django-fineforms Documentation

Release 0.6.1

Feinheit AG

Feb 17, 2021

Contents

1 Goals **3**

2 Non-goals **5**

3 Installation **7**

4 High-level overview **9**

5 Template tags **11**
5.1 `{% ff_field field [type=field] %}` 11
5.2 `{% ff_fields form [fields='a,b,c' | exclude='a,b,c'] %}` 11
5.3 `{% ff_errors form1 [form2 ...] %}` 12
5.4 `{% ff_hidden_fields form1 [form2 ...] %}` 12

6 Change log **13**
6.1 Next version 13
6.2 0.6 (2021-02-16) 13
6.3 0.5 (2019-02-13) 13
6.4 0.4 (2018-12-26) 13
6.5 0.3 (2017-09-04) 14
6.6 0.2 (2017-05-25) 14
6.7 0.1 (2017-05-16) 14

This library offers an improved replacement for Django's own form rendering methods (`as_p`, `as_table` etc.) while staying simple and extensible but without introducing a whole new framework.

django-fineforms consists of a template tag library and a few opinionated default templates.

CHAPTER 1

Goals

- Stay simple and extensible
- Avoid options, settings and customizability as much as possible

CHAPTER 2

Non-goals

- Compete with django-crispy-forms or any of the more flexible libraries out there

CHAPTER 3

Installation

Simply `pip install django-fineforms`, and add `fineforms` to your `INSTALLED_APPS`.

CHAPTER 4

High-level overview

The template tags mostly wrap their arguments in wrapper classes that do the real work. For example, `{% ff_field %}` simply wraps the passed field in a wrapper defined in the `FINEFORMS_WRAPPERS` setting. All wrappers use a template to render their output. The default wrapper types are as follows:

```
{
    "errors": ErrorsWrapper,
    "field": FieldWrapper,
    "field-plain": PlainFieldWrapper,
    "fields": FieldsWrapper,
}
```

The wrappers themselves mostly aren't configurable, but you can replace individual wrappers (or all of them) by adding a `FINEFORMS_WRAPPERS` setting. You do not have to override all of them; if you only want to add another wrapper for a specific field type you could just set:

```
FINEFORMS_WRAPPERS = {
    "specific": "app.wrappers.SpecificWrapper",
}
```

... and use this wrapper as `{% ff_field some_field type='specific' %}` somewhere in your templates.

CHAPTER 5

Template tags

All template tags are contained in the `fineforms` library.

5.1 `{% ff_field field [type=field] %}`

Template: `fineforms/field.html`

Render a single field. The wrapper can be optionally overridden by passing a different type. The key has to exist in the `FINEFORMS_WRAPPERS` dictionary.

The default implementation renders the label, the widget, help text and errors related to the field. It is recommended to also set the `error_css_class` and `required_css_class` form attributes; those classes are also added to the output.

The `field-plain` type can be used if the widget should be rendered alone. A wrapping `<span>` tag still contains the CSS classes mentioned above.

5.2 `{% ff_fields form [fields='a,b,c' | exclude='a,b,c'] %}`

Template: `fineforms/fields.html`

Render fields of a form. `fields` and `exclude` are comma-separated strings that can be used to only render a selection of fields. The `fields` parameter takes precedence if both are given.

Hidden fields are rendered separately at the end, all other fields are wrapped using `FINEFORMS_WRAPPERS["field"]` and rendered as well.

5.3 {% ff_errors form1 [form2 ...] %}

Template: `fineforms/errors.html`

Render form errors at the top. The default implementation renders all non-field errors, and all errors from hidden fields. Falsy parameters (i.e. `None`) are filtered out for you. If there aren't any errors at all nothing is rendered.

5.4 {% ff_hidden_fields form1 [form2 ...] %}

This template tag is the outlier in that it does not use a template at all. The return value is the concatenated result of rendering all hidden fields of all passed forms. Falsy parameters (i.e. `None`) are filtered out for you.

Please note that `{% ff_fields %}` adds hidden fields to the output automatically.

CHAPTER 6

Change log

6.1 Next version

6.2 0.6 (2021-02-16)

- Added `forms` to the context of `fineforms/errors.html`, `form` to the context of `fineforms/fields.html`.
- Switched from Travis CI to GitHub actions.

6.3 0.5 (2019-02-13)

- Added a `ff_submit` tag for showing submit buttons.
- All defaults are arbitrary. Arbitrarily changed the default for the `foundation_xy_grid` package to only show fields and labels in one line for large screend.

6.4 0.4 (2018-12-26)

- Added CSS classes to the default `ff_errors` template's top level element and also to the error list.
- Moved all wrappers into `fineforms.wrappers`.
- Reformatted the code using black.
- Modified template tags to pass on keyword arguments to the wrappers they instantiate.
- Added a template package using Foundation's XY-grid, usable by adding `fineforms.pacakges.foundation_xy_grid` before `fineforms` to `INSTALLED_APPS`.

6.5 0.3 (2017-09-04)

- Django also adds `required_css_class` to the label tag; do the same (and also add `error` because it simplifies the code).
- Parametrize the CSS classes used for `error_css_class` and `required_css_class`.
- Add an additional wrapping `div` to the default `widget_then_label` field template to avoid flex children layout.
- Add the widget class name as a `widget--$name` CSS class to the output. This makes styling radio selects more straightforward (use a `.widget--radioselect ...` CSS selector)

6.6 0.2 (2017-05-25)

- Documentation work.
- Allow specifying additional wrappers as dotted Python paths to avoid problems with circular imports.
- The `FieldWrapper` now passes `label_tag` and `css_classes` into the template; `label_tag` uses `FieldWrapper.label_suffix` (defaulting to `''`), and `css_classes` contains `required` and `error` if fitting and if `Form.required_css_class` and `Form.error_css_class` are undefined.

6.7 0.1 (2017-05-16)

- Initial public version.