## Contents

1 Getting started
   1.1 How to Install .................................................. 3
   1.2 Demo ................................................................. 3
   1.3 How to Use .......................................................... 3
   1.4 Edit-related buttons ............................................. 5

2 Advanced topics
   2.1 Select2 Integration ................................................ 7
   2.2 How it Works ......................................................... 7
   2.3 Reference ............................................................ 8

3 Indices and tables


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

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.

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.

How to Use

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:

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

class FooForm(forms.ModelForm):
    class Meta:
        ...  # other Meta properties
    widgets = {
        # add your widgets here
doctest.addclip before
```
'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 }}
%
```

## 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:

```python
from django_addanother.views import CreatePopupMixin
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 %}
%
```

## 3. Profit

That’s it!

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

```python
# 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.
Select2 Integration

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

Usage example:

```python
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.

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*.

## Reference

### Views Reference

### Widgets Reference

### Select2 Widgets Reference

<table>
<thead>
<tr>
<th>django-select2 Widget</th>
<th>...AddAnother</th>
<th>...EditSelected</th>
<th>...AddAnotherEditSelected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select2Widget</td>
<td>Select2AddAnother</td>
<td>Select2EditSelected</td>
<td>Select2AddAnotherEditSelected</td>
</tr>
<tr>
<td>HeavySelect2Widget</td>
<td>HeavySelect2AddAnother</td>
<td>HeavySelect2EditSelected</td>
<td>HeavySelect2AddAnotherEditSelected</td>
</tr>
<tr>
<td>Select2Multiple</td>
<td>Select2MultipleAddAnother</td>
<td>Select2MultipleEditSelected</td>
<td>Select2MultipleAddAnotherEditSelected</td>
</tr>
<tr>
<td>HeavySelect2Multiple</td>
<td>HeavySelect2MultipleAddAnother</td>
<td>HeavySelect2MultipleEditSelected</td>
<td>HeavySelect2MultipleAddAnotherEditSelected</td>
</tr>
<tr>
<td>HeavySelect2Tag</td>
<td>HeavySelect2TagAddAnother</td>
<td>HeavySelect2TagEditSelected</td>
<td>HeavySelect2TagAddAnotherEditSelected</td>
</tr>
<tr>
<td>ModelSelect2Widget</td>
<td>ModelSelect2AddAnother</td>
<td>ModelSelect2EditSelected</td>
<td>ModelSelect2AddAnotherEditSelected</td>
</tr>
<tr>
<td>ModelSelect2Multiple</td>
<td>ModelSelect2MultipleAddAnother</td>
<td>ModelSelect2MultipleEditSelected</td>
<td>ModelSelect2MultipleAddAnotherEditSelected</td>
</tr>
<tr>
<td>ModelSelect2Tag</td>
<td>ModelSelect2TagAddAnother</td>
<td>ModelSelect2TagEditSelected</td>
<td>ModelSelect2TagAddAnotherEditSelected</td>
</tr>
</tbody>
</table>
Indices and tables

- genindex
- modindex
- search