
Django Timezone Utils Documentation

Release 0.7

Michael Barr

February 27, 2015

1	Contents	3
1.1	Setup	3
1.2	Fields	3
1.3	Choices	7
2	Contributing	11
3	Indices and tables	13

Adds automatic time zone conversions and support utilities to Django.

Contents

1.1 Setup

1.1.1 Installation

Install from PyPi with pip:

```
pip install django-timezone-utils
```

To use `django-timezone-utils` in your Django project, just import and use the utility classes described in this documentation; there is no need to modify your `INSTALLED_APPS` setting.

1.1.2 Dependencies

- Supports [Django 1.4+](#) on Python 2.6, 2.7, 3.2, 3.3, and 3.4.
- Requires [pytz](#) for time zone conversions.

1.1.3 Deprecation Policy

`django-timezone-utils` will support any version of Django that is currently supported by the Django Software Foundation. Please view [Django's Supported Version documentation](#) for more information.

1.2 Fields

Contains timezone-related fields.

1.2.1 TimeZoneField

class `TimeZoneField(*args, **kwargs)`

A `models.CharField` subclass that stores a valid Olson Timezone string found in `pytz.all_timezones` to the database.

Raises `django.core.exceptions.ValidationError` if the value is not a valid Olson Time zone string.

Returns A `datetime.tzinfo` object based on the value or `None`.

Return type `datetime.tzinfo`

```
from timezone_utils.fields import TimeZoneField
from timezone_utils.choices import PRETTY_ALL_TIMEZONES_CHOICES

class Location(models.Model):
    # ...
    timezone = TimeZoneField(choices=PRETTY_ALL_TIMEZONES_CHOICES)
```

You can pass a default Olson timezone value to the model as a string:

```
from timezone_utils.fields import TimeZoneField

class Location(models.Model):
    # ...
    timezone = TimeZoneField(default='US/Pacific')
```

You can also pass a default Olson timezone value to the model as a `pytz.timezone` instance:

```
import pytz
from timezone_utils.fields import TimeZoneField

class Location(models.Model):
    # ...
    timezone = TimeZoneField(default=pytz.timezone('US/Eastern'))
```

Accessing a `TimeZoneField` on a model

As a convenience, `TimeZoneField` automatically converts its model instance attribute representation to a `pytz.timezone` instance:

```
>>> from datetime import datetime
>>> from django.utils.timezone import make_aware
>>> from app_label.models import Location
>>> location = Location.objects.create(timezone='US/Eastern')
>>> print(location.timezone)
<DstTzInfo 'US/Eastern' LMT-1 day, 19:04:00 STD>
>>> unaware_dt = datetime(2015, 1, 1)
>>> aware_dt = make_aware(unaware_dt, location.timezone)
>>> print(aware_dt)
datetime.datetime(2015, 1, 1, 0, 0, tzinfo=<DstTzInfo 'US/Eastern' EST-1 day, 19:00:00 STD>)
```

1.2.2 `LinkedTZDateTimeField`

class `LinkedTZDateTimeField` (*populate_from=None, time_override=None, *args, **kwargs*)

A `models.DateTimeField` subclass that will automatically save a datetime object as a particular time zone as declared in the kwarg `populate_from`. You can also override the time each time the object is saved with the kwarg `time_override`, which must be a `datetime.time` instance.

Parameters

- **`populate_from`** – The location of the field which contains the time zone that will be used for this field. Must be either a function which returns a `models.ForeignKey` path to the timezone or a string (if the timezone field is located on the model).
- **`time_override`** – Automatically overrides the time value each time the object is saved to the time that is declared. Must be a `datetime.time` instance.

Raises

- **AttributeError** – if the `populate_from` parameter is invalid.
- **ValueError** – if the `time_override` is not a `datetime.time` instance.
- **pytz.UnknownTimeZoneError** – if the parsed model instance value of `populate_from` is not a valid Olson time zone string.

Returns A `datetime.datetime` object based on the time zone declared in `populate_from` and override from `time_override` or `None`.

Return type `datetime.datetime`

Note: If `auto_now` or `auto_now_add` is declared, the value of `time_override` is ignored.

Caution: Django cannot serialize lambdas! If you provide a lambda for `populate_from`, your model will fail to migrate in Django 1.7+.

```
from datetime import datetime
from timezone_utils.fields import LinkedTZDateTimeField, TimeZoneField
from timezone_utils.choices import PRETTY_ALL_TIMEZONES_CHOICES

class Location(models.Model):
    # ...
    timezone = TimeZoneField(choices=PRETTY_ALL_TIMEZONES_CHOICES)

def get_location_timezone(obj):
    return obj.location.timezone

class LocationReport(models.Model):
    # ...
    location = models.ForeignKey('app_label.Location', related_name='reports')
    timestamp = LinkedTZDateTimeField(populate_from=get_location_timezone)

class LocationPeriod(models.Model):
    # ...
    location = models.ForeignKey('app_label.Location', related_name='reports')
    start = LinkedTZDateTimeField(
        populate_from=get_location_timezone,
        time_override=datetime.min.time()
    )
    end = LinkedTZDateTimeField(
        populate_from=get_location_timezone,
        time_override=datetime.max.time()
    )
```

Accessing a LinkedTZDateTimeField on a model

As a convenience, `LinkedTZDateTimeField` automatically converts its model instance attribute representation to the time zone declared in the `populate_from`, regardless of what `settings.TIME_ZONE` is set to:

```
>>> from datetime import datetime
>>> from app_label.models import LocationPeriod
```

```
>>> location = location.objects.get(pk=1)
>>> location_period = LocationPeriod.objects.create(location=location, start=datetime(2015, 1, 1), end=datetime(2015, 1, 1), tzinfo=DstTzInfo('US/Eastern', EST-1 day, 19:00:00 STD))
>>> print(location_period.start)
datetime.datetime(2015, 1, 1, 0, 0, tzinfo=DstTzInfo('US/Eastern', EST-1 day, 19:00:00 STD))
>>> print(location_period.end)
datetime.datetime(2015, 12, 31, 23, 59, 59, 999999, tzinfo=DstTzInfo('US/Eastern', EST-1 day, 19:00:00 STD))
```

Accessing a LinkedTZDateTimeField in templates

Django templates will automatically cast the timezones to the currently-activated time zone for the user viewing the page. An example of how to display our fields with the appropriate time zone would look something like this:

app_label/views.py:

```
# Django
from django.views.generic import ListView

# App
from app_label.models import LocationPeriod

class LocationReportingPeriodListView(ListView):
    model = LocationPeriod
    template_name = 'app_label/location_period_list.html'

    def get_queryset(self):
        """Retrieve the queryset and perform select_related on 'location' since
        we will be using it in the template.

        """
        return super(
            LocationReportingPeriodListView,
            self
        ).get_queryset().select_related(
            'location'
        )
```

templates/app_label/location_period_list.html:

```
{% load tz %}
{% load i18n %}

{% block content %}
    <table>
        <thead>
            <tr>
                <th>{% trans "Location" %}</th>
                <th>{% trans "Start" %}</th>
                <th>{% trans "End" %}</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>{% for period in object_list %}
                    {# Activate the timezone for each location #}
                    {% timezone period.location.timezone %}
                        <td>{{ period.location.name }}</td>
                        <td>{{ period.start }}</td>
                    </td>
                </tr>
            </tbody>
        </table>
{% endblock %}
```

```
                <td>{{ period.end }}</td>
            </tr>
            {% endtimezone %}
        {% empty %}
        <tr>
            <td colspan=3>{% trans "No periods to display." %}</td>
        </tr>
        {% endfor %}
    </tdata>
</table>
{% endblock content %}
```

1.3 Choices

Contains constants and functions to generate model/form choices for time zones.

1.3.1 ALL_TIMEZONES_CHOICES

ALL_TIMEZONE_CHOICES

Returns choices directly populated from `pytz.all_timezones`.

```
>>> from timezone_utils.choices import ALL_TIMEZONES_CHOICES
>>> print ALL_TIMEZONES_CHOICES
(
    ('Africa/Abidjan', 'Africa/Abidjan'),
    ('Africa/Accra', 'Africa/Accra'),
    ('Africa/Addis_Ababa', 'Africa/Addis_Ababa'),
    ('Africa/Algiers', 'Africa/Algiers'),
    ('Africa/Asmara', 'Africa/Asmara'),
    ('Africa/Asmera', 'Africa/Asmera'),
    ('Africa/Bamako', 'Africa/Bamako'),
    ('Africa/Bangui', 'Africa/Bangui'),
    ('Africa/Banjul', 'Africa/Banjul'),
    ('Africa/Bissau', 'Africa/Bissau'),
    ...
)
```

1.3.2 COMMON_TIMEZONES_CHOICES

COMMON_TIMEZONE_CHOICES

Returns choices directly populated from `pytz.common_timezones`.

```
>>> from timezone_utils.choices import COMMON_TIMEZONES_CHOICES
>>> print COMMON_TIMEZONES_CHOICES
(
    ('Africa/Abidjan', 'Africa/Abidjan'),
    ('Africa/Accra', 'Africa/Accra'),
    ('Africa/Addis_Ababa', 'Africa/Addis_Ababa'),
    ('Africa/Algiers', 'Africa/Algiers'),
    ('Africa/Asmara', 'Africa/Asmara'),
    ('Africa/Bamako', 'Africa/Bamako'),
    ('Africa/Bangui', 'Africa/Bangui'),
    ('Africa/Banjul', 'Africa/Banjul'),
    ...
)
```

```
    ('Africa/Bissau', 'Africa/Bissau'),
    ('Africa/Blantyre', 'Africa/Blantyre'),
    ...
)
```

1.3.3 PRETTY_ALL_TIMEZONES_CHOICES

PRETTY_ALL_TIMEZONES_CHOICES

Returns choices formatted for display ordered by their timezone offsets populated from `pytz.all_timezones`.

```
>>> from timezone_utils.choices import PRETTY_ALL_TIMEZONES_CHOICES
>>> print PRETTY_ALL_TIMEZONES_CHOICES
(
    ('Etc/GMT+12', ' (GMT-12:00) Etc/GMT+12'),
    ('Etc/GMT+11', ' (GMT-11:00) Etc/GMT+11'),
    ('Pacific/Midway', ' (GMT-11:00) Pacific/Midway'),
    ('Pacific/Niue', ' (GMT-11:00) Pacific/Niue'),
    ('Pacific/Pago_Pago', ' (GMT-11:00) Pacific/Pago_Pago'),
    ('Pacific/Samoa', ' (GMT-11:00) Pacific/Samoa'),
    ('US/Samoa', ' (GMT-11:00) US/Samoa'),
    ('America/Adak', ' (GMT-10:00) America/Adak'),
    ('America/Atka', ' (GMT-10:00) America/Atka'),
    ('Etc/GMT+10', ' (GMT-10:00) Etc/GMT+10'),
    ...
)
```

1.3.4 PRETTY_COMMON_TIMEZONES_CHOICES

PRETTY_COMMON_TIMEZONES_CHOICES

Returns choices formatted for display ordered by their timezone offsets populated from `pytz.common_timezones`.

```
>>> from timezone_utils.choices import PRETTY_COMMON_TIMEZONES_CHOICES
>>> print PRETTY_COMMON_TIMEZONES_CHOICES
(
    ('Pacific/Midway', ' (GMT-11:00) Pacific/Midway'),
    ('Pacific/Niue', ' (GMT-11:00) Pacific/Niue'),
    ('Pacific/Pago_Pago', ' (GMT-11:00) Pacific/Pago_Pago'),
    ('America/Adak', ' (GMT-10:00) America/Adak'),
    ('Pacific/Honolulu', ' (GMT-10:00) Pacific/Honolulu'),
    ('Pacific/Johnston', ' (GMT-10:00) Pacific/Johnston'),
    ('Pacific/Rarotonga', ' (GMT-10:00) Pacific/Rarotonga'),
    ('Pacific/Tahiti', ' (GMT-10:00) Pacific/Tahiti'),
    ('US/Hawaii', ' (GMT-10:00) US/Hawaii'),
    ('Pacific/Marquesas', ' (GMT-09:30) Pacific/Marquesas'),
    ...
)
```

1.3.5 get_choices(timezones)

get_choices(*timezones*)

Retrieves timezone choices from any iterable (normally from `pytz`).

Parameters `timezones` (*iterable*) – Any iterable that contains valid Olson Time Zone strings.

Returns A tuple containing tuples of time zone choices.

Return type tuple

Raises

- `pytz.exceptions.UnknownTimeZoneError` – if the string from the iterable `timezones` parameter is not recognized as a valid Olson time zone.
- `TypeError` – if the `timezones` parameter is not iterable.

Using `get_choices(timezones)` for custom time zone choices

If you want to limit choices to a particular country (as an example), you could do this:

```
>>> import pytz
>>> from timezone_utils.choices import get_choices
>>> choices = get_choices(pytz.country_timezones('US'))
>>> print choices
(
    (u'America/Adak', ' (GMT-10:00) America/Adak'),
    (u'Pacific/Honolulu', ' (GMT-10:00) Pacific/Honolulu'),
    (u'America/Anchorage', ' (GMT-09:00) America/Anchorage'),
    (u'America/Juneau', ' (GMT-09:00) America/Juneau'),
    (u'America/Nome', ' (GMT-09:00) America/Nome'),
    (u'America/Sitka', ' (GMT-09:00) America/Sitka'),
    (u'America/Yakutat', ' (GMT-09:00) America/Yakutat'),
    (u'America/Los_Angeles', ' (GMT-08:00) America/Los_Angeles'),
    (u'America/Metlakatla', ' (GMT-08:00) America/Metlakatla'),
    (u'America/Boise', ' (GMT-07:00) America/Boise'),
    (u'America/Denver', ' (GMT-07:00) America/Denver'),
    ...
)
```

Contributing

Please file bugs and send pull requests to the [GitHub repository](#) and [issue tracker](#).

Indices and tables

- *genindex*

A

ALL_TIMEZONE_CHOICES (built-in variable), [7](#)

C

COMMON_TIMEZONE_CHOICES (built-in variable),
[7](#)

G

get_choices() (built-in function), [8](#)

L

LinkedTZDateTimeField (built-in class), [4](#)

P

PRETTY_ALL_TIMEZONES_CHOICES (built-in variable), [8](#)

PRETTY_COMMON_TIMEZONES_CHOICES (built-in variable), [8](#)

T

TimeZoneField (built-in class), [3](#)