How to override and extend basic Django admin templates?

Asked 13 years, 5 months ago Modified 8 months ago

Viewed 250k times

176

How do I override an admin template (e.g. admin/index.html) while at the same time extending it (see https://docs.djangoproject.com/en/dev/ref/contrib/admin/#overriding-vs-replacing-an-admin-template)?

First - I know that this question has been asked and answered before (see Django: Overriding AND extending an app template) but as the answer says it isn't directly applicable if you're using the app_directories template loader (which is most of the time).

My current workaround is to make copies and extend from them instead of extending directly from the admin templates. This works great but it's really confusing and adds extra work when the admin templates change.

It could think of some custom extend-tag for the templates but I don't want to reinvent the wheel if there already exists a solution.

On a side note: Does anybody know if this problem will be addressed by Django itself?

python django django-templates django-admin extend

Share

Improve this question

Follow

edited May 12, 2023 at 8:21

Super Kai - Kazuya Ito

1

asked Jul 5, 2011 at 13:59

Semmel

2,666 ● 3 ● 23 ● 32

2 Copying the admin templates, extending them and overriding/adding blocks is the most efficient, although not optimal workflow given the current state of Django. I haven't seen any other way to do what you're trying to do in three years of working with it :) – Brandon Taylor Jul 5, 2011 at 14:17

1 Well - I don't know if this is a good thing or not but at least people like you have come to the same conclusion. That's good to hear. :) – Semmel Jul 5, 2011 at 16:02

12 Answers

Sorted by: Highest score (default)

134

Update:

Read the Docs for your version of Django, e.g. the latest version or old LTS versions: 3.2, 2.2, 1.11

Original answer from 2011:

I had the same issue about a year and a half ago and I found a nice template loader on djangosnippets.org that makes this easy. It allows you to extend a template in a specific app, giving you the ability to create your own **admin/index.html** that extends the admin/index.html template from the admin app. Like this:

```
{% extends "admin:admin/index.html" %}

{% block sidebar %}
    {{block.super}}
    <div>
        <h1>Extra links</h1>
        <a href="/admin/extra/">My extra link</a>
    </div>
{% endblock %}
```

I've given a full example on how to use this template loader in a blog post on my website.

Share Improve this answer

Follow

edited Jul 27, 2022 at 8:42

djvg

14.2k ● 7 ● 82 ● 112

answered Jul 5, 2011 at 16:48

heyman

4,944 ● 3 ● 28 ● 19

22 For reference; the snippet in question has been converted to a django app, and is available in PyPi (pip/easy_install) as django-apptemplates: pypi.python.org/pypi/django-apptemplates – Romløk Oct 9, 2012 at 9:19

12 Just to be 100% explicit: the above solution WILL NO LONGER WORK for recent versions of Django (at least 1.4),

as one of the functions the script uses is depreciated. You can find the updated source on here – OldTinfoil Mar 28, 2013 at 17:11

3 Note that with Django 1.8 this will still work, but the setup needs to be made in a special way (see app_namespace.Loader setup as an example). django-app-namespace-template-loader is also a working alternative to `django-apptemplates` if it may stop working one day. – Peterino Sep 11, 2015 at 23:53

This answer was very good for the older Django versions. But as of now, Another answer by Cheng is more relevant. stackoverflow.com/a/29997719/7344164 – DevLoverUmar Jul 15, 2020 at 5:12

87

As for Django 1.8 being the current release, there is no need to symlink, copy the admin/templates to your project folder, or install middlewares as suggested by the answers above. Here is what to do:

1. create the following tree structure(recommended by the official documentation)

```
your_project
     |-- your_project/
     |-- myapp/
     |-- templates/
          |-- admin/
               |-- myapp/
                    |-- change_form.html  <- do not
```

Note: The location of this file is not important. You can put it inside your app and it will still work. As long as its location can be discovered by django. What's more

important is the name of the HTML file has to be the same as the original HTML file name provided by django.

2. Add this template path to your **settings.py**:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.djang
        'DIRS': [os.path.join(BASE_DIR, 'templates
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processor
                'django.template.context_processor
                'django.contrib.auth.context_proce
                'django.contrib.messages.context_p
            ],
        },
    },
]
```

3. Identify the name and block you want to override. This is done by looking into django's admin/templates directory. I am using virtualenv, so for me, the path is here:

```
~/.virtualenvs/edge/lib/python2.7/site-
packages/django/contrib/admin/templates/admin
```

In this example, I want to modify the add new user form. The template responsiblve for this view is **change_form.html**. Open up the change_form.html and find the {% block %} that you want to extend.

4. In **your change_form.html**, write somethings like this:

```
{% extends "admin/change_form.html" %}
{% block field_sets %}
     {# your modification here #}
{% endblock %}
```

5. Load up your page and you should see the changes

Share Improve this answer Follow

edited Sep 2, 2015 at 7:47

answered May 2, 2015 at 2:29

Cheng
17.8k ● 23 ● 75 ● 105

It is still not enough for extending the main "index.html" template, without copying all blocks. A solution is to write some `../` to "exetends" path and to specify the original path more unique `{% extends "../../admin/templates/admin/index.html" %}` . link to answer – hynekcer Jun 15, 2015 at 12:31

1 I think in TEMPLATES we should be using 'DIRS': [os.path.join(BASE_DIR, 'templates')], – Raul Reyes Jul 4, 2015 at 0:36

This is the type of thread that perfectly illustrates the flaw in SO. A framework gets updated and the question is no longer relevant, it is in fact a deterrent from the proper path. Great answer here. RTFM kids. – Derek Adair Mar 13, 2016 at 14:41

Thanks for this answer. Except for "The location of this file is not important.", everything worked great.
– Jaswanth Manigundan Nov 21, 2017 at 22:37

69

if you need to overwrite the `admin/index.html`, you can set the index_template parameter of the `AdminSite`.

e.g.

```
# urls.py
...
from django.contrib import admin

admin.site.index_template = 'admin/my_custom_index.htm
admin.autodiscover()
```

and place your template in

`<appname>/templates/admin/my_custom_index.html`

Share Improve this answer

Follow

answered Dec 26, 2013 at 19:21

gingerlime

5,326 ● 4 ● 41 ● 69

6 Brilliant! Doing this allows you to then do `{% extends "admin/index.html" %}` from my_custom_index.html and have that reference the django admin template without copying it. Thank you. – mattmc3 May 13, 2014 at 15:34

4 @Semmel should mark this as the correct answer, since it's the simplest approach that uses built-in django features and doesn't require using custom template loaders. – MrColes Jun 10, 2015 at 17:25

24

With `django` 1.5 (at least) you can define the template you want to use for a particular `modeladmin`

see
https://docs.djangoproject.com/en/1.5/ref/contrib/admin/#custom-template-options

You can do something like

```
class Myadmin(admin.ModelAdmin):
    change_form_template = 'change_form.htm'
```

With `change_form.html` being a simple html template extending `admin/change_form.html` (or not if you want to do it from scratch)

Share Improve this answer Follow

edited Nov 25, 2016 at 9:51

answered Jun 21, 2013 at 9:57

maazza
7,173 ● 16 ● 65 ● 98

12

I couldn't find a single answer or a section in the official Django docs that had *all* the information I needed to override/extend the default admin templates, so I'm writing this answer as a complete guide, hoping that it would be helpful for others in the future.

Assuming the standard Django project structure:

```
mysite-container/          # project container director
    manage.py
    mysite/                # project package
```

```
        __init__.py
        admin.py
        apps.py
        settings.py
        urls.py
        wsgi.py
    app1/
    app2/
    ...
    static/
    templates/
```

Here's what you need to do:

1. In `mysite/admin.py`, create a sub-class of `AdminSite`:

    ```
    from django.contrib.admin import AdminSite


    class CustomAdminSite(AdminSite):
        # set values for `site_header`, `site_title`,
        site_header = 'Custom Admin Site'
        ...

        # extend / override admin views, such as `inde
        def index(self, request, extra_context=None):
            extra_context = extra_context or {}

            # do whatever you want to do and save the
            extra_context['world'] = 'Earth'

            return super(CustomAdminSite, self).index(


    custom_admin_site = CustomAdminSite()
    ```

    Make sure to import `custom_admin_site` in the `admin.py` of your apps and register your models on it

to display them on your customized admin site (if you want to).

2. In `mysite/apps.py`, create a sub-class of `AdminConfig` and set `default_site` to `admin.CustomAdminSite` from the previous step:

```
from django.contrib.admin.apps import AdminConfig


class CustomAdminConfig(AdminConfig):
    default_site = 'admin.CustomAdminSite'
```

3. In `mysite/settings.py`, replace `django.admin.site` in `INSTALLED_APPS` with `apps.CustomAdminConfig` (your custom admin app config from the previous step).

4. In `mysite/urls.py`, replace `admin.site.urls` from the admin URL to `custom_admin_site.urls`

```
from .admin import custom_admin_site


urlpatterns = [
    ...
    path('admin/', custom_admin_site.urls),
    # for Django 1.x versions: url(r'^admin/',
include(custom_admin_site.urls)),
    ...
]
```

5. Create the template you want to modify in your `templates` directory, maintaining the default Django admin templates directory structure as specified in the docs. For example, if you were modifying

`admin/index.html`, create the file `templates/admin/index.html`.

All of the existing templates can be modified this way, and their names and structures can be found in Django's source code.

6. Now you can either override the template by writing it from scratch or extend it and then override/extend specific blocks.

   For example, if you wanted to keep everything as-is but wanted to override the `content` block (which on the index page lists the apps and their models that you registered), add the following to `templates/admin/index.html`:

```
{% extends 'admin/index.html' %}

{% block content %}
  <h1>
    Hello, {{ world }}!
  </h1>
{% endblock %}
```

   To preserve the original contents of a block, add `{{ block.super }}` wherever you want the original contents to be displayed:

```
{% extends 'admin/index.html' %}

{% block content %}
  <h1>
    Hello, {{ world }}!
  </h1>
  {{ block.super }}
{% endblock %}
```

You can also add custom styles and scripts by modifying the `extrastyle` and `extrahead` blocks.

Share Improve this answer Follow

edited Aug 10, 2019 at 17:19

answered Mar 2, 2019 at 8:54

Faheel

2,535 ● 25 ● 36

do you have a source or documentation about this?
– user12951147 May 23, 2020 at 7:37

Apart from the two references I've added in point 5, no, I don't have anything else. – Faheel Jun 4, 2020 at 17:03

10

Chengs's answer is correct, howewer according to the admin docs not every admin template can be overwritten this way:
https://docs.djangoproject.com/en/1.9/ref/contrib/admin/#overriding-admin-templates

> *Templates which may be overridden per app or model*
>
> Not every template in contrib/admin/templates/admin may be overridden per app or per model. The following can:

> ```
> app_index.html
> change_form.html
> change_list.html
> delete_confirmation.html
> object_history.html
> ```
>
> For those **templates that cannot be overridden** in this way, you may still override them for your entire project. Just **place** the new version in your **templates/admin** directory. This is particularly useful to create custom 404 and 500 pages

I had to overwrite the login.html of the admin and therefore had to put the overwritten template in this folder structure:

```
your_project
 |-- your_project/
 |-- myapp/
 |-- templates/
      |-- admin/
          |-- login.html  <- do not misspell this
```

(without the myapp subfolder in the admin) I do not have enough reputation for commenting on Cheng's post this is why I had to write this as new answer.

Share Improve this answer Follow

edited Aug 20, 2016 at 9:31

answered Aug 18, 2016 at 21:41

matyas
2,796 ● 25 ● 29

Thank you for the feedback hyneker I hope my answer is clearer and more straight to the point now. – matyas Aug 20, 2016 at 9:33

Yes, it is useful to know that templates can be customized on the project level even if some of them can be changed optionally on the application level. – hynekcer Aug 20, 2016 at 16:19

6

The best way to do it is to put the Django admin templates inside your project. So your templates would be in `templates/admin` while the stock Django admin templates would be in say `template/django_admin`. Then, you can do something like the following:

templates/admin/change_form.html

```
{% extends 'django_admin/change_form.html' %}

Your stuff here
```

If you're worried about keeping the stock templates up to date, you can include them with svn externals or similar.

Share Improve this answer

Follow

answered Jul 5, 2011 at 14:26

Chris Pratt
239k ● 37 ● 407 ● 464

Using svn externals is a great idea. The problem this introduces is that all my translators are going to translate all those templates (because makemessages will collect the translation strings from all admin templates) which adds a lot of extra work if you're working with multiple languages. Maybe there is a way to exclude those templates from makemessages? – Semmel Jul 5, 2011 at 16:06

Use the `--ignore` argument with `makemessages`. See: docs.djangoproject.com/en/dev/ref/django-admin/#makemessages – Chris Pratt Jul 5, 2011 at 16:22

I think the the other answer fits my need better. But I like your solution and think it's a good alternative if you don't want to mess around with your template loaders. – Semmel Jul 6, 2011 at 10:15

6

for app index add this line to somewhere common py file like url.py

```
admin.site.index_template = 'admin/custom_index.html'
```

for app module index : add this line to admin.py

```
admin.AdminSite.app_index_template = "servers/servers-
```

for change list : add this line to admin class:

```
change_list_template = "servers/servers_changelist.htm
```

for app module form template : add this line to your admin class

```
change_form_template = "servers/server_changeform.html
```

etc. and find other in same admin's module classes

Share Improve this answer Follow

edited Jan 12, 2019 at 2:35

answered Jan 12, 2019 at 2:28

Saurabh Chandra Patel

13.5k ● 7 ● 92 ● 79

3

You can override django admin templates in several ways.

For example, there is `django-project` as shown below:

```
django-project
 |-core
 |   └-settings.py
 |-app1
 |   |-models.py
 |   └-admin.py
 |-app2
 └-templates
```

Then, `BASE_DIR / 'templates'` is set to `DIRS` in `TEMPLATES` in `settings.py` so that `templates` folder is recognized as shown below:

```
# "core/settings.py"

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.Dj
        'DIRS': [
            BASE_DIR / 'templates', # Here
        ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.de
                'django.template.context_processors.re
                'django.contrib.auth.context_processor
                'django.contrib.messages.context_proce
            ],
        },
    },
]
```

And, there are `Food` and `Drink` models in `app1/models.py` as shown below:

```
# "app1/models.py"

class Food(models.Model):
    name = models.CharField(max_length=20)

class Drink(models.Model):
    name = models.CharField(max_length=20)
```

And, there are `Food` and `Drink` admins in `app1/admin.py` as shown below:

```
# "app1/admin.py"

@admin.register(Food)
class FoodAdmin(admin.ModelAdmin):
```

```
    pass

@admin.register(Drink)
class DrinkAdmin(admin.ModelAdmin):
    pass
```

Now, you can override one of the django admin templates change_form.html in `templates/admin/`, `templates/admin/app1/` and `templates/admin/app1/food/` as shown below. *You can copy django admin templates from `django/contrib/admin/templates/admin/` in your virtual environment and some django admin templates cannot be overridden in `templates/admin/app1/` or `templates/admin/app1/food/` but these django admin templates can be overridden all in `templates/admin/`, `templates/admin/app1/` and `templates/admin/app1/food/` and you can see my answer explaining which django admin templates can be overridden in which directories.

`change_form.html` in `templates/admin/` below can automatically apply to all admins in all apps. *The lowercase folder name `admin` works properly:

```
django-project
 |-core
 |   └-settings.py
 |-app1
 |   |-models.py
 |   └-admin.py
 |-app2
 └-templates
```

```
    └-admin
       └-change_form.html # Here
```

`change_form.html` in `templates/admin/app1/` below can automatically apply to all admins in `app1`. *The lowercase folder name `app1` works properly:

```
django-project
 |-core
 |  └-settings.py
 |-app1
 |  |-models.py
 |  └-admin.py
 |-app2
 └-templates
    └-admin
       |-app1
       |  └-change_form.html # Here
       └-app2
```

`change_form.html` in `templates/admin/app1/food/` below can automatically apply to `food` admin in `app1`. *The lowercase folder name `food` works properly:

```
django-project
 |-core
 |  └-settings.py
 |-app1
 |  |-models.py
 |  └-admin.py
 |-app2
 └-templates
    └-admin
       |-app1
       |  |-food
       |  |  └-change_form.html # Here
```

```
|   └-drink
└-app2
```

And now, you can rename `change_form.html` to `custom_change_form.html` but `custom_change_form.html` in any folders cannot automatically apply to any admins in any apps. So, you need to manually apply `custom_change_form.html` to any admins in any apps which you want to apply `custom_change_form.html` to.

For `custom_change_form.html` in `templates/admin/` below:

```
django-project
 |-core
 |   └-settings.py
 |-app1
 |   |-models.py
 |   └-admin.py
 |-app2
 └-templates
     └-admin
         └-custom_change_form.html # Here
```

Set `admin/custom_change_form.html` to change_form_template in `Food` and `Drink` admins as shown below. *You can find more custom template options:

```
# "app1/admin.py"

@admin.register(Food)
class FoodAdmin(admin.ModelAdmin):
    change_form_template = 'admin/custom_change_form.h
```

```
@admin.register(Drink)
class DrinkAdmin(admin.ModelAdmin):
    change_form_template = 'admin/custom_change_form.h
```

For `custom_change_form.html` in `templates/admin/app1/` below:

```
django-project
 |-core
 |  └-settings.py
 |-app1
 |  |-models.py
 |  └-admin.py
 |-app2
 └-templates
    └-admin
       |-app1
       |  └-custom_change_form.html # Here
       └-app2
```

Set `admin/app1/custom_change_form.html` to `change_form_template` in `Food` and `Drink` admins as shown below:

```
# "app1/admin.py"

@admin.register(Food)
class FoodAdmin(admin.ModelAdmin):
    change_form_template = 'admin/app1/custom_change_f

@admin.register(Drink)
class DrinkAdmin(admin.ModelAdmin):
    change_form_template = 'admin/app1/custom_change_f
```

For `custom_change_form.html` in `templates/admin/app1/food` below:

```
django-project
 |-core
 |   └-settings.py
 |-app1
 |   |-models.py
 |   └-admin.py
 |-app2
 └-templates
     └-admin
         |-app1
         |   |-food
         |   |   └-custom_change_form.html # Here
         |   └-drink
         └-app2
```

Set `admin/app1/food/custom_change_form.html` to `change_form_template` in `Food` and `Drink` admins as shown below:

```
# "app1/admin.py"

@admin.register(Food)
class FoodAdmin(admin.ModelAdmin):
    change_form_template = 'admin/app1/food/custom_cha

@admin.register(Drink)
class DrinkAdmin(admin.ModelAdmin):
    change_form_template = 'admin/app1/food/custom_cha
```

Share Improve this answer Follow

edited Jul 11, 2023 at 13:48

answered May 12, 2023 at 8:09

Super Kai - Kazuya Ito

1

1

I agree with Chris Pratt. But I think it's better to create the symlink to original Django folder where the admin templates place in:

```
ln -s /usr/local/lib/python2.7/dist-
packages/django/contrib/admin/templates/admin/ templat
```

and as you can see it depends on python version and the folder where the Django installed. So in future or on a production server you might need to change the path.

Share Improve this answer

Follow

answered Mar 9, 2013 at 8:48

James May

1,537 ● 3 ● 21 ● 41

0

[This](#) site had a simple solution that worked with my Django 1.7 configuration.

FIRST: Make a symlink named **admin_src** in your project's template/ directory to your installed Django templates. For me on Dreamhost using a virtualenv, my "source" Django admin templates were in:

```
~/virtualenvs/mydomain/lib/python2.7/site-
packages/django/contrib/admin/templates/admin
```

SECOND: Create an **admin** directory in templates/

So my project's template/ directory now looked like this:

```
/templates/
   admin
   admin_src -> [to django source]
   base.html
   index.html
   sitemap.xml
   etc...
```

THIRD: In your new template/admin/ directory create a **base.html** file with this content:

```
{% extends "admin_src/base.html" %}

{% block extrahead %}
<link rel='shortcut icon' href='{{ STATIC_URL }}img/fa
{% endblock %}
```

FOURTH: Add your admin favicon-admin.ico into your static root img folder.

Done. Easy.

Share Improve this answer

Follow

answered Mar 18, 2015 at 7:06

mitchf

3,797 ● 4 ● 27 ● 29

-1

You can use django-overextends, which provides circular template inheritance for Django.

It comes from the Mezzanine CMS, from where Stephen extracted it into a standalone Django extension.

More infos you find in "Overriding vs Extending Templates" (http:/mezzanine.jupo.org/docs/content-architecture.html#overriding-vs-extending-templates) inside the Mezzanine docs.

For deeper insides look at Stephens Blog "Circular Template Inheritance for Django" (http:/blog.jupo.org/2012/05/17/circular-template-inheritance-for-django).

And in Google Groups the discussion (https:/groups.google.com/forum/#!topic/mezzanine-users/sUydcf_IZkQ) which started the development of this feature.

Note:

I don't have the reputation to add more than 2 links. But I think the links provide interesting background information. So I just left out a slash after "http(s):". Maybe someone with better reputation can repair the links and remove this note.

Share Improve this answer Follow

edited Feb 6, 2016 at 21:51

answered Feb 4, 2016 at 22:16

Henri Hulski
9 ● 2

Since Django 1.9, this project hasn't been relevant, the maintainer just isn't advertising it, see

code.djangoproject.com/ticket/15053 and github.com/stephenmcd/django-overextends/pull/37. To take complete control of which app a template is loaded from, there is django-apptemplates and django-app-namespace-template-loader, which are both still relevant if you want to extend from one app to another. – benjaoming Jul 22, 2020 at 8:47