django-osm-field

Release 0.3.1

February 06, 2017

Contents

Contents:

CHAPTER 1

Installation

Install **django-osm-field** into your virtual environment or you site-packages using pip:

```
$ pip install django-osm-field
```

To make **django-osm-field** available in your Django project, you first have to add it to the INSTALLED_APPS in your settings.py. If you are unsure where to put it, just append it:

```
INSTALLED_APPS = (
    ...
    'osm_field',
    ...
)
```

CHAPTER 2

Usage

2.1 Model Layer

You need to add three model fields to your model:

1. *OSMField*
2. *LatitudeField*
3. *LongitudeField*

django-osm-field expects them to have a certain name schema: The *OSMField* defines the base name, the *LatitudeField* and *LongitudeField* have the same name appended with `_lat` and `_lon` respectively. See the following example to get an idea:

```
from django.db import models

from osm_field.fields import LatitudeField, LongitudeField, OSMField


class MyModel(models.Model):
    location = OSMField()
    location_lat = LatitudeField()
    location_lon = LongitudeField()
```

It is possible, though, to overwrite the default naming for latitude and longitude fields by giving their names as arguments to the *OSMField*:

```
class MyModel(models.Model):
    location = OSMField(lat_field='latitude', lon_field='longitude')
    latitude = LatitudeField()
    longitude = LongitudeField()
```

2.2 Form Layer

```
from django import forms

from .models import MyModel


class MyModelForm(forms.ModelForm):
```

```
    class Meta:
        fields = ('location', 'location_lat', 'location_lon', )
        model = MyModel
```

2.3 Formset Layer

To use OSMField with formsets with Django < 1.9, you must mixin the `OSMFormMixin` to your child form class:

`models.py`:

```
from django.db import models

from osm_field.fields import LatitudeField, LongitudeField, OSMField


class ParentModel(models.Model):
    name = models.CharField(max_length=31)


class ChildModel(models.Model):
    parent = models.ForeignKey(ParentModel, related_name='children')
    location = OSMField()
    location_lat = LatitudeField()
    location_lon = LongitudeField()
```

`forms.py`:

```
from django import forms

from osm_field.forms import OSMFormMixin

from .models import ChildModel, ParentModel


class ChildModelInlineForm(OSMFormMixin, forms.ModelForm):
    class Meta:
        fields = ('location', 'location_lat', 'location_lon', )
        model = ChildModel

ChildModelFormset = forms.models.inlineformset_factory(
    ParentModel, ChildModel, form=ChildModelInlineForm
)
```

Note that you ONLY need to do this for Django < 1.9, but this will still work without modification (but is unnecessary) for Django >= 1.9.

2.4 View Layer

```
from django.views.generic import CreateView

from .forms import MyModelForm
from .models import MyModel
```

```
class MyCreateView(CreateView):
    form_class = MyModelForm
    model = MyModel
```

2.5 Template Layer

django-osm-field shipps with a minimized jQuery version. To access it in a template use the `static` templatetag from the `staticfiles` Django app:

```
<script type="text/javascript" src="{% static "js/vendor/jquery-2.1.0.min.js" %}"></script>
```

You can of course load jQuery from a CDN as well:

```
<script type="text/javascript" src="//code.jquery.com/jquery-2.1.0.min.js"></script>
```

To get the front-end to work, you also need to include some CSS and JavaScript files. You can do this by simply using `{{ form.media }}` or by adding those lines explicitly:

```
<link href="{% static "css/vendor/leaflet.css" %}" type="text/css" media="screen" rel="stylesheet" />
<link href="{% static "css/osm_field.css" %}" type="text/css" media="screen" rel="stylesheet" />
<script type="text/javascript" src="{% static "js/vendor/leaflet.js" %}"></script>
<script type="text/javascript" src="{% static "js/osm_field.js" %}"></script>
```

In the end your template should look similar to this:

```
{% load static from staticfiles %}<!DOCTYPE HTML>
<html>
  <head>
    <title></title>
    <link rel="stylesheet" href="{% static "css/example.css" %}">
    <!-- Either serve jQuery yourself -->
    <script type="text/javascript" src="{% static "js/vendor/jquery-2.1.0.min.js" %}"></script>
    <!-- or from a CDN -->
    <script type="text/javascript" src="//code.jquery.com/jquery-2.1.0.min.js"></script>
  </head>
  <body>
    {{ form.media }}
    <form action="" method="post">
      {% csrf_token %}
      {{ form.as_p }}
      <input type="submit" value="Save" />
    </form>
  </body>
</html>
```

Project has been started by Sinnwerkstatt Medienagentur GmbH in April 2014.

CHAPTER 3

History

3.1 0.3.1 (2017-02-06)

- Properly rendered widgets in formsets
- Added preliminary support for Django 1.11

3.2 0.3.0 (2016-07-23)

- Added support for Django 1.8, 1.9, 1.10
- Dropped support for Django 1.4, 1.5, 1.6
- Switched from MapQuest to CartoDB (#15)
- Fixed bug in formsets (#7)
- Added `location_data` field

3.3 0.2.0 (2014-11-10)

- Added support for Django 1.7 migrations (#2)
- Updated styling (#1)
- Forced map refresh on show
- Added tests
- Changed license from BSD to MIT
- Added support for non-default named form fields (those not ending with `_lat` and `_lon` respectively).
- Added documentation

3.4 0.1.4 (2014-06-02)

- Add minified JavaScript and CSS sources

3.5 0.1.3 (2014-05-28)

- jQuery is not automatically added by the widgets media class anymore

3.6 0.1.0 (2014-05-20)

- First release on PyPI.

CHAPTER 4

References

4.1 Fields

4.1.1 OSMField

class osm_field.fields.**OSMField**(**args*, ***kwargs*)
Bases: django.db.models.TextField

Parameters

- **lat_field** (*str*) – The name of the latitude field. None (and thus standard behavior) by default.
- **lon_field** (*str*) – The name of the longitude field. None (and thus standard behavior) by default.

All default field options.

formfield(***kwargs*)

Returns A *OSMFormField* with a *OSMWidget*.

latitude_field_name
The name of the related *LatitudeField*.

longitude_field_name
The name of the related *LongitudeField*.

4.1.2 LatitudeField

class osm_field.fields.**LatitudeField**(**args*, ***kwargs*)
Bases: django.db.models.FloatField

All default field options.

The validators parameter will be appended with *validate_latitude()* if not already present.

formfield(***kwargs*)

Returns A FloatField with max_value 90 and min_value -90.

4.1.3 LongitudeField

class osm_field.fields.**LongitudeField**(**args*, ***kwargs*)
Bases: django.db.models.FloatField

All default field options.

The validators parameter will be appended with *validate_longitude()* if not already present.

formfield(***kwargs*)

Returns A FloatField with max_value 180 and min_value -180.

4.1.4 Utilities

Location

class osm_field.fields.**Location**(*lat*, *lon*, *text*)
A wrapper class bundling the description of a location (text) and its geo coordinates, latitude (lat) and longitude (lon).

Parameters

- **lat** (*float*) – The latitude
- **lon** (*float*) – The longitude
- **str** – The description

4.2 Forms

4.2.1 OSMBoundField

4.2.2 OSMFormField

class osm_field.forms.**OSMFormField**(*max_length=None*, *min_length=None*, *strip=True*, **args*, ***kwargs*)
Bases: osm_field.forms.PrefixedFormFieldMixin, django.forms.fields.CharField

4.3 Validators

4.3.1 validate_latitude

osm_field.fields.**validate_latitude**(*value*)
Validates that the given value does not exceed the range [-90, 90].

Raises Raises a ValidationError if value is not within the range.

4.3.2 validate_longitude

osm_field.fields.**validate_longitude**(*value*)
Validates that the given value does not exceed the range [-180, 180].

Raises Raises a `ValidationError` if `value` is not within the range.

4.4 Widgets

4.4.1 OSMWidget

class osm_field.widgets.**OSMWidget**(*lat_field*, *lon_field*, *data_field=None*, *attrs=None*)
Bases: `django.forms.widgets.TextInput`

Adds a OpenStreetMap Leaflet dropdown map to the front-end once the user focuses the form field. See *the usage chapter* on how to integrate the CSS and JavaScript code.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

f

o

w

F

L

O

V