form-designer Documentation

Release 0.21.2

Feinheit AG

Jan 19, 2023

Contents

This form designer does not try to offer every last configuration possibility of Django's forms, just through the administration interface instead of directly in Python code. Instead, it strives to be a tool which everyone can use right away, without the need for long explanations.

It offers a small set of predefined input fields such as:

- Text fields (One line and multi line widgets possible)
- E-mail address fields
- Checkboxes
- Dropdowns
- Radio Buttons
- Multiple selection checkboxes
- Hidden input fields

Every field can optionally be declared mandatory, default values and help texts are available too. That's it.

The default actions (which can be enabled individually) are to send the form data to a list of freely definable email addresses and to store the data in the database so that it can be exported later. An XLSX export of saved submissions is provided too. It is possible to add your own actions as well.

CHAPTER 1

Installing the form designer

Install the package using pip:

```
$ pip install form-designer
```

CHAPTER 2

Setting up the form designer

- Add "form_designer" and "admin_ordering" to INSTALLED_APPS.
- Run ./manage.py migrate form_designer
- Go into Django's admin panel and add one or more forms with the fields you require. Also select at least one action in the configuration options selectbox, most often you'd want to select both the "E-mail" and the "Save form submission" option and fill in one ore more email addresses.

If you're using the form designer with FeinCMS, the content type can be imported from form_designer.contents.FormContent. Otherwise, your code should use the following methods (the code would probably reside in a view):

```
# Somehow fetch the form_designer.models.Form instance:
instance = get_object_or_404()

# Build the form class:
form_class = instance.form_class()

# Standard form processing:
if request.method == "POST":
    form = form_class(request.POST)

    if form.is_valid():
        # Do what you want, or run the configured processors:
        result = instance.process(form, request)

        # Maybe there's something useful in here:
        pprint(result)

        return HttpResponseRedirect("thanks/")
else:
    form = form_class()

return render(request, "form.html", {"form": form})
```

CHAPTER 3

Adding custom actions

Custom actions can be added by appending them to `Form.CONFIG_OPTIONS`:

```python
from form_designer.models import Form

def do_thing(model_instance, form_instance, request, config, **kwargs):
    pass

def do_validate(form_instance, data):
    pass

Form.CONFIG_OPTIONS.append(
    (
        "do_thing",
        {
            "title": _("Do a thing"),
            "form_fields": lambda form: [
                ("optional_form_field", forms.CharField(
                    label=_("Optional form field"),
                    required=False,
                    # validators...
                    # help_text...
                )),
            ],
            "process": do_thing,
            "validate": do_validate,
        },
    )
)
```

The interesting part is the `do_thing` callable. It currently receives four arguments, however you should also accept `**kwargs` to support additional arguments added in the future:

- `model_instance`: The `Form` model instance
- `form_instance`: The dynamically generated form instance

- `request`: The current HTTP request
- `config`: The config options (keys and values defined through `form_fields`; for example the `email` action defines an `email` char field, and accesses its value using `config["email"]`.

CHAPTER 4

ReCaptcha

To enable [ReCaptcha](http://www.google.com/recaptcha) install [django-recaptcha](https://github.com/praekelt/django-recaptcha) and add *captcha* to your *INSTALLED_APPS*. This will automatically add a ReCaptcha field to the form designer. For everything else read through the django-recaptcha readme.

CHAPTER 5

Override field types

Define `FORM_DESIGNER_FIELD_TYPES` in your settings file like:

```
FORM_DESIGNER_FIELD_TYPES = "your_project.form_designer_config.FIELD_TYPES"
```

In `your_project.form_designer_config.py` something like:

```
from django import forms
from django.utils.translation import gettext_lazy as _

FIELD_TYPES = [
    {"type": "text", "verbose_name": _("text"), "field": forms.CharField},
    {"type": "email", "verbose_name": _("email address"), "field": forms.EmailField},
]
```

CHAPTER 6

Visit these sites for more information

- form_designer: https://github.com/feincms/form_designer
- FeinCMS: http://www.feinheit.ch/labs/feincms-django-cms/
- feincms3: https://feincms3.readthedocs.io/

CHAPTER 7

Change log

7.1 Next version

- Changed the default ordering of forms in the admin change list to alphabetical.

7.2 0.21

- Started using real JSON fields for the configuration and submitted data instead of stringifying the JSON explicitly. A side effect of this change may be that the JSON object key order isn't preserved but it was a bad idea to rely on this anyway. **Please check carefully if the database migration preserves past form submissions and form configurations. It worked when I tested it but you probably want to check twice.**
- Added a date field type to the list to the default fields.
- Rewrote the submissions export to export all available data (also from removed fields) in order.
- Replaced `FormSubmission.sorted_data` etc. with a centralized facility on the form model which properly handles missing fields, titles, old names etc. and which doesn't produce stupidly expensive 1+N queries when processing a list of submissions.

7.3 0.20

- Added Django 4.1 to the CI matrix.
- Extended the `email` action with the optional ability to add the author of the sibmission to the Cc: of the sent email.
- Changed mails to prefer the titles of fields instead of their name.
- Included field titles in the XLSX export when possible.
- Removed our usage of `collections.OrderedDict`, not necessary anymore.

- Changed submissions to save the full URL, not just the path. Cleaned up the form submissions model while at it.

7.4 0.19

- Specified the default `AutoField` for the form designer to avoid migrations.
- Added a reCAPTCHA v3 field to the default field types.
- Added Django 4.0a1 to the CI matrix.
- Disallowed non-sluggy contents on the name field. The old field name is preserved for exporting form submissions etc. so the change *should* be backwards compatible but don't rely on it too much.
- Raised the minimum requirements to Python 3.8, Django 3.2.

7.5 0.18

- Stopped hardcoding the admin base form class. Overriding the `form` attribute on the `ModelAdmin` class for forms now works as expected.
- Raised the minimum requirements to Python 3.6, Django 2.2.
- Switched to a declarative setup.
- Switched to GitHub actions.

7.6 0.17

- Fixed a typo and changed e-mail to email. Removed the help text from config options which isn't correct anymore.
- Added code to avoid crashing the admin interface if form submissions cannot be deserialized and/or rendered.
- Worked around changes in the initialization of change forms in the administration interface.
- Added Django 3.1 and Python 3.8 to the Travis CI matrix.
- Reordered the configuration options in the administration panel; moved activation of processors closer to their configuration.
- Fixed a recurring bug where migrations would be created when changing field types.
- Made the ordering field a bit wider so that the value is still visible even on small screens.

7.7 0.16

- Fixed the config fieldsets code to work when using Django 3.0.
- Added an unit test and docs for the `"validate"` config option.
- Passed additional data to the `"validate"` config option. Not accepting arbitrary keyword arguments is now deprecated.

- Changed the forms administration interface to show all form options from the beginning. This requires a change to the `form_fields` configuration option: Instead of a list it has to be a callable accepting the form instance now.
- Added support for specifying a description for config options.

7.8 0.15

- Restructured the `FIELD_TYPES` data structure to use a dictionary instead of a tuple to allow for future expansion.
- Dropped compatibility with Python 2.

7.9 0.14

- Fixed the package to include static files and templates.
- Raised the minimum django-recaptcha version to 2.0.

7.10 0.13

- Added tox configuration for easily running linters and tests locally.
- Reformatted the project using black
- Made django-admin-ordering a dependency.
- Replaced the CSV export with an XLSX export based on xlsxdocument. It just is a better format.
- Improved the test coverage a bit and fixed an edge case where form field model validation would crash.

7.11 0.12

- Changed `FormSubmission.sorted_data` (and by extension also `formatted_data(_html)` and the CSV export) to use field names instead of field titles as keys. Field names are guaranteed to be unique, titles are not.

7.12 0.11

- Moved form processing into `FormContent.process`; this removes the need to pass the request to `FormContent.render`. `render` is not expected to require a request parameter in FeinCMS content types.
- Added Django 1.11 to the test matrix. No changes were necessary for 1.11 support.
- Added documentation for adding new actions.
- Fixed a bug where activated config options were lost because of differences between `list()` and `dict_keys()` objects.

7.13 0.10

- Make the fields tabular inline a bit less wide.
- Added czech translations.
- Fixed the usage of `render_to_string` to actually work correctly with Django 1.10.

7.14 0.9

- The form admin uses django-admin-ordering for fields if available.
- Now supports sending notification mails to multiple addresses.

7.15 0.8

- Moved the `FormContent` to the new module `form_designer.contents` to make the form designer usable without FeinCMS.
- Replaced `SortedDict` with `collections.OrderedDict`.
- Fixed an XSS vulnerability in the administration.
- Dropped compatibility with old Django versions (<1.8).
- Replaced the horrible form submission serialization of `repr()` and `eval()` with JSON.
- General packaging and code cleanups.

7.16 0.7

- Avoid the deprecated `mimetype` argument to HTTP responses.
- Fixed infinite recursion in `jsonize`.
- Made field type choices lazy so that changing available field types is easier resp. actually possible.

7.17 0.6

- Improve code coverage, less warnings, less complaining.

7.18 0.5

- Added an app config for a nicer app name.

7.19 0.4

- Built-in support for Django 1.7-style migrations. If you're using South, update to South 1.0 or better.

7.20 0.3

- Support for Python 3.3, 2.7 and 2.6.
- Support for overridding field types with `FORM_DESIGNER_FIELD_TYPES`.