

---

# **django-anylink Documentation**

*Release 0.0.1*

**Moccu**

**Oct 30, 2020**



---

# Contents

---

<b>1</b>	<b>What is django-anylink</b>	<b>3</b>
<b>2</b>	<b>Contents</b>	<b>5</b>
2.1	Installation . . . . .	5
2.2	TinyMCE Integration . . . . .	5
2.3	Configuration . . . . .	6
2.4	Usage . . . . .	6
2.5	Writing your own link extension . . . . .	7
2.6	Contribution . . . . .	8
<b>3</b>	<b>Indices and tables</b>	<b>9</b>



Generic linking in Django. Includes support for RichText editors like TinyMCE.



---

## What is django-anylink

---

*django-anylink* is a generic linking module for Django. Using this module, you can create links for many usecases. You'll find yourself just using the `AnyLinkField` to create links to different Django models or external urls. You don't have to take care for changing urls. `AnyLink` resolves links on request.

*django-anylink* provides a `Link` database, an model field and some handy widgets for the daily use.

Besides that, *django-anylink* is easy extendable. By default, the module provides external urls and model links wich have a `get_absolute_url` method.



## 2.1 Installation

To install *django-anylink* just use your preferred Python package installer:

```
pip install django-anylink
```

Add *anylink* to your Django settings

```
INSTALLED_APPS = (  
    # other apps  
    'anylink',  
)
```

Now, you should define at least one link extension, for example external links.

```
ANYLINK_EXTENSIONS = (  
    'anylink.extensions.ExternalLink',  
)
```

*django-anylink* auto-creates models for those migrations. To prevent them landing in Python's site packages directory, explicitly define (and create!) a module for them:

```
MIGRATION_MODULES = {  
    'anylink': 'migrations.anylink',  
}
```

Details on how to use *django-anylink* in your Django application can be found in the [Configuration](#) section.

## 2.2 TinyMCE Integration

*django-anylink* comes with a TinyMCE plugin already integrated. To use it you only have to install *django-tinymce* according to its documentation and enable the *anylink* plugin.

```
TINYMCE_DEFAULT_CONFIG = {
    'theme': 'advanced',
    'plugins': 'anylink',
    'theme_advanced_buttons1': (
        'anylink',
    ),
    'anylink_url': '/anylink/anylink/',
}
```

## 2.3 Configuration

Luckily you don't have to configure that much to use *django-anylink*.

### 2.3.1 ANYLINK\_EXTENSIONS

To add a new link target, you have to update the `ANYLINK_EXTENSIONS` setting.

This directive is a list of linkable target (external urls, Django models with `get_absolute_url` methods and so on). Every entry can be a single class path or a tuple consisting of a class path and a configuration dictionary.

#### ExternalLink

This extension provides a external url field. No other configuration is needed.

```
# Example with external links
ANYLINK_EXTENSIONS = (
    'anylink.extensions.ExternalLink',
)
```

#### ModelLink

The `ModelLink` extension provides a foreign key the configured model. It is required that the model is registered in the Django admin interface. Also, the model needs to have a `get_absolute_url` method.

```
# Example with model links with MyModel
ANYLINK_EXTENSIONS = (
    ('anylink.extensions.ModelLink', {'model': 'myapp.MyModel'}),
)
```

For details on writing your own extensions, please see the *Writing your own link extension* section.

#### Link Multiusage

To use anylink instance multiple times set `ANYLINK_ALLOW_MULTIPLE_USE` to `True`

```
# Example with app using link multiple times
ANYLINK_ALLOW_MULTIPLE_USE = True
```

## 2.4 Usage

Before you can use *django-anylink*, you have to install the module and configure it. Please see *Installation* for more details.

## 2.4.1 Adding an link to your model

To add a link field to your model, just use the AnyLinkField

```
from django.db import models

from anylink.fields import AnyLinkField

class MyModel(models.Model):
    whatever = models.CharField(max_length=255)

    link = AnyLinkField()
```

Now, you have an link field in your model. This link field is a ForeignKey internally.

## 2.4.2 Get the link url and other attributes

Lets assume, you implemented your Django model like the example above. Here is a example, how you would access the attributes of the link.

```
url = obj.link.get_absolute_url() # URL to link.
name = obj.link.text # link text/link name
title = obj.link.title # title attribute of the link
target = obj.link.target # target of the link, for example _self or _blank
css_class = obj.link.css_class # optional css class
```

---

**Hint:** Please remember, only the `get_absolute_url` method and `target` always return a values. All other attributes (`text`, `title`, `css_class` can be blank).

---

Please see the example projects for more details.

## 2.5 Writing your own link extension

To extend `django-anylink` lets assume you have a Download model. This model doesn't have a `get_absolute_url` method. Theirfore you want to write your own link extension.

Lets have a look at the code first.

```
from django.core.urlresolvers import reverse

from anylink.extensions import BaseLink

class DownloadLink(BaseLink):
    def configure_model(self, model):
        # configure_model is called by django-anylink upon initialization.
        # We add a field to anylink model to keep the object reference.
        # Make sure the field is null-able, anylink will ensure its filled out
        # if the link type is set to DownloadLink.
        model.add_to_class(self.get_name(), models.ForeignKey(
            'myapp.Download', blank=True, null=True))

    def get_absolute_url(self, link):
        # Get the obj instance using the get_name method.
        obj = getattr(link, self.get_name())
        # return a reverse'd url or None if no obj is set.
        return obj and reverse('myurl', kwargs={'id': obj.pk}) or None
```

As you can see here, the Link extension has two important methods. The `configure_model` method and the `get_absolute_url` method. Please refer to the comments and the code for more details on this topic.

## 2.6 Contribution

If you like to contribute to this project please read the following guides.

### 2.6.1 Django Code

You have to install some dependencies for development and testing.

```
$ pip install -e .[tests]
```

#### Testing the code

*django-anylink* uses `py.test` for testing. Please ensure that all tests pass before you submit a pull request. `py.test` also runs PEP8 and PyFlakes checks on every run.

We created a Makefile to make some commands more easy to run.

This is how you execute the tests and checks from the repository root directory.

```
$ py.test
```

Or with the shortcut in the Makefile.

```
$ make tests
```

If you want to generate a coverage report, you can use the following command.

```
$ make coverage
```

If you want a coverage report with html output.

```
$ make coverage-html
```

#### Documentation

*django-anylink* uses Sphinx for documentation. You find all the sources files in the `docs/source` folder.

To update/generate the html output of the documentation, use the following command inside the `docs` folder.

```
$ make html
```

Please make sure that you don't commit the build files inside `docs/build`.

### 2.6.2 JavaScript Code

TBD

## CHAPTER 3

---

### Indices and tables

---

- genindex
- search