django-addanother Documentation

Release 0.1.2

Jonas Haag

September 06, 2016

Contents

django-addanother provides you with add-another and edit-related buttons for forms outside the Django administration interface. It also provides an optional integration with django-select2.

Supported Django versions: 1.8, 1.9, 1.10 (others may work well though)

CHAPTER 1

Getting started

1.1 How to Install

1. `pip install django-addanother`.
2. Add `'django_addanother'` to your `INSTALLED_APPS`.
3. Make sure static `'django.contrib.staticfiles'` and `'django.contrib.admin'` are part of your `INSTALLED_APPS`.

1.2 Demo

To run the demo, clone the repository, run `test_project/manage.py migrate` and `test_project/manage.py runserver` and then go to `http://localhost:8000`.

1.3 How to Use

1.3.1 1. Add the add-another button

Wrap *`django_addanother.widgets.AddAnotherWidgetWrapper`* around your widget to show the add-another button next to it.

For example, let's say we want to add add-another buttons to a model form:

```
from django.core.urlresolvers import reverse_lazy
from django_addanother.widgets import AddAnotherWidgetWrapper

class FooForm(forms.ModelForm):
    class Meta:
        ...
        widgets = {
            'sender': AddAnotherWidgetWrapper(
                forms.Select,
                reverse_lazy('person_create'),
            ),
            'recipients': AddAnotherWidgetWrapper(
                forms.SelectMultiple,
                reverse_lazy('person_create'),
```

```
            )
        }
```

This will add an add-another button next to the `sender` and `recipients` fields. When clicked, these will open the `'person_create'` URL in a popup.

Important: Be sure to include form media and jQuery in your templates:

```
{{ form }}
<script src="{% static 'admin/js/vendor/jquery/jquery.js' %}"></script>
{{ form.media }}
```

1.3.2 2. Make your view popup-compatible

Note: This assumes you're using Django's generic `CreateView`. django-addanother doesn't support function-based views at the point of writing. You'll have to convert any function-based views to Class Based Views first.

Making your `CreateView` compatible with django-addanother is as simple as making it inherit the *`django_addanother.views.CreatePopupMixin`* class:

```
from django_addanother.views import AddPopupMixin

class PersonCreate(CreatePopupMixin, CreateView):
    model = Foo
    ...
```

This overwrites your view's `form_valid()` method to return a special JavaScript response in case a form has been submitted from a popup.

You may want to hide header, footer and navigation elements for the popups. When the create view is opened in a popup, the `view.is_popup` template variable is set:

```
    {% if not view.is_popup %}
      <nav>...</nav>
    {% endif %}
```

1.3.3 3. Profit

That's it!

See *Edit-related buttons* on how to add edit buttons too.

1.4 Edit-related buttons

Similarly to add-another buttons (see *How to Use*), to add edit-related buttons to your widget, proceed with the following steps:

1. Wrap your widget with the *`AddAnotherEditSelectedWidgetWrapper`* class, and provide an edit URL in addition to the add URL.
2. Make your edit view popup-compatible by having it inherit the *`CreatePopupMixin`* class.

The edit URL must contain the `__fk__` string as a placeholder for the actual object's primary key. Example:

```
# forms.py
from django.core.urlresolvers import reverse_lazy
from django_addanother.widgets import AddAnotherEditSelectedWidgetWrapper

class FooForm(forms.ModelForm):
    class Meta:
        ...
        widgets = {
            'sender': AddAnotherEditSelectedWidgetWrapper(
                forms.Select,
                reverse_lazy('person_create'),
                reverse_lazy('person_update', args=['__fk__']),
            )
        }


# views.py
from django_addanother.views import UpdatePopupMixin

class PersonUpdate(UpdatePopupMixin, UpdateView):
    model = Foo
    ...
```

If you need the edit-related button only, but not the add-another, wrap your widget with the *EditSelectedWidgetWrapper* class and remove the add URL.

CHAPTER 2

Advanced topics

2.1 Select2 Integration

django-addanother provides optional lightweight integration with django-select2.

Usage example:

```
from django_addanother.contrib.select2 import Select2AddAnother

class FooForm(forms.ModelForm):
    class Meta:
        ...
        widgets = {
            'sender': Select2AddAnother(reverse_lazy('person_create')),
        }
```

See *Select2 Widgets Reference* for a list of provided widgets.

2.2 How it Works

Note: django-addanother works exactly like the add-another and edit-related features in Django's admin.

django-addanother works twofold: Firstly, it adds an add-another and an edit-related button next to your form fields. When one of these buttons is clicked, a special popup window with the "inline" creation form is opened.

The popup window isn't much different from your usual form handling views. The main difference is that when the form has been submitted and validated successfully, after saving the newly created object, the user is not redirected to the view's `success_url`. Instead, a special JavaScript-only response is being sent to the browser, adding the new object to the selection (or modifying the option) in the original window and closing the popup window.

Any `CreateView` or `UpdateView` can be made compatible with django-addanother. When opened in a popup, the view gets appended the `?_popup=1` GET parameter, which is how the view knows when to respond with its special JavaScript response. This special handling is taken care of in *`django_addanother.views.CreatePopupMixin`* and *`django_addanother.views.UpdatePopupMixin`*, whose usage is explained in *How to Use*.

2.3 Reference

2.3.1 Views Reference

class django_addanother.views.**CreatePopupMixin**
Mixin for CreateView classes that handles the case of the view being opened in an add-another popup window.

Changed in version 2.0.0: This used to be called PopupMixin and has been renamed with the introduction of edit-related buttons and *UpdatePopupMixin*.

class django_addanother.views.**UpdatePopupMixin**
Mixin for UpdateView classes that handles the case of the view being opened in an edit-related popup window.

New in version 2.0.0.

2.3.2 Widgets Reference

class django_addanother.widgets.**AddAnotherWidgetWrapper**(*widget*, *add_related_url*, *add_icon=None*)
Widget wrapper that adds an add-another button next to the original widget.

class django_addanother.widgets.**EditSelectedWidgetWrapper**(*widget*, *edit_related_url*, *edit_icon=None*)
Widget wrapper that adds an edit-related button next to the original widget.

class django_addanother.widgets.**AddAnotherEditSelectedWidgetWrapper**(*widget*, *add_related_url*, *edit_related_url*, *add_icon=None*, *edit_icon=None*)
Widget wrapper that adds both add-another and edit-related button next to the original widget.

2.3.3 Select2 Widgets Reference

django-select2 Widget	...AddAnother	...EditSelected	...AddAnotherEditSelected
Select2Widget	Select2AddAnother	Select2EditSelected	Select2AddAnotherEditSelected
HeavySelect2Widget	HeavySelect2AddAnother	HeavySelect2EditSelected	HeavySelect2AddAnotherEditSelected
Select2MultipleWidget	Select2MultipleAddAnother	Select2MultipleEditSelected	Select2MultipleAddAnotherEditSelected
HeavySelect2MultipleWidget	HeavySelect2MultipleAddAnother	HeavySelect2MultipleEditSelected	HeavySelect2MultipleAddAnotherEditSelected
HeavySelect2TagWidget	HeavySelect2TagAddAnother	HeavySelect2TagEditSelected	HeavySelect2TagAddAnotherEditSelected
ModelSelect2Widget	ModelSelect2AddAnother	ModelSelect2EditSelected	ModelSelect2AddAnotherEditSelected
ModelSelect2MultipleWidget	ModelSelect2MultipleAddAnother	ModelSelect2MultipleEditSelected	ModelSelect2MultipleAddAnotherEditSelected
ModelSelect2TagWidget	ModelSelect2TagAddAnother	ModelSelect2TagEditSelected	ModelSelect2TagAddAnotherEditSelected

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

A

C

E

U