django-location-field Documentation

Release 2.0.0

Caio Ariede

Jul 08, 2020

Contents

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.

CHAPTER 1

Contents

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 Tutorials

1.2.1 Using django-location-field in the Django admin

Table of Contents

Installation

Using pip:

```
python -m pip install django-location-field
```

Update your settings

Example:

```
INSTALLED_APPS = [
    ...
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app',
    'location_field.apps.DefaultConfig',
]
```

Use one of the model fields (LocationField or PlainLocationField)

Example:

```
from django.db import models
from location_field.models.plain import PlainLocationField

class Place(models.Model):
    city = models.CharField(max_length=255)
    location = PlainLocationField(based_fields=['city'], zoom=7)
```

Register your model in the admin

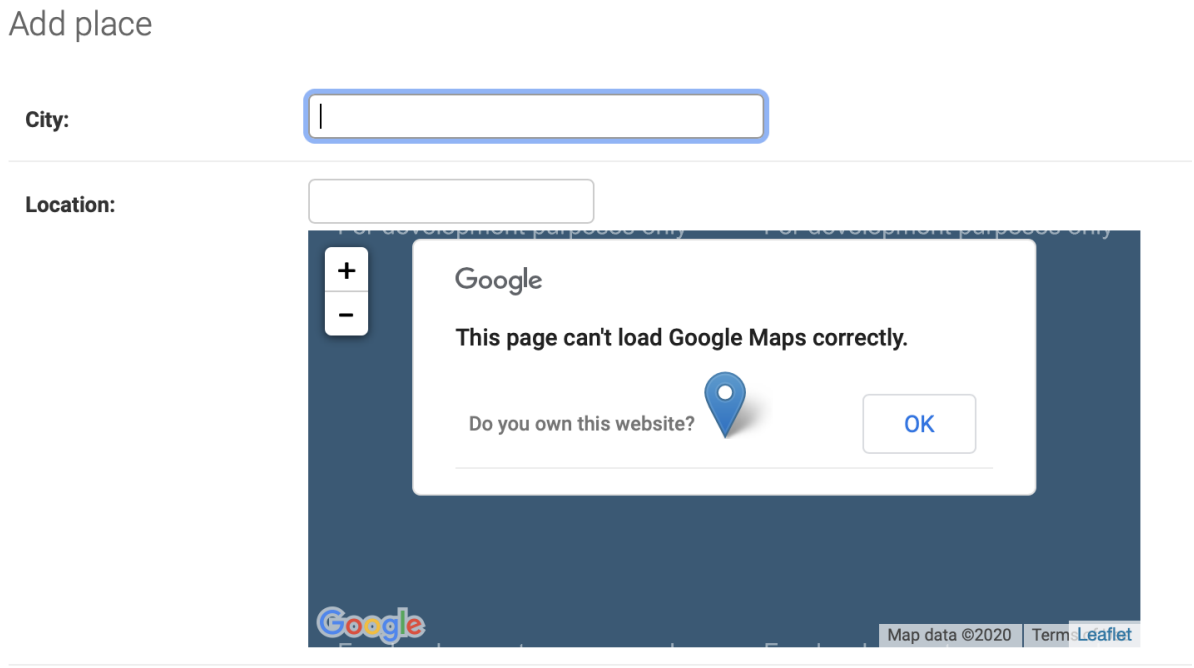
Example:

```
from django.contrib import admin

from .models import Place

admin.site.register(Place)
```

Note: You now should be able to see the map widget in the admin form. You will also notice that map is not loading correctly. This is because the default map provider is Google and it requires an API key.

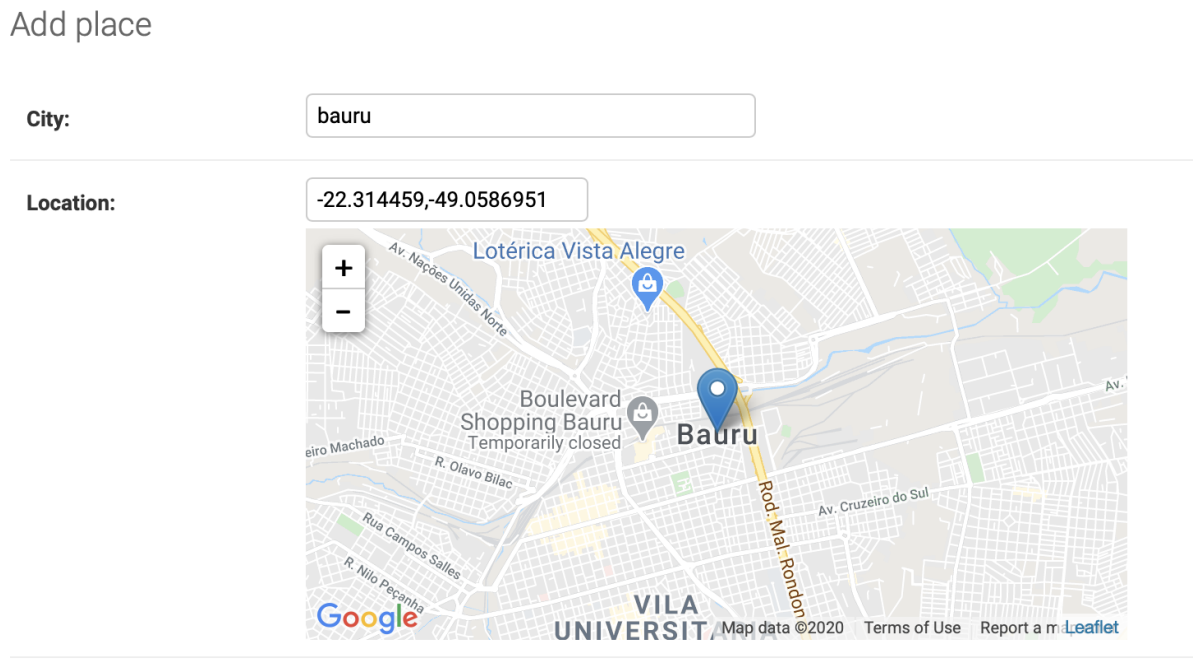


Providing an API Key

In your settings file, we need to provide the API key as shown below. Please notice that this will change depending on the Map Provider you want to use. For other map providers, please check the *Settings* page.

```
LOCATION_FIELD = {
    'provider.google.api': '//maps.google.com/maps/api/js?sensor=false',
    'provider.google.api_key': '<PLACE YOUR API KEY HERE>',
    'provider.google.api_libraries': '',
    'provider.google.map.type': 'ROADMAP',
}
```

And now you should be able to see the map render correctly:



Note: If you are not sure how to get the API key, one recommendation is to search for: `<your api provider> api key` (eg. "google maps api key"), using your favorite search engine :)

Note: If your map still does not work as expected, look at the browser console. Usually Map Providers will use the console to communicate possible issues (billing, for example)

1.3 Settings

1.3.1 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',
```

(continues on next page)

(continued from previous page)

```
    # Mapbox
    'provider.mapbox.access_token': '',
    'provider.mapbox.max_zoom': 18,
    '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/form.js',
        ),
    },
}
```

1.3.2 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.

1.4 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
based_fields	A list of fields that will be used to populate the location field
zoom	The default zoom level for the map
suffix	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.

1.4.1 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

```

(continues on next page)

(continued from previous page)

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

1.4.2 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))
```

1.5 Changelog

1.5.1 Versions

2.0.0

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

1.5.2 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