
django-nano Documentation

Release 0.10.0

kaleissin

Nov 20, 2018

Contents

1	Installation	3
2	Usage	5
2.1	Common for all apps	5
2.2	Specific apps	5
2.2.1	activation	5
2.2.2	badge	5
2.2.3	blog	6
2.2.4	chunk	6
2.2.5	comments	6
2.2.6	countries	7
2.2.7	faq	7
2.2.8	privmsg	7
2.2.9	tools	7
2.2.10	user	8
3	Indices and tables	11
	Python Module Index	13

Does less!

This is a set of nano-size tools and apps for Django 1.4 and later.

Contents:

CHAPTER 1

Installation

Thanks for downloading django-nano.

Install via - unpacking and running 'python setup.py install' - pip install django-nano - easy_install django-nano

These tools and applications require Python 2.6 or later, and Django 1.3 or later. You can obtain Python from <http://www.python.org/> and Django from <http://www.djangoproject.com/>.

This version will run on Django 1.3 if you rewrite the templates (search for `{% url '...` and Django 1.4 and newer otherwise.

2.1 Common for all apps

Append `nano.<subapp>` to your `INSTALLED_APPS`, where `subapp` is any of the tools listed above except `tools`.

Check the docs for the apps themselves for anything app-specific.

2.2 Specific apps

2.2.1 activation

A place to store activation-codes for e.g. authentication

```
nano.activation.baseNgenerator (base=10, keylength=5, step=1)  
    Generate keys of base <base> and length <keylength>
```

```
nano.activation.generate_keys (generator, amount=50)  
    Generate <amount> keys with <generator>
```

```
nano.activation.to_base (num, base, numerals='0123456789abcdefghijklmnopqrstuvwxyz')  
    Convert <num> to <base> using the symbols in <numerals>
```

Models

2.2.2 badge

User-badges worth certain points ala. StackOverflow

```
nano.badge.add_badge (badge, model)  
    Put a badge on a model
```

`nano.badge.batchbadge` (*badge, queryset*)

Put a badge on all models that do not already have the badge

Models

2.2.3 blog

A very basic blog-app.

It has optional support for *django-taggit*.

Models

Tools

Changes to settings

NANO_BLOG_USE_TAGS (optional) Set to True to use *django-taggit* if it is installed.

Default: Not set

NANO_BLOG_SPECIAL_TAGS (optional) A list of tags that may be treated specially.

Default: ('pinned',)

2.2.4 chunk

A chunk is a template that is stored in the database.

It has a unique name, the `slug`, and some `content`, which is whatever you'd put in an ordinary template.

You use it by setting the `template_name` to the slug of the chunk, or by `{% include %}`-ing it directly.

If the chunk uses non-builtin template tags, remember to `{% load %}` the template tag library in the chunk.

There's a model and a template loader:

Models

Chunks can be created, updated and deleted via the django admin.

Changes to settings

Add `'nano.chunk.loader.Loader'` to `TEMPLATE_LOADERS`.

2.2.5 comments

Unmoderated comments for logged-in users.

Models

Views

Forms

Changes to settings

COMMENT_MAX_LENGTH (optional)

Default: 3000

2.2.6 countries

Drop-in, nanofied replacement for <https://code.google.com/p/django-countries/> .

To use: import from `nano.countries` instead of `countries`. The primary key is the two-letter iso country code, `name` and `printable_name` points to the same thing: what was known as `printable_name` in `django-countries`.

There is an admin, there are no template tags, views, forms or fields.

Models

2.2.7 faq

Just about as simple a FAQ as is possible

Models

2.2.8 privmsg

Private messages with separate archives for sent and received

Models

2.2.9 tools

Utility-functions used by some of the other apps.

exception `nano.tools.SiteProfileNotAvailable`

`nano.tools.asciiify` (*string*)

Convert unicode string to ascii, normalizing with NFKD

Strips away all non-ascii symbols

`nano.tools.grouper` (*n, iterable, fillvalue=None*)

`grouper(3, 'ABCDEFGF', 'x') -> ABC DEF Gxx`

Models

Tree

Denormalized text-field

Model with one generic foreign key

Template tags

Either import the tags into some other templatetags-library. or add 'nano.tools' to INSTALLED_APPS.

```
nano.tools.templatetags.nano_tags.come_back(context)
```

Turn current url path into a query-part for use in urls

With default template:

If the path is stored in the context as /foo/bar, tag returns '?next=/foo/bar'

```
nano.tools.templatetags.nano_tags.endswith(value, arg)
```

Usage { % if valueendswith:"arg" % }

```
nano.tools.templatetags.nano_tags.fraction(text, arg=1)
```

Get fractional part of float, pad with zeroes until <arg> length

```
nano.tools.templatetags.nano_tags.integer(text)
```

Get integer-part of float

```
nano.tools.templatetags.nano_tags.nbr(*args, **kwargs)
```

Replace whitespace with non-breaking-space

```
nano.tools.templatetags.nano_tags.partition(iterable, cols=4)
```

Split an iterable into columns

```
nano.tools.templatetags.nano_tags.startswith(value, arg)
```

Usage { % if valuestartswith:"arg" % }

2.2.10 user

Doesn't have any models so just hook up the views in an urls.py:

```
exception nano.user.views.NanoUserError
```

```
exception nano.user.views.NanoUserExistsError
```

- `signup()`
- `password_change()`
- `password_reset()`

Changes to settings

NANO_USER_EMAIL_SENDER The From:-address on a password-reset email. If unset, no email is sent.

Default: Not set

NANO_USER_TEST_USERS Special-cased usernames for live testing.

Default: ()

NANO_USER_BLOG_TEMPLATE Template used for auto-blogging new users.

Default: `blog/new_user.html`

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

n

- `nano.activation`, 5
- `nano.badge`, 5
- `nano.blog`, 6
- `nano.chunk`, 6
- `nano.comments`, 6
- `nano.countries`, 7
- `nano.faq`, 7
- `nano.privmsg`, 7
- `nano.tools`, 7
- `nano.tools.templatetags.nano_tags`, 8
- `nano.user`, 9
- `nano.user.views`, 8

A

activation
 nano.activation, 5
add_badge() (in module nano.badge), 5
asciiify() (in module nano.tools), 7

B

badge
 nano.badge, 5
baseNgenerator() (in module nano.activation), 5
batchbadge() (in module nano.badge), 5
blog
 nano.blog, 6

C

chunk
 nano.chunk, 6
come_back() (in module
 nano.tools.templatetags.nano_tags), 8
comments
 nano.comments, 6
countries
 nano.countries, 7

E

endswith() (in module
 nano.tools.templatetags.nano_tags), 8

F

faq
 nano.faq, 7
fraction() (in module nano.tools.templatetags.nano_tags),
 8

G

generate_keys() (in module nano.activation), 5
grouper() (in module nano.tools), 7

I

integer() (in module nano.tools.templatetags.nano_tags),
 8

N

nano.activation
 activation, 5
nano.activation (module), 5
nano.badge
 badge, 5
nano.badge (module), 5
nano.blog
 blog, 6
 settings, 6
nano.blog (module), 6
nano.chunk
 chunk, 6
 settings, 6
nano.chunk (module), 6
nano.comments
 comments, 6
nano.comments (module), 6
nano.countries
 countries, 7
nano.countries (module), 7
nano.faq
 faq, 7
nano.faq (module), 7
nano.privmsg
 privmsg, 7
nano.privmsg (module), 7
nano.tools
 tools, 7
nano.tools (module), 7
nano.tools.templatetags.nano_tags (module), 8
nano.user
 settings, 8
 user, 8
nano.user (module), 9

nano.user.views (module), [8](#)
NanoUserError, [8](#)
NanoUserExistsError, [8](#)
nbr() (in module nano.tools.templatetags.nano_tags), [8](#)

P

partition() (in module nano.tools.templatetags.nano_tags), [8](#)
privmsg
 nano.privmsg, [7](#)

S

settings
 nano.blog, [6](#)
 nano.chunk, [6](#)
 nano.user, [8](#)
SiteProfileNotAvailable, [7](#)
startswith() (in module nano.tools.templatetags.nano_tags), [8](#)

T

to_base() (in module nano.activation), [5](#)
tools
 nano.tools, [7](#)

U

user
 nano.user, [8](#)