

---

# **Django-Admin-Object-Actions Documentation**

*Release 0.1.5*

**Nine More Minutes, Inc.**

**Aug 11, 2018**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Usage</b>	<b>5</b>



**Django-Admin-Object-Actions** provides object-level actions in the Django admin interface.

In contrast to the built-in admin actions, which are only applied to a list of objects from the change list, object actions apply to a single object and can be accessed from either the change list view or the change form view.

Each action is defined declaratively on the `ModelAdmin` instance and may provide a form for additional input or validation before executing the action.

Inspired by [This Article](#).

**It is tested against:**

- Django 1.11 (Python 2.7, 3.4, 3.5 and 3.6)
- Django 2.0 (Python 3.4, 3.5, 3.6 and 3.7)
- Django 2.1 (Python 3.5, 3.6 and 3.7)
- Django master/2.2 (Python 3.5, 3.6 and 3.7)



# CHAPTER 1

---

## Installation

---

Install the application from PYPI:

```
pip install django-admin-object-actions
```

Add `admin_object_actions` to your `INSTALLED_APPS`:

```
INSTALLED_APPS += ('admin_object_actions',)
```

Django-Admin-Object-Actions requires `django-crum` and will install it as a dependency. To enable it, add `CurrentRequestUserMiddleware` to your `MIDDLEWARE` setting:

```
MIDDLEWARE += ('crum.CurrentRequestUserMiddleware',)
```





## CHAPTER 2

---

### Usage

---

Ensure your `ModelAdmin` class inherits from the `ModelAdminObjectActionsMixin`:

```
# myapp/admin.py
from django.contrib import admin
from admin_model_actions.admin import ModelAdminObjectActionsMixin
from .models import MyModel

@admin.register(MyModel)
class MyModelAdmin(ModelAdminObjectActionsMixin, admin.ModelAdmin):
    pass
```

Create custom subclasses of `AdminObjectActionForm` for actions requiring additional input or confirmation and implement the `do_object_action` method:

```
# myapp/forms.py
from django import forms
from admin_model_actions.forms import AdminObjectActionForm
from .models import MyModel

class MyActionForm(AdminObjectActionForm):

    confirm = forms.BooleanField()

    class Meta:
        model = MyModel
        fields = ()

    def do_object_action(self):
        self.instance.action()
```

Define `object_actions` on your `ModelAdmin` class similar to the following:

```
from .forms import MyActionForm
```

(continues on next page)

(continued from previous page)

```
class MyModelAdmin(ModelAdminObjectActionsMixin, admin.ModelAdmin):

    object_actions = [
        {
            'slug': 'myaction',
            'verbose_name': _('my action'),
            'verbose_name_past': _('acted upon'),
            'form_class': MyActionForm,
            'fields': ('id', 'confirm'),
            'readonly_fields': ('id',),
            'permission': 'change',
        },
        {
            'slug': 'myotheraction',
            'verbose_name': _('my other action'),
            'verbose_name_past': _('acted upon'),
            'form_method': 'GET',
            'function': 'do_other_action',
            'permission': 'otheraction',
        },
    ]

    def has_otheraction_permission(self, request, obj=None):
        return True

    def do_other_action(self, obj, form):
        obj.other_action(form.cleaned_data)
```

Each object action listed on the `ModelAdmin` must define a `slug` field and may define additional fields to customize the action's behavior:

- slug** The internal name of this action; will be used to create the custom URL used by the action.
- verbose\_name** The translatable name of this action displayed on the action buttons in the admin interface. Defaults to `slug.title()`.
- verbose\_name\_title** The translatable name shown on the object action form page. Defaults to `verbose_name`.
- verbose\_name\_past** The translatable past tense version of the action displayed to users in messages and admin log entries.
- form\_class** The form class used to implement validation/confirmation of this action, should be a subclass of `admin_object_actions.forms.AdminObjectActionForm`.
- function** Function called to execute the given object action. The default is to call `do_object_action` on the given `form_class`. This option may be a string, in which case the method with the same name from the `ModelAdmin` class or the `Model` class will be used.
- readonly\_field** List of readonly fields to display in the custom admin form.
- fields** List of fields to display in the custom action form.
- fieldsets** Custom fieldsets to display for the object action form. Defaults to a single fieldset with fields.
- permission** Custom permission required to display or execute this object action. Default is `change`. If defined, a `has_{<permission>}_permission` method on the `ModelAdmin` class will be called to check whether the action is allowed.

**form\_template** Custom form template used to render the object action form. Default is `admin/object_action_form.html`.

**list\_only** If `True`, this object action will only be shown in the changelist view.

**detail\_only** If `True`, this object action will only be shown in the change form view.

**view** Overrides the default view function called for this action. The default is the `object_action_view` method defined on the mixin class.

Additional methods of the `ModelAdminObjectActionsMixin` class may be overridden to further customize the behavior of object actions.