**Project description**

Tweak the form field rendering in templates, not in python-level form definitions. Altering CSS classes and HTML attributes is supported.

That should be enough for designers to customize field presentation (using CSS and unobtrusive javascript) without touching python code.

License is MIT.

**Installation**

You can get Django Widget Tweaks by using pip:

$ pip install django-widget-tweaks

To enable *widget\_tweaks* in your project you need to add it to *INSTALLED\_APPS* in your projects *settings.py* file:

INSTALLED\_APPS = [

...

'widget\_tweaks',

...

]

**Usage**

This app provides two sets of tools that may be used together or standalone:

1. a render\_field template tag for customizing form fields by using an HTML-like syntax.
2. several template filters for customizing form field HTML attributes and CSS classes

The render\_field tag should be easier to use and should make form field customizations much easier for designers and front-end developers.

The template filters are more powerful than the render\_field tag, but they use a more complex and less HTML-like syntax.

**render\_field**

This is a template tag that can be used as an alternative to aforementioned filters. This template tag renders a field using a syntax similar to plain HTML attributes.

Example:

{% load widget\_tweaks %}

<!-- change input type (e.g. to HTML5) -->

{% render\_field form.search\_query type="search" %}

<!-- add/change several attributes -->

{% render\_field form.text rows="20" cols="20" title="Hello, world!" %}

<!-- append to an attribute -->

{% render\_field form.title class+="css\_class\_1 css\_class\_2" %}

<!-- template variables can be used as attribute values -->

{% render\_field form.text placeholder=form.text.label %}

<!-- double colon -->

{% render\_field form.search\_query v-bind::class="{active:isActive}" %}

For fields rendered with {% render\_field %} tag it is possible to set error class and required fields class by using WIDGET\_ERROR\_CLASS and WIDGET\_REQUIRED\_CLASS template variables:

{% with WIDGET\_ERROR\_CLASS='my\_error' WIDGET\_REQUIRED\_CLASS='my\_required' %}

{% render\_field form.field1 %}

{% render\_field form.field2 %}

{% render\_field form.field3 %}

{% endwith %}

You can be creative with these variables: e.g. a context processor could set a default CSS error class on all fields rendered by {% render\_field %}.

**attr**

Adds or replaces any single html attribute for the form field.

Examples:

{% load widget\_tweaks %}

<!-- change input type (e.g. to HTML5) -->

{{ form.search\_query|attr:"type:search" }}

<!-- add/change several attributes -->

{{ form.text|attr:"rows:20"|attr:"cols:20"|attr:"title:Hello, world!" }}

<!-- attributes without parameters -->

{{ form.search\_query|attr:"autofocus" }}

<!-- attributes with double colon Vuejs output: v-bind:class="{active:ValueEnabled}" -->

{{ form.search\_query|attr:"v-bind::class:{active:ValueEnabled}" }}

**add\_class**

Adds CSS class to field element. Split classes by whitespace in order to add several classes at once.

Example:

{% load widget\_tweaks %}

<!-- add 2 extra css classes to field element -->

{{ form.title|add\_class:"css\_class\_1 css\_class\_2" }}

**set\_data**

Sets HTML5 data attribute ( <http://ejohn.org/blog/html-5-data-attributes/> ). Useful for unobtrusive javascript. It is just a shortcut for ‘attr’ filter that prepends attribute names with ‘data-‘ string.

Example:

{% load widget\_tweaks %}

<!-- data-filters:"OverText" will be added to input field -->

{{ form.title|set\_data:"filters:OverText" }}

**append\_attr**

Appends attribute value with extra data.

Example:

{% load widget\_tweaks %}

<!-- add 2 extra css classes to field element -->

{{ form.title|append\_attr:"class:css\_class\_1 css\_class\_2" }}

‘add\_class’ filter is just a shortcut for ‘append\_attr’ filter that adds values to the ‘class’ attribute.

**remove\_attr**

Removes any single html attribute for the form field.

Example:

{% load widget\_tweaks %}

<!-- removes autofocus attribute from field element -->

{{ form.title|remove\_attr:"autofocus" }}

**add\_label\_class**

The same as *add\_class* but adds css class to form labels.

Example:

{% load widget\_tweaks %}

<!-- add 2 extra css classes to field label element -->

{{ form.title|add\_label\_class:"label\_class\_1 label\_class\_2" }}

**add\_error\_class**

The same as ‘add\_class’ but adds css class only if validation failed for the field (field.errors is not empty).

Example:

{% load widget\_tweaks %}

<!-- add 'error-border' css class on field error -->

{{ form.title|add\_error\_class:"error-border" }}

**add\_error\_attr**

The same as ‘attr’ but sets an attribute only if validation failed for the field (field.errors is not empty). This can be useful when dealing with accessibility:

{% load widget\_tweaks %}

<!-- add aria-invalid="true" attribute on field error -->

{{ form.title|add\_error\_attr:"aria-invalid:true" }}

**add\_required\_class**

The same as ‘add\_error\_class’ adds css class only for required field.

Example:

{% load widget\_tweaks %}

<!-- add 'is-required' css class on field required -->

{{ form.title|add\_required\_class:"is-required" }}

**field\_type and widget\_type**

'field\_type' and 'widget\_type' are template filters that return field class name and field widget class name (in lower case).

Example:

{% load widget\_tweaks %}

<div class="field {{ field|field\_type }} {{ field|widget\_type }} {{ field.html\_name }}">

{{ field }}

</div>

Output:

<div class="field charfield textinput name">

<input id="id\_name" type="text" name="name" maxlength="100" />

</div>

**Mixing render\_field and filters**

The render\_field tag and filters mentioned above can be mixed.

Example:

{% render\_field form.category|append\_attr:"readonly:readonly" type="text" placeholder="Category" %}

returns:

<input name="category" placeholder="Profession" readonly="readonly" type="text">

**Filter chaining**

The order django-widget-tweaks filters apply may seem counter-intuitive (leftmost filter wins):

{{ form.simple|attr:"foo:bar"|attr:"foo:baz" }}

returns:

<input foo="bar" type="text" name="simple" id="id\_simple" />

It is not a bug, it is a feature that enables creating reusable templates with overridable defaults.

Reusable field template example:

{# inc/field.html #}

{% load widget\_tweaks %}

<div>{{ field|attr:"foo:default\_foo" }}</div>

Example usage:

{# my\_template.html #}

{% load widget\_tweaks %}

<form method='POST' action=''> {% csrf\_token %}

{% include "inc/field.html" with field=form.title %}

{% include "inc/field.html" with field=form.description|attr:"foo:non\_default\_foo" %}

</form>

With ‘rightmost filter wins’ rule it wouldn’t be possible to override |attr:"foo:default\_foo" in main template.

**Contributing**

If you’ve found a bug, implemented a feature or have a suggestion, do not hesitate to contact me, fire an issue or send a pull request.

* Source code: <https://github.com/jazzband/django-widget-tweaks/>
* Bug tracker: <https://github.com/jazzband/django-widget-tweaks/issues>

**Testing**

Make sure you have [tox](http://tox.testrun.org/) installed, then type

tox

from the source checkout.

**NOT SUPPORTED**

MultiWidgets: SplitDateTimeWidget, SplitHiddenDateTimeWidget

**Changes**

**1.4.8 (2020-03-12)**

* Fix Release version

**1.4.7 (2020-03-10)**

* Fix Travis deployment to Jazzband site

**1.4.6 (2020-03-09)**

* Feature remove attribute from field
* Added documentation for remove\_attr feature
* Reformat code with black for PEP8 compatibility
* More consistent tox configuration
* adding a new templatetag, unittest and documentation
* Deprecate Python 2.7 support
* Use automatic formatting for all files

**1.4.5 (2019-06-08)**

* Fix rST formatting errors.

**1.4.4 (2019-06-05)**

* Add support for type attr.
* Add Python 3.7 and drop Python 3.3 support.
* Add support for double colon syntax.

**1.4.3 (2018-09-6)**

* Added add\_label\_class filter for CSS on form labels
* Removed compatibility code for unsupported Django versions
* Fixed support for non-value attributes in Django < 1.8
* Support non-value attributes in HTML5 by setting their value to True

**1.4.2 (2018-03-19)**

* update readme to make installation more clear
* shallow copy field before updating attributes
* drop Python 2.6 and Python 3.2 support
* always cast the result of render to a string
* fix import for django >= 2.0
* moved to jazzband

**1.4.1 (2015-06-29)**

* fixed a regression in django-widget-tweaks v1.4 (the field is no longer deep copied).

**1.4 (2015-06-27)**

* Django 1.7, 1.8 and 1.9 support;
* setup.py is switched to setuptools;
* testing improvements;
* Python 3.4 support is added;
* Python 2.5 is not longer supported;
* bitbucket repository is no longer supported (development is moved to github).

**1.3 (2013-04-05)**

* added support for WIDGET\_ERROR\_CLASS and WIDGET\_REQUIRED\_CLASS template variables that affect {% render\_field %}.

**1.2 (2013-03-23)**

* new add\_error\_attr template filter;
* testing improvements.

**1.1.2 (2012-06-06)**

* support for template variables is added to render\_field tag;
* new field\_type and widget\_type filters.

**1.1.1 (2012-03-22)**

* some issues with render\_field tag are fixed.

**1.1 (2012-03-22)**

* render\_field template tag.

**1.0 (2012-02-06)**

* filters return empty strings instead of raising exceptions if field is missing;
* test running improvements;
* python 3 support;
* undocumented ‘behave’ filter is removed.

**0.3 (2011-03-04)**

* add\_error\_class filter.

**0.2.1 (2011-02-03)**

* Attributes customized in widgets are preserved;
* no more extra whitespaces;
* tests;

**0.1 (2011-01-12)**

Initial release.