

---

# Django utils2 Documentation

*Release 2.8.0*

**Rick van Hattem**

**Mar 22, 2020**



<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Install . . . . .	3
1.2	Links . . . . .	4
<b>2</b>	<b>django_utils package</b>	<b>5</b>
2.1	Subpackages . . . . .	5
2.1.1	django_utils.management package . . . . .	5
2.1.1.1	Subpackages . . . . .	5
2.1.1.1.1	django_utils.management.commands package . . . . .	5
2.1.1.1.1.1	Submodules . . . . .	5
2.1.1.1.1.2	django_utils.management.commands.admin_autogen module . . . . .	5
2.1.1.1.1.3	django_utils.management.commands.base_command module . . . . .	5
2.1.1.1.1.4	django_utils.management.commands.settings module . . . . .	6
2.1.1.1.1.5	Module contents . . . . .	6
2.1.1.2	Module contents . . . . .	6
2.1.2	django_utils.templatetags package . . . . .	6
2.1.2.1	Submodules . . . . .	6
2.1.2.2	django_utils.templatetags.debug module . . . . .	6
2.1.2.3	Module contents . . . . .	9
2.2	Submodules . . . . .	9
2.3	django_utils.base_models module . . . . .	9
2.4	django_utils.choices module . . . . .	11
2.4.1	Usage . . . . .	11
2.4.1.1	Example with explicit values: . . . . .	11
2.4.1.2	Example with implicit values: . . . . .	12
2.5	django_utils.fields module . . . . .	14
2.6	django_utils.queryset module . . . . .	14
2.7	django_utils.utils module . . . . .	15
2.8	django_utils.view_decorators module . . . . .	15
2.9	django_utils.views module . . . . .	15
2.10	Module contents . . . . .	15
<b>3</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>
	<b>Index</b>	<b>21</b>



Contents:



Travis status:



Coverage:



Django Utils is a collection of small Django helper functions, utilities and classes which make common patterns shorter and easier. It is by no means a complete collection but it has served me quite a bit in the past and I will keep extending it.

Examples are:

- Enum based choicefields
- Models with automatic `__str__`, `__unicode__` and `__repr__` functions based on names and/or slugs using simple mixins.
- Models with automatic `updated_at` and `created_at` fields
- Models with automatic slugs based on the `name` property.
- Iterating through querysets in predefined chunks to prevent out of memory errors

The library depends on the Python Utils library.

Documentation is available at: <http://django-utils-2.readthedocs.io/en/latest/>

## 1.1 Install

To install:

1. Run `pip install django-utils2` or execute `python setup.py install` in the source directory
2. Add `django_utils` to your `INSTALLED_APPS`

If you want to run the tests, run `py.test` (requirements in `tests/requirements.txt`)

## 1.2 Links

- **Documentation**
  - <http://django-utils-2.readthedocs.org/en/latest/>
- **Source**
  - <https://github.com/WoLpH/django-utils>
- **Bug reports**
  - <https://github.com/WoLpH/django-utils/issues>
- **Package homepage**
  - <https://pypi.python.org/pypi/django-utils2>
- **My blog**
  - <http://w.wol.ph/>



## 2.1 Subpackages

### 2.1.1 django\_utils.management package

#### 2.1.1.1 Subpackages

##### 2.1.1.1.1 django\_utils.management.commands package

###### 2.1.1.1.1.1 Submodules

###### 2.1.1.1.1.2 django\_utils.management.commands.admin\_autogen module

```
class django_utils.management.commands.admin_autogen.Command
    Bases: django_utils.management.commands.base_command.CustomBaseCommand

    handle (*args, **kwargs)
```

###### 2.1.1.1.1.3 django\_utils.management.commands.base\_command module

```
class django_utils.management.commands.base_command.CustomAppCommand
    Bases: django_utils.management.commands.base_command.CustomBaseCommand,
    django.core.management.base.AppCommand

class django_utils.management.commands.base_command.CustomBaseCommand
    Bases: django.core.management.base.BaseCommand, python_utils.logger.Logged

    create_logger()

    handle (*args, **kwargs)

    loggers = ()
```

#### 2.1.1.1.1.4 django\_utils.management.commands.settings module

**class** django\_utils.management.commands.settings.**Command**

Bases: *django\_utils.management.commands.base\_command.CustomBaseCommand*

**add\_arguments** (*parser*)

**can\_import\_settings** = True

**handle** (*\*args, \*\*options*)

**help** = 'Get a list of the current settings, any arguments given will be\n used to match'

**output\_types** = ['pprint', 'print', 'json', 'csv']

**render\_output** (*data, output\_type='pprint', show\_keys=False, \*\*options*)

**requires\_model\_validation** = False

django\_utils.management.commands.settings.**json\_default** (*obj*)

#### 2.1.1.1.1.5 Module contents

#### 2.1.1.2 Module contents

### 2.1.2 django\_utils.templatetags package

#### 2.1.2.1 Submodules

#### 2.1.2.2 django\_utils.templatetags.debug module

**class** django\_utils.templatetags.debug.**Formatter** (*max\_depth=3*)

Bases: *django\_utils.templatetags.debug.\_Formatter*

**MAX\_LENGTH** = 100

**MAX\_LENGTH\_DOTS** = 3

**format** (*value, depth, show\_protected, show\_special*)

Call the formatter with the given value to format and optional depth

```
>>> formatter = Formatter()
>>> class Eggs: pass
>>> formatter(Eggs)
'<Eggs {}>'
```

**format\_datetime** (*value, depth, show\_protected, show\_special*)

Format a date

#### Parameters

- **value** – a date to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> formatter(datetime.date(2000, 1, 2))
'<date:2000-01-02>'
>>> formatter(datetime.datetime(2000, 1, 2, 3, 4, 5, 6))
'<datetime:2000-01-02 03:04:05.000006>'
```

**format\_dict** (*value, depth, show\_protected, show\_special*)

Format a string

#### Parameters

- **value** – a str value to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> formatter({'a': 1, 'b': 2}, 5)
'{a: 1, b: 2}'
```

**format\_int** (*value, depth, show\_protected, show\_special*)

Format an integer/long

#### Parameters

- **value** – an int/long to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> str(formatter(1, 0))
'1'
>>> formatter(1, 1)
'1'
```

**format\_list** (*value, depth, show\_protected, show\_special*)

Format a string

#### Parameters

- **value** – a list to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> formatter(list(range(5)))
'[0, 1, 2, 3, 4]'
```

**format\_model** (*value, depth, show\_protected, show\_special*)

Format a string

#### Parameters

- **value** – a str value to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> from django.contrib.auth.models import User
>>> user = User()
>>> del user.date_joined
>>> str(formatter(user, 5, show_protected=False)[:30])
'<User {email: , first_name: , '
```

**format\_object** (*value, depth, show\_protected, show\_special*)

Format an object

#### Parameters

- **value** – an object to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> original_max_length = formatter.MAX_LENGTH
>>> formatter.MAX_LENGTH = 50
```

```
>>> class Spam(object):
...     x = 1
...     _y = 2
...     __z = 3
...     __hidden_ = 4
>>> spam = Spam()
```

```
>>> str(formatter(spam, show_protected=True, show_special=True))
'<Spam {x: 1, _Spam__hidden_: 4, _Spam__z: 3, __dict__:...}>'
>>> str(formatter(spam, show_protected=False, show_special=False))
'<Spam {x: 1}>'
```

```
>>> formatter.MAX_LENGTH = original_max_length
```

**format\_str** (*value, depth, show\_protected, show\_special*)

Format a string

#### Parameters

- **value** – a str value to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> str(formatter('test'))
'test'
>>> str(formatter(six.b('test')))
'test'
```

**format\_unicode** (*value, depth, show\_protected, show\_special*)

Format a string

#### Parameters

- **value** – a unicode value to format
- **depth** – the current depth

**Returns** a formatted string

```
>>> formatter = Formatter()
>>> original_max_length = formatter.MAX_LENGTH
>>> formatter.MAX_LENGTH = 10
>>> str(formatter('x' * 11))
'xxxxxxx...'
>>> formatter.MAX_LENGTH = original_max_length
```

`django_utils.templatetags.debug.debug` (*value*, *max\_depth=3*)

Debug template filter to print variables in a pretty way

```
>>> str(debug(123).strip())
'<pre style="border: 1px solid #fcc; background-color: #ccc;">123</pre>'
```

### 2.1.2.3 Module contents

## 2.2 Submodules

### 2.3 django\_utils.base\_models module

**class** `django_utils.base_models.CreatedAtModelBase` (*\*args*, *\*\*kwargs*)

Bases: `django_utils.base_models.ModelBase`

**class** `Meta`

**abstract** = `False`

**db\_table** = `'django_utils_created_at_model_base'`

**created\_at**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**get\_next\_by\_created\_at** (*\*\*morekwargs*)

**get\_next\_by\_updated\_at** (*\*\*morekwargs*)

**get\_previous\_by\_created\_at** (*\*\*morekwargs*)

**get\_previous\_by\_updated\_at** (*\*\*morekwargs*)

**updated\_at**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**class** `django_utils.base_models.ModelBase` (*\*args*, *\*\*kwargs*)

Bases: `django.db.models.base.Model`

**class** `Meta`

**abstract** = `False`

**db\_table** = `'django_utils_model_base'`

**class** `django_utils.base_models.ModelBaseMeta`

Bases: `django.db.models.base.ModelBase`

Model base with more readable naming convention

Example: Assuming the model is called `app.FooBarObject`

Default Django table name: `app_foobarobject` Table name with this base: `app_foo_bar_object`

**class** `django_utils.base_models.NameCreatedAtModelBase` (\*args, \*\*kwargs)

Bases: `django_utils.base_models.NameModelBase`, `django_utils.base_models.CreatedAtModelBase`

**class** `Meta`

`abstract = False`

`db_table = 'django_utils_name_created_at_model_base'`

`get_next_by_created_at` (\*\*morekwargs)

`get_next_by_updated_at` (\*\*morekwargs)

`get_previous_by_created_at` (\*\*morekwargs)

`get_previous_by_updated_at` (\*\*morekwargs)

**class** `django_utils.base_models.NameMixin`

Bases: `object`

Mixin to automatically get a unicode and repr string base on the name

```
>>> x = NameMixin()
>>> x.pk = 123
>>> x.name = 'test'
>>> repr(x)
'<NameMixin[123]: test>'
>>> str(x)
'test'
>>> str(six.text_type(x))
'test'
```

**class** `django_utils.base_models.NameModelBase` (\*args, \*\*kwargs)

Bases: `django_utils.base_models.NameMixin`, `django_utils.base_models.ModelBase`

**class** `Meta`

`abstract = False`

`db_table = 'django_utils_name_model_base'`

**name**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**class** `django_utils.base_models.SlugCreatedAtModelBase` (\*args, \*\*kwargs)

Bases: `django_utils.base_models.SlugModelBase`, `django_utils.base_models.CreatedAtModelBase`

**class** `Meta`

`abstract = False`

```

    db_table = 'django_utils_slug_created_at_model_base'

    get_next_by_created_at (**morekwargs)

    get_next_by_updated_at (**morekwargs)

    get_previous_by_created_at (**morekwargs)

    get_previous_by_updated_at (**morekwargs)

```

```

class django_utils.base_models.SlugMixin
    Bases: django_utils.base_models.NameMixin

```

Mixin to automatically slugify the name and add both a name and slug to the model

```

>>> x = NameMixin()
>>> x.pk = 123
>>> x.name = 'test'
>>> repr(x)
'<NameMixin[123]: test>'
>>> str(x)
'test'
>>> str(six.text_type(x))
'test'

```

```

class Meta

```

```

    Bases: object

```

```

    unique_together = ('slug',)

```

```

    save (*args, **kwargs)

```

```

class django_utils.base_models.SlugModelBase(*args, **kwargs)

```

```

    Bases: django_utils.base_models.SlugMixin, django_utils.base_models.NameModelBase

```

```

class Meta

```

```

    abstract = False

```

```

    db_table = 'django_utils_slug_model_base'

```

```

slug

```

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

## 2.4 django\_utils.choices module

### 2.4.1 Usage

Create a *Choices* class and add *Choice* objects to the class to define your choices.

#### 2.4.1.1 Example with explicit values:

The normal Django version:

```
class Human(models.Model):
    GENDER = (
        ('m', 'Male'),
        ('f', 'Female'),
        ('o', 'Other'),
    )
    gender = models.CharField(max_length=1, choices=GENDER)
```

The Django Utils Choices version:

```
from django_utils import choices

class Human(models.Model):
    class Gender(choices.Choices):
        Male = choices.Choice('m')
        Female = choices.Choice('f')
        Other = choices.Choice('o')

    gender = models.CharField(max_length=1, choices=Gender)
```

To reference these properties:

```
Human.create(gender=Human.Gender.Male)
```

### 2.4.1.2 Example with implicit values:

The normal Django version:

```
class SomeModel(models.Model):
    SOME_ENUM = (
        (1, 'foo'),
        (2, 'bar'),
        (3, 'spam'),
        (4, 'eggs'),
    )
    enum = models.IntegerField(choices=SOME_ENUM, default=1)
```

The Django Utils Choices version:

```
from django_utils import choices

class SomeModel(models.Model):
    class Enum(choices.Choices):
        Foo = choices.Choice()
        Bar = choices.Choice()
        Spam = choices.Choice()
        Eggs = choices.Choice()

    enum = models.IntegerField(
        choices=Enum, default=Enum.Foo)
```

To reference these properties:

```
SomeModel.create(enum=SomeModel.Enum.Spam)
```

```
class django_utils.choices.Choice(value=None, label=None)
    Bases: object
```



The choice object has an optional label and value. If the value is not given an autoincrementing id (starting from 1) will be used

```
>>> choice = Choice('value', 'label')
>>> choice
<Choice[1]:label>
>>> str(choice)
'label'
```

```
>>> choice = Choice()
>>> choice
<Choice[2]:None>
>>> str(choice)
'None'
```

**deconstruct()**

**order = 0**

**class** django\_utils.choices.Choices

Bases: object

The choices class is what you should inherit in your Django models

```
>>> choices = Choices()
>>> choices.choices[0]
Traceback (most recent call last):
...
KeyError: 'Key 0 does not exist'
>>> choices.choices
OrderedDict()
>>> str(choices.choices)
'OrderedDict()'
>>> choices.choices.items()
[]
>>> choices.choices.keys()
[]
>>> choices.choices.values()
[]
>>> list(choices)
[]
```

```
>>> class ChoiceTest(Choices):
...     a = Choice()
>>> choices = ChoiceTest()
>>> choices.choices.items()
[(0, <Choice[...]:a>)]
>>> choices.a
0
>>> choices.choices['a']
<Choice[...]:a>
>>> choices.choices[0]
<Choice[...]:a>
>>> choices.choices.keys()
[0]
>>> choices.choices.values()
['a']
>>> list(choices)
```

(continues on next page)

(continued from previous page)

```
[(0, <Choice[...]:a>)]
>>> list(ChoiceTest)
[(0, <Choice[...]:a>)]
```

```
choices = OrderedDict()
```

```
class django_utils.choices.ChoicesDict
```

```
Bases: object
```

The choices dict is an object that stores a sorted representation of the values by key and database value

```
items()
```

```
keys()
```

```
values()
```

```
class django_utils.choices.ChoicesMeta
```

```
Bases: type
```

The choices metaclass is where all the magic happens, this automatically creates a ChoicesDict to get a sorted list of keys and values

```
class django_utils.choices.LiteralChoices
```

```
Bases: django_utils.choices.Choices
```

Special version of the Choices class that uses the label as the value

```
>>> class Role(LiteralChoices):
...     admin = Choice()
...     user = Choice()
...     guest = Choice()
```

```
>>> Role.choices.values()
['admin', 'user', 'guest']
>>> Role.choices.keys()
['admin', 'user', 'guest']
```

```
choices = OrderedDict()
```

## 2.5 django\_utils.fields module

```
class django_utils.fields.RecursiveField(field_name=None, parent_field='parent', default=None)
```

```
Bases: object
```

```
PREFIX = 'get_'
```

```
contribute_to_class(cls, name)
```

```
get(instance)
```

## 2.6 django\_utils.queryset module

```
django_utils.queryset.queryset_iterator(queryset, chunksize=1000, getfunc=<built-in function getattr>)
```

“ Iterate over a Django Queryset ordered by the primary key

This method loads a maximum of chunksize (default: 1000) rows in it's memory at the same time while django normally would load all rows in it's memory. Using the iterator() method only causes it to not preload all the classes.

Note that the implementation of the iterator does not support ordered query sets.

## 2.7 django\_utils.utils module

`django_utils.utils.to_json(request, data)`

## 2.8 django\_utils.view\_decorators module

**exception** `django_utils.view_decorators.UnknownViewResponseError`

Bases: `django_utils.view_decorators.ViewError`

**exception** `django_utils.view_decorators.ViewError`

Bases: `exceptions.Exception`

`django_utils.view_decorators.env` (*function=None, login\_required=False, response\_class=<class 'django.http.response.HttpResponse'>*)

View decorator that automatically adds context and renders response

Keyword arguments: `login_required` – is everyone allowed or only authenticated users

Adds a RequestContext (`request.context`) with the following context items: `name` – current function name

Stores the template in `request.template` and assumes it to be in `<app>/<view>.html`

`django_utils.view_decorators.json_default_handler` (*obj*)

`django_utils.view_decorators.permanent_redirect` (*url, \*args, \*\*kwargs*)

`django_utils.view_decorators.redirect` (*url='./', \*args, \*\*kwargs*)

## 2.9 django\_utils.views module

`django_utils.views.error_403` (*request, \*args, \*\*kwargs*)

`django_utils.views.error_404` (*request, \*args, \*\*kwargs*)

`django_utils.views.error_500` (*request, \*args, \*\*kwargs*)

## 2.10 Module contents



## CHAPTER 3

---

### Indices and tables

---

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



### d

- `django_utils`, 15
- `django_utils.base_models`, 9
- `django_utils.choices`, 11
- `django_utils.fields`, 14
- `django_utils.management`, 6
- `django_utils.management.commands`, 6
  - `admin_autogen`, 5
  - `base_command`, 5
  - `settings`, 6
- `django_utils.queryset`, 14
- `django_utils.templatetags`, 9
  - `debug`, 6
- `django_utils.utils`, 15
- `django_utils.view_decorators`, 15
- `django_utils.views`, 15





A

abstract (*django\_utils.base\_models.CreatedAtModelBase.Meta* *django\_utils.base\_models*), 9  
     *attribute*), 9  
 abstract (*django\_utils.base\_models.ModelBase.Meta* *django\_utils.management.commands.base\_command*),  
     *attribute*), 9  
 abstract (*django\_utils.base\_models.NameCreatedAtModelBase.Meta* *django\_utils.management.commands.base\_command*),  
     *attribute*), 10  
 abstract (*django\_utils.base\_models.NameModelBase.Meta* *django\_utils.management.commands.base\_command*),  
     *attribute*), 10  
 abstract (*django\_utils.base\_models.SlugCreatedAtModelBase.Meta* *django\_utils.management.commands.base\_command*),  
     *attribute*), 10  
 abstract (*django\_utils.base\_models.SlugModelBase.Meta* *django\_utils.management.commands.base\_command*),  
     *attribute*), 11  
 add\_arguments() (*django\_utils.management.commands.settings.Command* *django\_utils.management.commands.settings.Command*), 6  
     *method*), 6

C

can\_import\_settings (*django\_utils.management.commands.settings.Command* *django\_utils.management.commands.settings.Command*), 6  
     *attribute*), 6  
 Choice (*class in django\_utils.choices*), 12  
 Choices (*class in django\_utils.choices*), 13  
 choices (*django\_utils.choices.Choices attribute*), 14  
 choices (*django\_utils.choices.LiteralChoices attribute*), 14  
 ChoicesDict (*class in django\_utils.choices*), 14  
 ChoicesMeta (*class in django\_utils.choices*), 14  
 Command (*class in django\_utils.management.commands.admin\_autogen*), 5  
     5  
 Command (*class in django\_utils.management.commands.settings*), 6  
     6  
 contribute\_to\_class() (*django\_utils.fields.RecursiveField method*), 14  
 create\_logger() (*django\_utils.management.commands.base\_command.CustomBaseCommand* *django\_utils.management.commands.base\_command*), 5  
     *method*), 5  
 created\_at (*django\_utils.base\_models.CreatedAtModelBase* *django\_utils.management.commands.base\_command*), 9  
     *attribute*), 9  
 CreatedAtModelBase (*class in django\_utils.base\_models*), 9  
     9  
 CreatedAtModelBase.Meta (*class in django\_utils.base\_models*), 9  
     9  
 CustomAppCommand (*class in django\_utils.management.commands.base\_command*), 5  
 CustomBaseCommand (*class in django\_utils.management.commands.base\_command*), 5  
 CustomCommand (*class in django\_utils.management.commands.base\_command*), 5  
 CustomMeta (*class in django\_utils.management.commands.base\_command*), 5  
 db\_table (*django\_utils.base\_models.CreatedAtModelBase.Meta* *django\_utils.management.commands.base\_command*), 9  
     *attribute*), 9  
 db\_table (*django\_utils.base\_models.ModelBase.Meta* *django\_utils.management.commands.base\_command*), 9  
     *attribute*), 9  
 db\_table (*django\_utils.base\_models.NameCreatedAtModelBase.Meta* *django\_utils.management.commands.base\_command*), 10  
     *attribute*), 10  
 db\_table (*django\_utils.base\_models.NameModelBase.Meta* *django\_utils.management.commands.base\_command*), 10  
     *attribute*), 10  
 db\_table (*django\_utils.base\_models.SlugCreatedAtModelBase.Meta* *django\_utils.management.commands.base\_command*), 10  
     *attribute*), 10  
 db\_table (*django\_utils.base\_models.SlugModelBase.Meta* *django\_utils.management.commands.base\_command*), 11  
     *attribute*), 11  
 debug() (*in module django\_utils.templatetags.debug*), 9  
 deconstruct() (*django\_utils.choices.Choice* *django\_utils.choices.Choice*), 13  
     *method*), 13  
 django\_utils (*module*), 15  
 django\_utils.base\_models (*module*), 9  
 django\_utils.choices (*module*), 11  
 django\_utils.fields (*module*), 14  
 django\_utils.management (*module*), 6  
 django\_utils.management.commands (*module*), 6  
     6  
 django\_utils.management.commands.admin\_autogen (*module*), 5  
 django\_utils.management.commands.base\_command (*module*), 5  
 django\_utils.management.commands.settings (*module*), 6  
 django\_utils.queryset (*module*), 14

django\_utils.templatetags (*module*), 9  
 django\_utils.templatetags.debug (*module*),  
 6  
 django\_utils.utils (*module*), 15  
 django\_utils.view\_decorators (*module*), 15  
 django\_utils.views (*module*), 15

## E

env () (*in module django\_utils.view\_decorators*), 15  
 error\_403 () (*in module django\_utils.views*), 15  
 error\_404 () (*in module django\_utils.views*), 15  
 error\_500 () (*in module django\_utils.views*), 15

## F

format () (*django\_utils.templatetags.debug.Formatter*  
*method*), 6  
 format\_datetime ()  
 (*django\_utils.templatetags.debug.Formatter*  
*method*), 6  
 format\_dict () (*django\_utils.templatetags.debug.Formatter*  
*method*), 7  
 format\_int () (*django\_utils.templatetags.debug.Formatter*  
*method*), 7  
 format\_list () (*django\_utils.templatetags.debug.Formatter*  
*method*), 7  
 format\_model () (*django\_utils.templatetags.debug.Formatter*  
*method*), 7  
 format\_object () (*django\_utils.templatetags.debug.Formatter*  
*method*), 8  
 format\_str () (*django\_utils.templatetags.debug.Formatter*  
*method*), 8  
 format\_unicode () (*django\_utils.templatetags.debug.Formatter*  
*method*), 8  
 Formatter (*class in django\_utils.templatetags.debug*),  
 6

## G

get () (*django\_utils.fields.RecursiveField method*), 14  
 get\_next\_by\_created\_at ()  
 (*django\_utils.base\_models.CreatedAtModelBase*  
*method*), 9  
 get\_next\_by\_created\_at ()  
 (*django\_utils.base\_models.NameCreatedAtModelBase*  
*method*), 10  
 get\_next\_by\_created\_at ()  
 (*django\_utils.base\_models.SlugCreatedAtModelBase*  
*method*), 11  
 get\_next\_by\_updated\_at ()  
 (*django\_utils.base\_models.CreatedAtModelBase*  
*method*), 9  
 get\_next\_by\_updated\_at ()  
 (*django\_utils.base\_models.NameCreatedAtModelBase*  
*method*), 10

get\_next\_by\_updated\_at ()  
 (*django\_utils.base\_models.SlugCreatedAtModelBase*  
*method*), 11  
 get\_previous\_by\_created\_at ()  
 (*django\_utils.base\_models.CreatedAtModelBase*  
*method*), 9  
 get\_previous\_by\_created\_at ()  
 (*django\_utils.base\_models.NameCreatedAtModelBase*  
*method*), 10  
 get\_previous\_by\_created\_at ()  
 (*django\_utils.base\_models.SlugCreatedAtModelBase*  
*method*), 11  
 get\_previous\_by\_updated\_at ()  
 (*django\_utils.base\_models.CreatedAtModelBase*  
*method*), 9  
 get\_previous\_by\_updated\_at ()  
 (*django\_utils.base\_models.NameCreatedAtModelBase*  
*method*), 10  
 get\_previous\_by\_updated\_at ()  
 (*django\_utils.base\_models.SlugCreatedAtModelBase*  
*method*), 11

## H

handle () (*django\_utils.management.commands.admin\_autogen.Command*  
*method*), 5  
 handle () (*django\_utils.management.commands.base\_command.CustomBase*  
*method*), 5  
 handle () (*django\_utils.management.commands.settings.Command*  
*method*), 6  
 help (*django\_utils.management.commands.settings.Command*  
*attribute*), 6

## I

items () (*django\_utils.choices.ChoicesDict method*),  
 14

## J

json\_default () (*in module*  
*django\_utils.management.commands.settings*),  
 6  
 json\_default\_handler () (*in module*  
*django\_utils.view\_decorators*), 15

## K

keys () (*django\_utils.choices.ChoicesDict method*), 14

## L

LiteralChoices (*class in django\_utils.choices*), 14  
 loggers (*django\_utils.management.commands.base\_command.CustomBase*  
*attribute*), 5

## M

MAX\_LENGTH (*django\_utils.templatetags.debug.Formatter*  
*attribute*), 6

- MAX\_LENGTH\_DOTS (*django\_utils.templatetags.debug.ForStatus* attribute), 6
- ModelBase (class in *django\_utils.base\_models*), 9
- ModelBase.Meta (class in *django\_utils.base\_models*), 9
- ModelBaseMeta (class in *django\_utils.base\_models*), 9
- N**
- name (*django\_utils.base\_models.NameModelBase* attribute), 10
- NameCreatedAtModelBase (class in *django\_utils.base\_models*), 10
- NameCreatedAtModelBase.Meta (class in *django\_utils.base\_models*), 10
- NameMixin (class in *django\_utils.base\_models*), 10
- NameModelBase (class in *django\_utils.base\_models*), 10
- NameModelBase.Meta (class in *django\_utils.base\_models*), 10
- O**
- order (*django\_utils.choices.Choice* attribute), 13
- output\_types (*django\_utils.management.commands.settings.Command* attribute), 6
- P**
- permanent\_redirect () (in module *django\_utils.view\_decorators*), 15
- PREFIX (*django\_utils.fields.RecursiveField* attribute), 14
- Q**
- queryset\_iterator () (in module *django\_utils.queryset*), 14
- R**
- RecursiveField (class in *django\_utils.fields*), 14
- redirect () (in module *django\_utils.view\_decorators*), 15
- render\_output () (*django\_utils.management.commands.settings.Command* method), 6
- requires\_model\_validation (*django\_utils.management.commands.settings.Command* attribute), 6
- S**
- save () (*django\_utils.base\_models.SlugMixin* method), 11
- slug (*django\_utils.base\_models.SlugModelBase* attribute), 11
- SlugCreatedAtModelBase (class in *django\_utils.base\_models*), 10
- SlugMixin (class in *django\_utils.base\_models*), 11
- SlugMixin.Meta (class in *django\_utils.base\_models*), 11
- SlugModelBase (class in *django\_utils.base\_models*), 11
- SlugModelBase.Meta (class in *django\_utils.base\_models*), 11
- T**
- to\_json () (in module *django\_utils.utils*), 15
- U**
- unique\_together (*django\_utils.base\_models.SlugMixin.Meta* attribute), 11
- UnknownViewResponseError, 15
- updated\_at (*django\_utils.base\_models.CreatedAtModelBase* attribute), 9
- V**
- values () (*django\_utils.choices.ChoicesDict* method), 14
- VIEW\_ERROR, 15