent']
```

When this form is rendered in the template, the "foo" `<input>` tag will have the "green" CSS class applied, and the "bar" `<input>` will have both "purple" and "translucent" applied. Additionally, the "bar" `<input>` would have `placeholder="25"` set as an HTML attribute.

CHAPTER 4

Getting Help

If you think you've found a bug in django-pgfields itself, please post an issue on the Issue Tracker.

For usage help, you're free to e-mail the author, who will provide help (on a best effort basis) if possible.

CHAPTER 5

License

New BSD

CHAPTER 6

Index

Using django-gollum

In order to use gollum, you must use gollum's `Form` and `ModelForm` superclass, rather than the ones that ship with Django.

This is extremely important: nothing documented here will work unless the gollum Form classes are superclasses of your form. This also gives you the ability to opt-in to gollum forms on an as-needed basis, if you prefer.

The easiest way to do this is just to import your `forms` module from `gollum` rather than `django`:

```
from gollum import forms

class MyGollumForm(forms.Form):
    [...]
```

It's that simple. Read on to learn how to make use of what gollum provides. The next topic is specifying HTML attributes.

Specifying HTML Attributes

In Form Classes

gollum gives you the ability to specify HTML attributes easily on a form class. You can specify HTML attributes on any number of fields on your form by using an `Attrs` inner class within your form:

```
from gollum import forms

class MyGollumForm(forms.Form):
    foo = forms.IntegerField()
    bar = forms.IntegerField()
```

```
    class Attrs:
        foo = { 'disabled': 'true' }
```

The above code will cause the form, when rendered, to add a `disabled="true"` HTML attribute to the `foo` field (but do nothing to the `bar` field).

A common use-case for this, if you're using HTML5 forms, is to set a placeholder attribute. This causes the browser to display default text in the input field until it gains focus:

```
from gollum import forms

class UserForm(forms.Form):
    first_name = forms.CharField(max_length=30)
    last_name = forms.CharField(max_length=30)
    email = forms.EmailField(max_length=75)
    phone = forms.CharField(max_length=15, required=False)

    class Attrs:
        phone = { 'placeholder': 'Optional' }
```

This code would cause the phone widget to look like this:

```
<input name="phone" id="id_phone" type="text" placeholder="optional">
```

In Templates

Even though the Django documentation only exposes a way to set attributes in form widgets themselves, one could argue that a better place for HTML attribute information to live is in the template itself, especially since a form could be used in different templates and need different attributes set.

gollum does not expose a special mechanism to do this (yet). However, this can be accomplished by directly calling the `as_widget` method of a bound field in Django.

The problem: `as_widget` takes arguments, so you'll either need to write a template tag to send the necessary arguments to it, or use a template language that supports arguments (such as Jinja).

Here's a quick sample of the latter option:

```
{% for field in user_form %}
    {{ field.as_widget(attrs={ 'placeholder': field.label }) }}
{% endfor %}
```

This method allows you not to specify HTML attributes on your form class at all, and may be preferable, especially if the HTML attributes change depending on where the form is rendered.

CSS

It would be possible to specify CSS classes in this way, by writing directly to the `class` HTML attribute. But don't; gollum also exposes a way to specify CSS classes.

Specifying CSS Classes

Much like HTML attributes, gollum provides a way to specify CSS classes both in forms and also exposes a new way to do so in templates.

However, unlike HTML attributes, where the most recent attribute specified wins, the CSS class specification understands to take the union of any CSS classes sent to it. That makes it preferable to use this mechanism rather than specifying `class` as an HTML attribute directly.

In Form Classes

In order to specify a CSS class on a form, declare a `CSS` inner class in your `Form` class, and specify any classes to apply:

```
from gollum import forms

class MyForm(forms.Form):
    foo = models.CharField(max_length=50)
    bar = models.IntegerField()

    class CSS:
        foo = {'spam', 'eggs'}
```

The above code, when rendered in an template, will cause the `<input>` tag for the `foo` field to have two CSS classes: `spam` and `eggs`.

You can specify CSS classes here using a list, set, or tuple. You can also use a string if you have only one CSS class, or you can even use a space-separated string like you would in actual HTML markup.

The following CSS inner-class is identical to the one in the example above:

```
class CSS:
    foo = 'spam eggs'
```

Order of class specification doesn't matter; it'll be normalized to a Python `set`, which is unordered. Duplicate classes don't matter either, for the same reason.

In Templates

It may be preferable for your use case to specify CSS classes in templates rather than in the form itself. (This does seem like where such information naturally belongs.)

Again, like in HTML Attributes, the solution is the `as_widget` method. The story's a little different this time, though: Django doesn't provide a clean way to specify CSS classes, so gollum actually subclasses BoundField to provide one.

That mechanism is the `css_classes` keyword argument:

```
{% for field in form %}
    {{ field.as_widget(css_classes='myclass') }}
{% endfor %}
```

Much like the specification above, you can send a list (or other, similar iterable) or a string, and gollum will do the right thing.

Settings

gollum exposes a small number of settings that you can apply to get certain global behaviors on gollum forms.

FORM_REQUIRED_CSS_CLASS

- Default: *(not set)*
- Type: `str`

If specified, the given CSS class will be added to every required field on every gollum form.

FORM_ERROR_CSS_CLASS

- Default: *(not set)*
- Type: `str`

If specified, the given CSS class will be added to every field that is in an error state on every gollum form.

FORM_REQUIRED_ATTRS

- Default: *(not set)*
- Type: `dict`

If specified, the given HTML attributes will be applied to every required field on every gollum form.

Example:

```
FORM_REQUIRED_ATTRS = { 'required': 'true' }
```

FORM_OPTIONAL_ATTRS

- Default: *(not set)*
- Type: `dict`

If specified, the given HTML attributes will be applied to every field that is *not* required on any gollum form.

Example:

```
FORM_OPTIONAL_ATTRS = { 'placeholder': 'Optional' }
```

FORM_GLOBAL_ATTRS

- Default: *(not set)*
- Type: `dict`

If specified, the given HTML attributes will be applied to *every* form field on every gollum form.