django-zenforms Documentation

Release 0.1.0

Ivan Gromov

July 31, 2015

Contents

CHAPTER 1

Dealing with tags

There are two ways of using zenfroms app.

First way - use `{% zenform %}` and `{% endzenform %}` template tags. In the middle of them you can do whatever you want, tags defines a special conetxt and track form fields usage.

Second way is the use `{% izenform %}` tempalte tag. This tag renders all form, and don't allow you to make modifications in field flow.

1.1 {% zenform %} and {% endzenform %}

Usage

```
{% zenform form [options key1=value1, key2=value2] %}
    Your form goes here!
    {% fieldset unused_fields title 'All my form' %}
{% endzenform %}
```

- `form` - Django's form
- `key1=value1, key2=value2` - options string, which will be converted to a dict. you can add your options and use them later in templates

To use this tag pair, you should use `{% fieldset %}` tags. Follow documentation about this tag usage.

Context

Tag creates special inner conetxt.

- `form` - original form, passed in arguments
- `unused_fields` - fields, that were not rendered within the tag.
- `options` - options dict specified in tag definition

Tag can watch what fields were used only when you are rendering them with 'zenforms' tags. I.e. it couldn't track used and unused fields if you place them manually.

Example

Render Django's default `UserCreationForm`:

```
{% zenform form %}
    {% fieldset 'username' title 'User data' %}
    {% fieldset unused_fields title 'The rest' %}
{% endzenform %}
```

Templates

Tags `{% zenform %}` and `{% endzenform %}` uses two templates to wrap the rendered form:

- `zenforms/zenform_prefix.html`
- `zenforms/zenform_postfix.html`

You are welcome to ovreride them in your project.

Note: Zenforms will add a bit of css classes to your widgets. I hope, it wil not crash your app. It adds `textInput` css class for `forms.CharField` and `forms.EmailField`, and `error` class for bound fields with errors.

1.2 {% fieldset %}

Second part of `{% zenform %}`-`{% endzenform %}` tags is `{% fieldset %}` tag.

Usage:

```
{% fieldset 'field1' 'field2' [title 'MyFieldset'] %}
{% fieldset unused_fields %}
```

- `'field1' 'field2'` - strings with field names, which will be included in fieldset
- `MyFieldset` - optionally you can set fieldsset's title. Therefore, it will be rendered as <h3> tag.

Warning: Aware of use iterable arguments in the `{% fieldset %}` tag. Because of `unused_fields` argument support, this tag consider all iterables as `unused_fields`.

Template

This tag uses `'zenforms/fieldset.html'` template. There is nothing interesting there, only includes and cycles. You wouldn't like to override it.

But in `zenforms/fields` you may find something interesting. Common fields are rendered with `zenforms/fields/single.html` template.

1.3 {% multifield %}

If you want to group several fields in one line of form this tag is for you.

Usage:

```
{% multifield args as varname [label 'Label'] %}
```

- args - List of form field names, which you want to group. Quotes are nessecary,
- varname - output variable name. Quotes are not nessecary,
- label - optional group's name.

Multifield tag returns object recognizable by `{% fieldset %}` so you do the following:

```
{% multifield 'first_name' 'last_name' as credentials label 'Enter your name' %}
{% fieldset credentials 'password1' 'password2' %}
```

Template

`{% multifield %}` tag renders it's contents via `zenforms/fields/multi.html` template. You may override it.

1.4 {% readonly %}

I faced with the task to display data and then edit it. I created this tag as a solution. `{% readonly %}` tag renders django's model instance data like it would be in form.

Usage:

```
{% readonly instance 'field1' 'field2' [label 'MyLabel'] [as varname] %}
```

- `instance` - Django model instance
- `'field1' 'field2'` - list of instance fields. It is important that model must have that fields
- `MyLabel` - you can optionally specify a label for all fields, for example, User data
- `varname` - optionally saves rendered fields into template variable for futher usage.

ReadonlyTag also returns recognizable by `{% fieldset %}` value, you can mix fields, multifields and read-olny-fields as you wish.

```
{% zenform form %}
{% readonly admin 'username' 'last_name' label 'Your admin data' as admin_data %}
{% multifield 'first_name' 'last_name' as credentials label 'Enter your name' %}
{% fieldset admin_data credentials unused_fields %}
{% endzenform %}
```

Template

`{% readonly %}` tag renders it's contents via `zenforms/fields/readonly.html` template. You may override it too.

1.5 {% submit %}

Very simple tag. Renders submit button in button holder for you.

Usage:

```
{% submit [value] %}
```

- `value` - submit value, for example, 'Save' or 'Send'

Template

Tag uses `zenforms/submit.html` tempalte. Override it if you wish.

1.6 {% izenform %}

Finally! The last tag `{% izenform %}` renders for without bunch of template tags, if simply renders all form fields into one fieldset. In the most cases it is tag-what-you-need.

Usage:

```
{% izenform form [options key1=value1, key2=value2] %}
```

Options are the same as for {% zenform %} tags:

- `form` - Django's form
- `key1=value1, key2=value2` - options string, which will be converted to a dict. you can add your options and use them later in templates

Template

Tag uses `zenforms/zenform_inline.html` template. Nothing interesting there.

1.7 {{ form.field|attrs:"class=required"}}

If you want to quickly modify input attribute, you can use `attrs` tempalte filter.

Usage:

```
{{ form.field|attrs:"attr1=value1,attr2=value2,attr3=value3 value4" }}
```

Pretty simple to write more.

CHAPTER 2

Zenform options

There are several pre-defined options.

- notag - if `options.notag` returns True, form will be rendered without tag
- nocsrf - zeonforms automatically insert `{% scrf_token %}` in all forms. If you're using GET method, you'll definetly need this option.
- action - form action
- method - form method
- inline - if `options.inline` returns True, form will be rendered in alternate layout, the label is on the left side of field, rather than on the top
- submit - submit text, used in `zenforms/submit.html` template, as submit control value.

CHAPTER 3

Changelog

- 0.1.0 - Initial verion
- 0.1.1 - Templates and tags adapted to real world, fixed javascript bug.
- 0.1.2 - Cumulative fix for some annoying things
- 0.1.6 - Template-only fixes
- 0.1.7 - Added attrs filter

CHAPTER 4

Django zenforms

I definitely going mad about forms in Django. They keep all layout buisness in python code.

Dragan Babić make a simple yet wonderful form css+jquery framework, called uniforms. I liked it a lot. In this app I tried to save all original Dragan's work and adapt to it.

Since I met great app, django-crispy-forms (ex. django-uni-forms), I promised to myself to do something like that but templates. And I did.

CHAPTER 5

Quickstart:

Django way:

```
<form action="/contact/" method="post">
    {{ form.non_field_errors }}
    {% for field in form %}
        <div class="fieldWrapper">
            {{ field.errors }}
            {{ field.label }} {{ field }}
            {% if field.help_text %}
            <p class="formHint">{{ field.help_text }}</p>
            {% emdif %}
        </div>
    {% endfor %}
    {% csrf_token %}
</form>
```

Zenforms way:

```
{% izenform form %}
```

or:

```
<form action="." method="post">
{% izenform form1 options no_form_tag=1 %}
{% izenform form2 options no_form_tag=1 %}
<input type="submit" />
</form>
```

Regroup fields? as you wish:

```
{% zenform form %}
    {% fieldset 'username' title 'User data' %}
    {% multifield 'phone1' 'phone2' as phones label 'Phones' %}
    {% fieldset 'first_name' 'last_name' phones %}
    {% fieldset unused_fields title 'Rest stuff' %}  <!-- you can foget something, zenforms can take
{% endzenform %}
```

Moreover, you can include read-only values from Django models:

```
{% readonly admin 'username' 'last_name' label 'Information' %}
```

See full documentation.

CHAPTER 6

Installation

Istall package:

```
pip install django-zenforms
```

Install 'zenforms' in INSTALLED_APPS.

And put static files on page:

```
<link rel="stylesheet" href="{{ STATIC_URL }}zenforms/css/uni-form.css" type="text/css" />
<link rel="stylesheet" href="{{ STATIC_URL }}zenforms/css/default.uni-form.css" type="text/css" />

<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/1.7/jquery.min.js"></scrip
<script type="text/javascript" src="{{ STATIC_URL }}zenforms/js/uni-form.jquery.min.js"></script>
```