
django-location-field Documentation

Release 2.0.0

Caio Ariede

May 30, 2019

Contents

1	Contents	3
1.1	Getting started	3
1.2	Settings	4
1.3	Form fields	5
1.4	Changelog	6
2	Indices and tables	7

The **django-location-field** app provides model and form fields, and widgets that allows your users to easily pick locations and store in the database.

1.1 Getting started

1.1.1 Installation

Using pip:

```
pip install django-location-field
```

1.1.2 Configuration

Add `location_field.apps.DefaultConfig` to `INSTALLED_APPS`.

Example:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.sites',  
  
    'location_field.apps.DefaultConfig',  
]
```

Choose a map provider

At this moment we support three map providers:

- Google
- Mapbox

- [OpenStreetMap](#)

Choose a search provider

At this moment we support three search providers:

- [Google](#)
- [Yandex](#)
- [Nominatim](#)

1.1.3 Rendering form with a map

NOTE: If you will be using the map widget only inside Django admin, you can skip this section as it should work out-of-the-box.

The way to render a form containing a map is the same as any other forms containing their own javascript and stylesheets, except that you need to ensure your page loads jQuery.

```
<head>
  <script src="[ link to jquery.js goes here ]"></script>
  {{form.media}}
</head>
<body>
  {{form}}
</body>
```

You can find a running example in the example app (example/ directory).

1.2 Settings

1.2.1 Default settings

These are the default settings:

```
LOCATION_FIELD_PATH = settings.STATIC_URL + 'location_field'

LOCATION_FIELD = {
    'map.provider': 'google',
    'map.zoom': 13,

    'search.provider': 'google',
    'search.suffix': '',

    # Google
    'provider.google.api': '//maps.google.com/maps/api/js?sensor=false',
    'provider.google.api_key': '',
    'provider.google.api_libraries': '',
    'provider.google.map.type': 'ROADMAP',

    # Mapbox
    'provider.mapbox.access_token': '',
    'provider.mapbox.max_zoom': 18,
```

(continues on next page)

(continued from previous page)

```

'provider.mapbox.id': 'mapbox.streets',

# OpenStreetMap
'provider.openstreetmap.max_zoom': 18,

# misc
'resources.root_path': LOCATION_FIELD_PATH,
'resources.media': {
    'js': (
        LOCATION_FIELD_PATH + '/js/jquery.livequery.js',
        LOCATION_FIELD_PATH + '/js/form.js',
    ),
},
}

```

1.2.2 Override default settings

To override the default settings, you can define only what you want to override, for example:

```

LOCATION_FIELD = {
    'map.provider': 'openstreetmap',
    'search.provider': 'nominatim',
}

```

This will keep all other settings the same.

1.3 Form fields

There are two type of form fields, the `PlainLocationField` *for non-spatial databases* and the `LocationField` *for spatial databases* like PostGIS and SpatiaLite.

The attributes that can be passed to both form fields are:

Attribute	Description
<code>based_fields</code>	A list of fields that will be used to populate the location field
<code>zoom</code>	The default zoom level for the map
<code>suffix</code>	A suffix that will be added to the search string, like a city name. Useful when you want to restrict the search to determined areas.

1.3.1 For non-spatial databases

For non-spatial databases you may want to use the `PlainLocationField`, which stores the latitude and longitude values as plain text.

Example:

```

from django import forms
from location_field.forms.plain import PlainLocationField

class Address(forms.Form):
    city = forms.CharField()

```

(continues on next page)

(continued from previous page)

```
location = PlainLocationField(based_fields=['city'],
                              initial='-22.2876834,-49.1607606')
```

1.3.2 For spatial databases

For spatial databases like `PostGIS` and `SpatiaLite` you may want to use the `LocationField`, which stores the latitude and longitude values as a `Point` object.

Example:

```
from django import forms
from django.contrib.gis.geos import Point

from location_field.forms.plain import PlainLocationField

class Address(forms.Form):
    city = forms.CharField()
    location = LocationField(based_fields=['city'],
                            initial=Point(-49.1607606, -22.2876834))
```

1.4 Changelog

1.4.1 Versions

2.0.0

- Added support to multiple map providers (Google, Mapbox, OpenStreetMap)
- Refactored the Javascript code to use [Leaflet.js](#)

1.4.2 Upgrading

From 1.x to 2.x

Backward incompatible changes

- Form fields no longer accepts the `default` parameter. If you are using them directly, without the model field, you have to replace the `default` parameter by the well known standard parameter `initial`.
- The `based_fields` parameter of model fields now only expects field names as strings and not field instances.

CHAPTER 2

Indices and tables

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