django-slugify-processor Documentation

Release 0.8.4

devel.tech

Jul 04, 2020

CONTENTS

pipeline for handling slugification edgecases

CHAPTER

ONE

WHAT ARE SLUGS?

Slugs are URL's, typically generated from post titles, that you want to be both human readable and a valid URL. They are SEO friendly.

Django provides a slugify function in `django.utils.text.slugify` which is also made available as a default filter.

Django slugs can be automatically generated in django models via packages such as:

- django-autoslug (docs) (github)
- django-extensions via AutoSlugField (docs) (github)

CHAPTER

TWO

THE PROBLEM

This project is based on an article from devel.tech covering django's import strings.

Corner cases exist with slugification. For instance:

Term	`django.utils.text.slugify`	What you want
C	c (correct)	n/a
C++	c	cpp
C#	c	c-sharp

To make matters worse, if using a specialized model field like `AutoSlugField` from django-autoslug or django-extensions, the default behavior may be to name the slugs for C++ and C# to "c-1", "c-2" after "c" is taken.

Here's another case, acronyms / shorthands:

Term	`django.utils.text.slugify`	What you (may) want
New York City	new-york-city	nyc
Y Combinator	y-combinator	yc
Portland	portland	pdx
Texas	texas	tx
$	'' (empty)	usd, aud, etc?
US$	us	usd
A$	a	aud
bitcoin	bitcoin	btc
United States	united-states	usa
League of Legends	league-of-legends	league
Apple® iPod Touch	apple-ipod-touch	ipod-touch

Each website and niche has its own edge cases for slugs. So we need a solution that can scale, where you can craft your own functions.

CHAPTER

THREE

HOW DJANGO-SLUGIFY-PROCESSOR HELPS

This builds on top of django.utils.text.slugify to handle your django project's edgecases. By default, django-slugify-processor will be a pass through to django's default behavior. Adding slugification functions via your Django project's settings file allows you to adjust.

CHAPTER

FOUR

INSTALLATION

```
$ pip install django-slugify-processor
```

CHAPTER

FIVE

CONFIGURE

To create a processor, create a function that accepts a string, and returns a string. Assume this is *project/app/slugify_processors.py*:

```
def my_processor(value):
   value = value.replace('++', 'pp')
   return value
```

Inside of your settings, add a `SLUGIFY_PROCESSORS` list of strings that points to the function. Anything that's compatible with import_string, in your settings file:

```
SLUGIFY_PROCESSORS = [
   'project.app.slugify_processors.my_processor'
]
```

CHAPTER

SIX

USAGE

6.1 In normal django code

Import `slugify` from `django_slugify_processor.text`:

```
from django_slugify_processor.text import slugify

print(slugify('C++'))
> 'cpp'
```

6.2 Template code

django-slugify-processor is designed to override the built-in``slugify`` filter.

6.2.1 via load

You can load by default via `{% load django_slugify_processor %}` in your template.

In your settings `INSTALLED_APPS`:

```
INSTALLED_APPS = [
    'django_slugify_processor'
]
```

In your template:

```
{% load slugify_processor %}
{{"C++"|slugify}}
```

6.2.2 via built-in

To make this available in all templates, in the `OPTIONS` of your template engine, add `django_slugify_processor.template_tags`:

```
TEMPLATES = [{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'OPTIONS': {
        'builtins': [
```

(continues on next page)

(continued from previous page)

```
            'django_slugify_processor.templatetags.slugify_processor',
        ],
    },
}]
```

From within the template file:

```
{{"C++"|slugify}}
```

Output should be: cpp

6.3 Models

For the most up to date documentation, view the documetation for the plugin you're using (e.g. django-autoslug or django-extensions).

To use django-slugify-processor's `slugify` instead of django's default, there will be a field option to use the function.

6.3.1 django-extensions

Tested with 1.9.7 (2017-11-26):

```
from django.db import models

from django_extensions.db.fields import AutoSlugField
from django_slugify_processors.text import slugify

class MyModel(models.Model):
    title = models.CharField(max_length=255)
    slug = AutoSlugField(
        populate_from='title',
        slugify_function=slugify
    )
```

6.3.2 django-autoslug

Tested with 1.9.3 (2017-11-26):

```
from django.db import models

from autoslug import AutoSlugField
from django_slugify_processors.text import slugify

class MyModel(models.Model):
    title = models.CharField(max_length=255)
    slug = AutoSlugField(
        populate_from='title',
        slugify=slugify
    )
```

CHAPTER

SEVEN

CREDITS

- travis.yml and tox.ini based off DRF's (BSD 2-clause licensed)
- yapf configuration based off RTD / devel.tech's (MIT-licensed)

CHAPTER

EIGHT

PROJECT DETAILS

python support	>= 3.6, pypy3
django support	3.0, 2.2
Source	https://github.com/develtech/django-slugify-processor
Docs	https://django-slugify-processor.devel.tech
API	https://django-slugify-processor.devel.tech/en/latest/api.html
Changelog	https://django-slugify-processor.devel.tech/en/latest/history.html
Issues	https://github.com/develtech/django-slugify-processor/issues
Travis	http://travis-ci.org/develtech/django-slugify-processor
Test Coverage	https://codecov.io/gh/develtech/django-slugify-processor
pypi	https://pypi.python.org/pypi/django-slugify-processor
Open Hub	https://www.openhub.net/p/django-slugify-processor
License	MIT
git repo	`$ git clone https://github.com/develtech/↪django-slugify-processor.git`
install stable	`$ pip install django-slugify-processor`

install dev	`$ git clone https://github.com/develtech/↪django-slugify-processor.git`
	`$ cd ./django-slugify-processor`
	`$ pipenv install --dev --skip-lock`
	`$ pipenv shell`

tests	`$ make test`

Contents:

8.1 API Reference

8.1.1 Slugify function

`django_slugify_processor.text.`**`slugify`**(*value*, *allow_unicode=False*)

Override default slugify in django to handle custom scenarios.

Run value through functions declared in SLUGIFY_PROCESSORS. The value is then passed-through to django's slugify.

Parameters

- **value** (*string*) – string to slugify
- **allow_unicode** (*bool*) – whether or not to allow unicode (e.g. chinese)

Examples of slugify processors, assume *project/app/slugify_processors.py*:

```
def slugify_programming_languages(value):
    value = value.lower()

    value = value.replace('c++', 'cpp')
    value = value.replace('c#', 'c-sharp')
    return value

def slugify_geo_acronyms(value):
    value = value.lower()

    value = value.replace('New York City', 'nyc')
    value = value.replace('United States', 'usa')
    return value

def slugify_us_currency(value):
    value = value.lower()

    value = value.replace('$', 'usd')
    value = value.replace('US$', 'usd')
    value = value.replace('US Dollar', 'usd')
    value = value.replace('U.S. Dollar', 'usd')
    return value
```

Settings:

```
SLUGIFY_PROCESSORS = [
    'project.app.slugify_programming_languages',
    'project.app.slugify_geo_acronyms',
    'project.app.slugify_us_currency',
]
```

8.1.2 Template tag

`django_slugify_processor.templatetags.slugify_processor.slugify(*args, **kwargs)`

Template filter intended to override django 1.11+'s default slugify.

This can be installed via a builtin, or via `{% load slugify_processor %}`.

Usage in a Django template:

```
{% load slugify_processor %}  {# unless you added it to builtins %}
{{variable|slugify}}  {# assuming "variable" is in context %}
{{"C++"|slugify}}
```

8.2 History

8.2.1 0.8.3 <2017-12-02>

- Add test_app, and tests for django-extensions and django-autoslug

8.2.2 0.8.2 <2017-12-02>

- Try to tweak README / fix on PyPI

8.2.3 0.8.1 <2017-12-01>

- README updates
- Support for overriding builtin slugify in templates
- Move template filter to *templatetags/slugify_processor.py*

8.2.4 0.8.0 <2017-11-26>

- Initial commit

INDEX

S