django-relatives Documentation

Release 1.4.0a1

Trey Hunner

Jan 13, 2024

CONTENTS

When using Django's admin site, I often find myself wishing I could easily visit model relations from my change lists and change forms. The django-relatives package is meant to do exactly this.

CHAPTER

ONE

USES

The main usages of this Django app are:

- *Linking to foreign keys in change lists*
- *Linking to foreign keys in change forms*
- *Linking to reverse relations on change forms*
- *Linking to foreign keys in inlines*

CHAPTER

TWO

CONTENTS

2.1 Usage

2.1.1 Installation

Install from PyPI:

```
$ pip install django-relatives
```

2.1.2 Linking to foreign keys in change lists

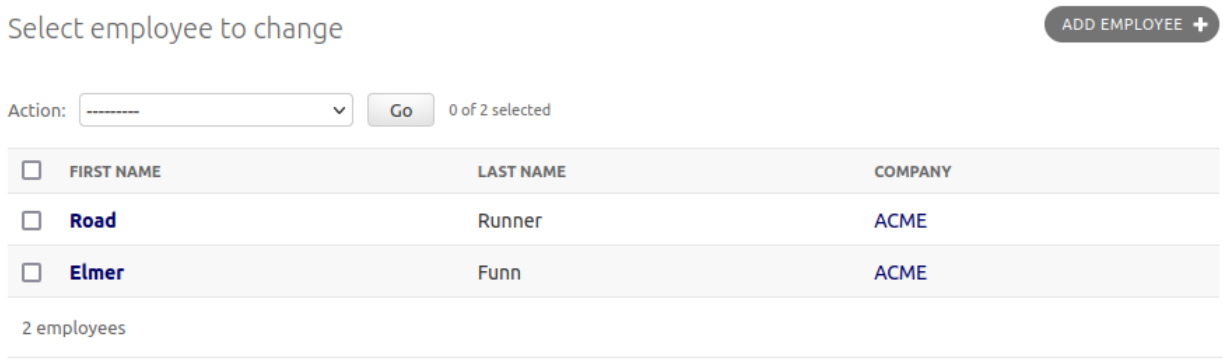
The easiest way to automatically to turn foreign key columns into hyperlinks in a Django admin change list is to inherit from *RelativesAdmin* in your model admin.

Example

```
from django.contrib import admin
from relatives import RelativesAdmin

from .models import Company, Employee


@admin.register(Company)
class CompanyAdmin(admin.ModelAdmin):
    list_display = ["name"]


@admin.register(Employee)
class EmployeeAdmin(RelativesAdmin):
    list_display = ["first_name", "last_name", "company"]
```

Screenshot

When viewing the change list for the employee model, you'll now see that the company column contents are all linked to the appropriate change form page for each company.



2.1.3 Linking to foreign keys in change forms

The *contents_or_fk_link* template filter can be used to link to foreign keys for readonly admin form fields.

Django Relatives also provides a replacement for the `admin/includes/fieldset.html` template which can be used to automatically link to all readonly foreign key fields in change forms.

To use the custom fieldset template you must add `relatives` to `INSTALLED_APPS` in your settings file:

```
INSTALLED_APPS = (
    ...
    'relatives',
)
```

Next create a `admin/includes/fieldset.html` template file:

```
{% extends "relatives/includes/fieldset.html" %}
```

Also make sure this template file is in a custom template directory or an app listed before your admin app in `INSTALLED_APPS`.

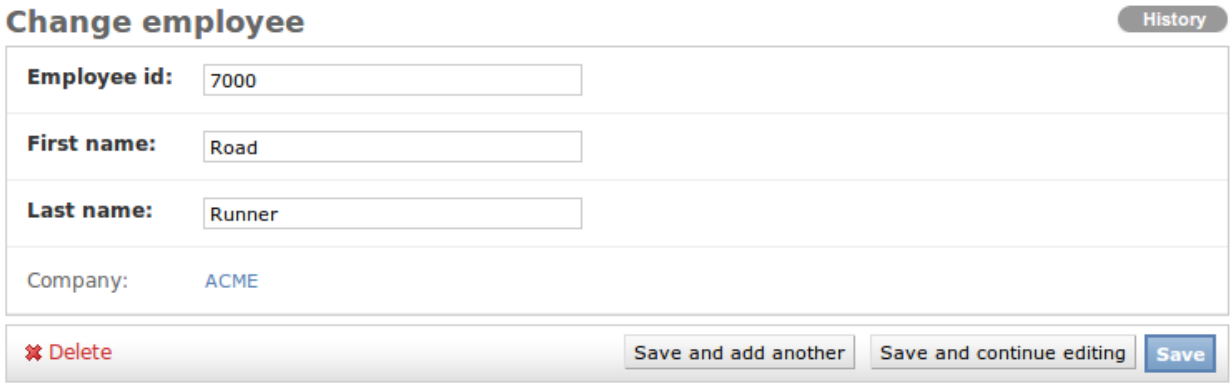
Example Screenshot

2.1.4 Linking to reverse relations on change forms

The *related_objects* template tag makes it easy to link to change lists filtered for reverse relations (objects that have a foreign key to a given object).

Django Relatives also provides a custom `change_form.html` template that may be used to add a "Relations" sidebar to change forms. This sidebar provides links to change list queries for all objects that contain a foreign key to the current object.

To use the custom fieldset template you must add `relatives` to `INSTALLED_APPS` in your settings file:




```
INSTALLED_APPS = (
    ...
    'relatives',
)
```

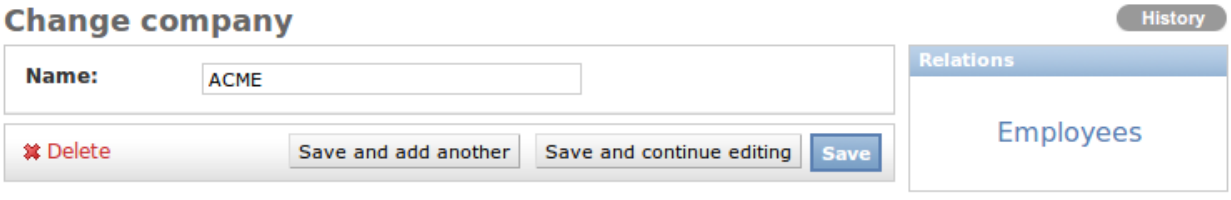
Now you can customize the change form template for your desired models/apps. The easiest way to link to reverse relations is to override the `change_form_template` in your `ModelAdmin` subclass.

Example

Code

```
from django.contrib import admin

from .models import Company, Employee


admin.site.register(Employee)


@admin.register(Company, CompanyAdmin)
class CompanyAdmin(admin.ModelAdmin):
    change_form_template = "relatives/change_form.html"
```

Screenshot



Linking to reverse relations with custom template

If you don't have access to change the `ModelAdmin` for your model or you are already customizing your model's admin change form, you will need to use a custom admin template instead.

Create a `admin/YOURAPP/YOURMODEL/change_form.html` template file that extends from `relatives/change_form.html`:

```
{% extends "relatives/change_form.html" %}
```

Also make sure this template file is in a custom template directory or an app listed before your admin app in `INSTALLED_APPS`.

2.1.5 Edit links in inlines

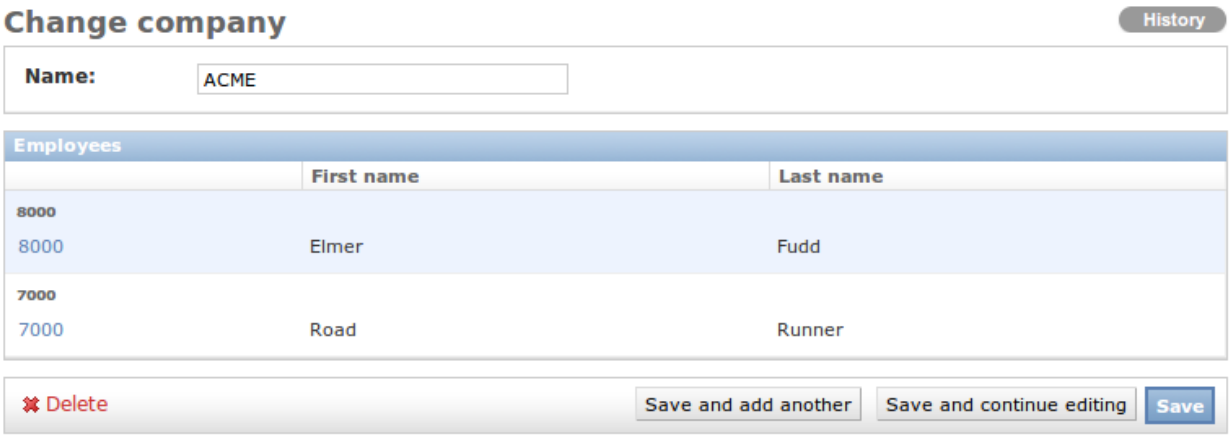
To link to an inline object, include the *object_link* utility function in your admin inline's `fields` list and `readonly_fields` list.

Example

Code

```
from django.contrib import admin
from relatives.utils import object_link

from .models import Company, Employee


admin.site.register(Employee)


class EmployeeInline(admin.TabularInline):
    model = Employee
    fields = [object_link, "first_name", "last_name"]
    readonly_fields = fields
    extra = 0
    max_num = 0
    can_delete = False


@admin.register(Company)
class CompanyAdmin(admin.ModelAdmin):
    inlines = [EmployeeInline]
```

Screenshot



2.1.6 Customizing inline edit links

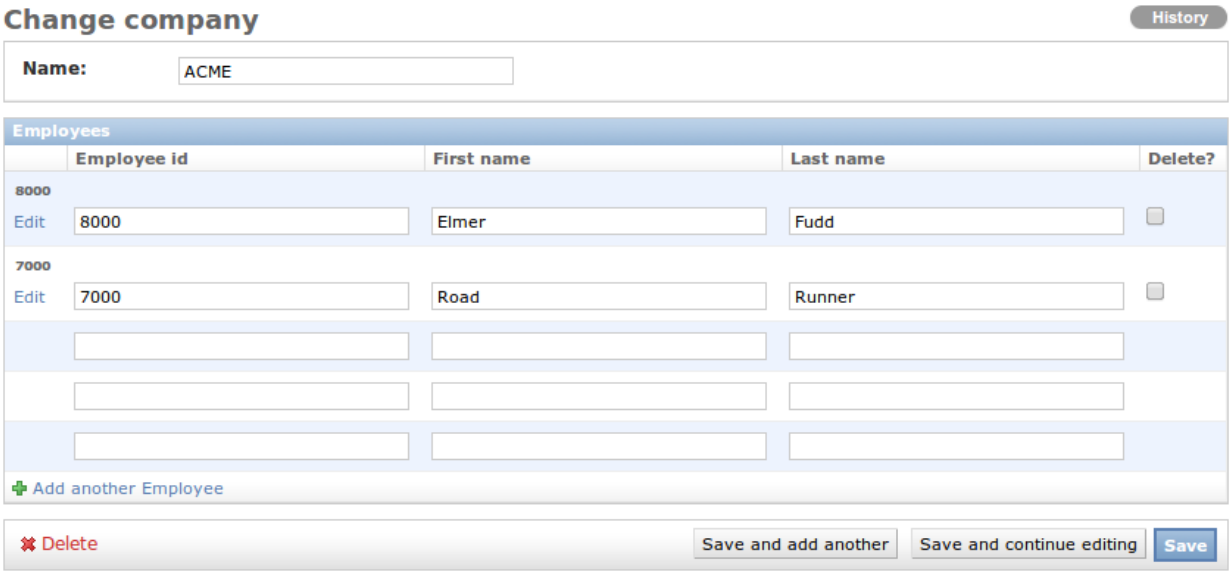
To customize the link text for your inline links, use the *object_edit_link* utility function instead, specifying the edit text and blank text (both are optional).

Example

Code

```
from django.contrib import admin
from relatives.utils import object_edit_link

from .models import Company, Employee


admin.site.register(Employee)


class EmployeeInline(admin.TabularInline):
    model = Employee
    edit_link = object_edit_link("Edit")
    fields = [edit_link, "employee_id", "first_name", "last_name"]
    readonly_fields = [edit_link]


@admin.register(Company)
class CompanyAdmin(admin.ModelAdmin):
    inlines = [EmployeeInline]
```

Screenshot



2.1.7 Customizing settings

Since relatives uses cache, you can update the settings module to change the defaults if you'd like:

```
RELATIVES_CACHE_KEY = 'relatives_cache'
RELATIVES_CACHE_TIME = int(60*60*24)
```

2.2 API Reference

2.2.1 RelativesAdmin

class relatives.**RelativesAdmin**(*model*, *admin_site*)

`ModelAdmin` that links to related fields.

This automatically applies a variation of the `object_link` utility to all `list_display` fields.

class relatives.**RelativesMixin**

`ModelAdmin` mixin that makes list display foreign keys linked.

This is used by `RelativesAdmin`. This can be used along with Django's default `ModelAdmin` instead of inheriting from `RelativesAdmin`.

2.2.2 Utility Functions

relatives.utils.**get_admin_url**(*obj*)

Return admin URL for given object (raise NoReverseMatch on error)

relatives.utils.**object_edit_link**(*edit_text=None*, *blank_text=None*)

Return function that takes an object and returns admin link to object

Arguments:

- `edit_text` is displayed in link text
- `blank_text` is displayed in unlinked text (when no admin link)

`edit_text` defaults to the object's unicode representation and `blank_text` defaults to the object's unicode representation if `edit_text` is None and an empty string otherwise

relatives.utils.**object_link**(*obj*)

Return admin link to given object or blank text if no link

Equivalent to object_edit_link()(obj)

2.2.3 Template Tags

relatives.templatetags.relatives.**contents_or_fk_link**(*field*)

Return field contents or link to related object if foreign key field

Example Usage:

```
{% load relatives %}
{{ field|contents_or_fk_link }}
```

relatives.templatetags.relatives.**related_objects**(*obj*)

Return list of objects related to the given model instance

Example Usage:

```
{% load relatives %}
{% related_objects obj as related_objects %}
{% for related_obj in related_objects %}
    <a href="{{ related_obj.url }}">{{ related_obj.plural_name }}</a>
{% endfor %}
```

CHAPTER

THREE

CONTRIBUTING

Visit the project on Github to view the source, submit issues and pull requests.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

r

INDEX