django-countries-flavor Documentation

Release 0.1.2

mongkok

Feb 18, 2018

Contents

A Django application that provides a data collection for internationalization and localization purposes.

CHAPTER 1

Contents

1.1 Setup

1.1.1 Dependencies

`django-countries-flavor` supports Django 1.9+ on Python 3.4, 3.5, 3.6 and 3.7.

> **Warning:** PostGIS database (PostgreSQL 9.4) is required.

1.1.2 Installation

Install last stable version from pypi.

```
pip install django-countries-flavor
```

Add `countries` to your *INSTALLED_APPS* settings:

```
INSTALLED_APPS = [
    ...
    'countries.apps.CountriesAppConfig',
]
```

Apply **migrations**:

```
python manage.py migrate
```

1.1.3 Load data

The `loadcountries` management command read all **fixtures** and re-loaded into the database:

```
python manage.py loadcountries
```

1.2 Countries

For example, we could look up the Country:

```
>>> country = Country.objects.get(cca2='ID')
>>> country.timezones.all()
<QuerySet [<Timezone: Asia/Jakarta>, <Timezone: Asia/Jayapura>, <Timezone: Asia/
↪Makassar>, <Timezone: Asia/Pontianak>]>
```

1.2.1 Borders

```
>>> country.borders.all()
<QuerySet [<Country: MY>, <Country: PG>, <Country: TL>]>
```

1.2.2 Geometry Lookups

Geographic queries with `Country` take the following general form:

```
>>> qs = Country.objects.filter(mpoly__<lookup_type>=<parameter>)
>>> qs = Country.objects.exclude(...)
```

For example:

```
>>> from django.contrib.gis.geos import Point
>>> point = Point(120.0, -5.0)
>>> Country.objects.filter(mpoly__contains=point)
<QuerySet [<Country: ID>]>
```

1.2.3 Distance

```
>>> from django.contrib.gis.measure import D
>>> Country.objects.filter(location__distance_lte=(point, D(km=2000)))
<QuerySet [<Country: BN>, <Country: CX>, <Country: ID>, <Country: MY>, <Country: SG>,
↪<Country: TL>]>
```

1.3 Locales

Retrieve locales by country:

```
>>> country = Country.objects.get(cca2='ID')
>>> country.locales.all()
<LocaleQuerySet [<Locale: id_ID>]>
```

```
>>> locale = country.locales.get(code='id_ID')
```

1.3.1 Properties

```
>>> locale.week_data
{'first_day': 6, 'min_days': 1, 'weekend_end': 6, 'weekend_start': 5}
```

```
>>> locale.number_symbols
{'alias': 'None',
 'decimal': ',',
 'exponential': 'E',
 'group': '.',
 'infinity': '∞',
 'list': ';',
 'minusSign': '-',
 'nan': 'NaN',
 'perMille': '‰',
 'percentSign': '%',
 'plusSign': '+',
 'superscriptingExponent': '×',
 'timeSeparator': '.'}
```

```
>>> locale.months['format']['wide']
{'1': 'Januari',
 '10': 'Oktober',
 '11': 'November',
 '12': 'Desember',
 '2': 'Februari',
 '3': 'Maret',
 '4': 'April',
 '5': 'Mei',
 '6': 'Juni',
 '7': 'Juli',
 '8': 'Agustus',
 '9': 'September'}
```

```
>>> locale.days['format']['wide']
{'0': 'Senin',
 '1': 'Selasa',
 '2': 'Rabu',
 '3': 'Kamis',
 '4': 'Jumat',
 '5': 'Sabtu',
 '6': 'Minggu'}
```

1.4 Timezones

Retrieve timezones by country:

```
>>> country = Country.objects.get(cca2='ID')
>>> country.timezones.all()
<QuerySet [<Timezone: Asia/Jakarta>, <Timezone: Asia/Jayapura>, <Timezone: Asia/
↪Makassar>, <Timezone: Asia/Pontianak>]>
```

```
>>> timezone = country.timezones.get(name='Asia/Makassar')
```

Use the use the `localize()` method to localize a naive datetime (datetime with no timezone information):

```
>>> from datetime import datetime
>>> dtime = datetime(year=2017, month=4, day=2, hour=16, minute=20)
>>> timezone.localize(dtime)
datetime.datetime(2017, 4, 2, 16, 20, tzinfo=<DstTzInfo 'Asia/Makassar' WITA+8:00:00␣
↪STD>)
```

Converting an existing localized time using the standard ‘‘astimezone()‘:

```
>>> utc_dtime = dtime.replace(tzinfo=pytz.utc)
>>> timezone.astimezone(utc_dtime)
datetime.datetime(2017, 4, 3, 0, 20, tzinfo=<DstTzInfo 'Asia/Makassar' WITA+8:00:00␣
↪STD>)
```

Get the current time with `now()`:

```
>>> from datetime import datetime
>>> timezone = country.timezones.get(name='Asia/Makassar')
>>> timezone.now()
datetime.datetime(2017, 5, 2, 16, 15, 13, 626913, tzinfo=<DstTzInfo 'Asia/Makassar'␣
↪WITA+8:00:00 STD>)
```

set the current time zone to the end user’s actual time zone with `activate()`

```
>>> timezone.activate()
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search