django-any-urlfield Documentation

Release 2.4.2

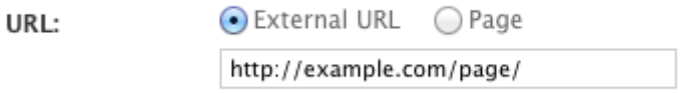
Diederik van der Boor

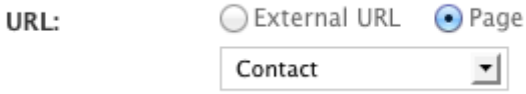
Jan 31, 2018

Contents

The `any_urlfield` module provides an improved URL selector that supports both URLs to internal models and external URLs.

This addresses is a common challenge in CMS interfaces; where providing a `URLField` makes it hard to enter internal URLs, while providing a `ModelChoiceField` makes it too inflexible.





Relevant public classes:

- Model fields:
- *`AnyUrlField`*: allow users to choose either a model or external link as URL value
- Form widget rendering:
- `HorizontalRadioFieldRenderer`
- *`SimpleRawIdWidget`*

CHAPTER 1

Contents

1.1 Installation

First install the module, preferably in a virtual environment:

```
pip install django-any-urlfield
```

Add the module to the installed apps:

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

1.1.1 Usage

Add the field to a Django model:

```
from django.db import models
from any_urlfield.models import AnyUrlField

class MyModel(models.Model):
    title = models.CharField("Title", max_length=200)
    url = AnyUrlField("URL")
```

By default, the `AnyUrlField` only supports linking to external pages.

Register any model that the `AnyUrlField` should support linking to:

```
from any_urlfield.models import AnyUrlField
AnyUrlField.register_model(Article)
```

Now, the *`AnyUrlField`* offers users a dropdown field to directly select an article.

The default field is a `django.forms.models.ModelChoiceField` field with a `django.forms.widgets.Select` widget. This can be customized using the `form_field` and `widget` parameters:

```
from any_urlfield.models import AnyUrlField
from any_urlfield.forms import SimpleRawIdWidget

AnyUrlField.register_model(Article, widget=SimpleRawIdWidget(Article))
```

That will display the `Article` model as raw input field with a browse button.

For more configuration options of the *`register_model()`* function, see the documentation of the *`AnyUrlField`* class.

1.2 API documentation

Contents:

1.2.1 any_urlfield.forms

The `AnyUrlField` class

class any_urlfield.forms.**AnyUrlField**(*url_type_registry*, *max_length=None*, **args*, ***kwargs*)

Form field that combines a Page ID and external page URL.

The form field is used automatically when the *`AnyUrlField`* is used in the model.

widget

alias of *`AnyUrlWidget`*

The `AnyUrlWidget` class

class any_urlfield.forms.**AnyUrlWidget**(*url_type_registry*, *attrs=None*)

The URL widget, rendering the URL selector.

The `SimpleRawIdWidget` class

class any_urlfield.forms.**SimpleRawIdWidget**(*model*, *limit_choices_to=None*, *admin_site=None*, *attrs=None*, *using=None*)

A wrapper class to create raw ID widgets.

It produces a same layout as the `raw_id_fields = (field',)` code does in the admin interface. This class wraps the functionality of the Django admin application into a usable format that is both compatible with Django 1.3 and 1.4.

The basic invocation only requires the model:

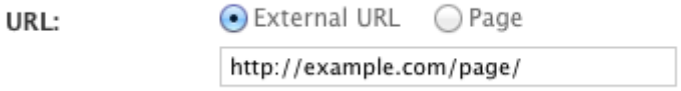
```
widget = SimpleRawIdWidget(MyModel)
```

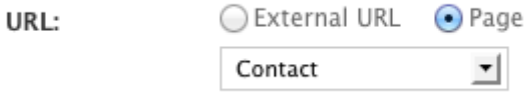
1.2.2 any_urlfield.models

The `AnyUrlField` class

class any_urlfield.models.**AnyUrlField**(**args*, ***kwargs*)

A CharField that can either refer to a CMS page ID, or external URL.





By default, the `AnyUrlField` only supports linking to external pages. To add support for your own models (e.g. an `Article` model), include the following code in `models.py`:

```
from any_urlfield.models import AnyUrlField
AnyUrlField.register_model(Article)
```

Now, the `AnyUrlField` offers users a dropdown field to directly select an article. By default, it uses a `django.forms.ModelChoiceField` field with a `django.forms.Select` widget to render the field. This can be customized using the `form_field` and `widget` parameters:

```
from any_urlfield.models import AnyUrlField
from any_urlfield.forms import SimpleRawIdWidget

AnyUrlField.register_model(Article, widget=SimpleRawIdWidget(Article))
```

Now, the `Article` model will be displayed as raw input field with a browse button.

classmethod register_model(*ModelClass*, *form_field=None*, *widget=None*, *title=None*, *prefix=None*)

Register a model to use in the URL field.

This function needs to be called once for every model that should be selectable in the URL field.

Parameters

- **ModelClass** – The model to register.
- **form_field** – The form field class used to render the field. This can be a lambda for lazy evaluation.
- **widget** – The widget class, can be used instead of the form field.
- **title** – The title of the model, by default it uses the models `verbose_name`.
- **prefix** – A custom prefix for the model in the serialized database format. By default it uses "appname.modelname".

The `AnyUrlValue` class

class any_urlfield.models.**AnyUrlValue**(*type_prefix*, *type_value*, *url_type_registry=None*)

Custom value object for the *AnyUrlField*. This value holds both the internal page ID, and external URL. It can be used to parse the database contents:

```
value = AnyUrlValue.from_db_value(url)
article = value.get_object()
print unicode(value)
```

A conversion to `unicode` or `str` causes the URL to be generated. This allows the field value to be used in string concatenations, or template variable evaluations:

```
{{ mymodel.url }}
```

exists()

Check whether the references model still exists.

classmethod from_db_value(*url*, *url_type_registry=None*)

Convert a serialized database value to this object.

The value can be something like:

- an external URL: `http://..`, `https://..`
- a custom prefix: `customid://214`, `customid://some/value`
- a default "app.model" prefix: `appname.model://31`

classmethod from_model(*model*, *url_type_registry=None*)

Convert a model value to this object.

get_model()

Return the model that this value points to.

get_object()

Return the database object that the value points to.

to_db_value()

Convert the value into a serialized format which can be stored in the database. For example: `http://www.external.url/` or `pageid://22`.

type_prefix

Return the URL type prefix. For external URLs this is always `"http"`.

1.3 Changelog

1.3.1 Changes in git

- Dropped Django 1.4, 1.5, 1.6 support

1.3.2 Version 2.4.2 (2017-07-31)

- Fixed form `has_changed` check, preventing inline fieldsets to be submitted.
- Fixed widget alignment inside inlines.

1.3.3 Version 2.4.1 (2017-05-05)

- Fixed packaging bugs that prevented including the HTML templates for Django 1.11.

1.3.4 Version 2.4 (2017-05-01)

- Added Django 1.11 support.
- Dropped Python 2.6 support.
- Fix for empty value.

1.3.5 Version 2.3 (2017-02-03)

- For Django 1.8 and up, the `URLValidator` now allows more URL schemes by default, specifically `mailto:` and `tel:` URLs.

1.3.6 Version 2.2.1 (2016-02-26)

- Fixed Django 1.10 deprecation warnings.

1.3.7 Version 2.2 (2015-12-30)

- Added Django 1.9 support
- Fixed saving blank/null values.

1.3.8 Version 2.1.1 (2015-04-15)

- Fix Django 1.7/1.8 model saving issues.
- Added `AnyUrlValue.from_model()` to directly wrap a model into an `AnyUrlValue`.

1.3.9 Version 2.1 (2015-04-10)

- Added Django 1.8 support
- Fix importing json fixture data.

Released as 2.1a1: (2014-09-15)

- Added caching support for URL values.

1.3.10 Version 2.0.4 (2014-12-30)

- Fixed Python 3.3 issues

1.3.11 Version 2.0.3 (2014-10-30)

- Fixed `__eq__()` for comparing against other object types.

1.3.12 Version 2.0.2 (2014-10-30)

- Added pickle support.
- Fixed Django 1.7 support.

1.3.13 Version 2.0.1 (2014-09-15)

- Fix performance issue with form fields.

1.3.14 Version 2.0 (2014-08-15)

Released as 2.0b1 (2014-06-05)

- Improved Python 3 support.
- Delay initialisation of `ModelChoiceField` objects.
- Fix `exists()` value for empty URLs

Released as 2.0a1 (2014-04-04)

- Added Python 3 support
- Allow passing callables to the form_field parameter of `AnyUrlField.register_model`

1.3.15 Version 1.0.12 (2014-02-24)

- Implement `AnyUrlField.__deepcopy__()` to workaround Django < 1.7 issue, where `__deepcopy__()` is missing for `MultiValueField` classes.

1.3.16 Version 1.0.11 (2014-02-20)

- Improve external URL support (https, ftps, smb, etc..)
- Fix unnecessary query at registration of custom models.

1.3.17 Version 1.0.10 (2013-12-12)

- Fix using `AnyUrlField` with `blank=True`.
- Fix `_has_changed` is no longer used in django >= 1.6.0

1.3.18 Version 1.0.9 (2013-10-15)

- Fixed exporting the value in the `dumpdata` command.

1.3.19 Version 1.0.8 (2013-09-20)

- Use `long()` for ID's, not `int()`.
- Improve `ObjectDoesNotExist` check in `AnyUrlValue.__unicode__()`, to support model translations.

1.3.20 Version 1.0.7 (2013-05-28)

- Fix using this widget with Django 1.6 alpha 1

1.3.21 Version 1.0.5 (2013-05-07)

- Fix errors during south migration
- Fix errors when deleting rows in an inline formset which uses an `AnyUrlField`.

1.3.22 Version 1.0.4 (2013-05-02)

- Fix https URL support

1.3.23 Version 1.0.3 (2013-04-24)

- Fix change detection, to support formsets and admin inlines.
- Fix widget alignment within a `TabularInline`.

1.3.24 Version 1.0.2 (2013-01-24)

- Fix `setup.py` code to generate translation files for the `sdist`.
- Remove `HorizonatalRadioFieldRenderer` from the public API.

1.3.25 Version 1.0.1 (2012-12-27)

- Use jQuery live events to support using the `AnyUrlField` in Django inlines.

1.3.26 Version 1.0.0 (2012-12-27)

First PyPI release.

The module design has been stable for quite some time, so it's time to release this module to the public.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index