Django Map Widgets Documentation

Release 0.1.9

Erdem Ozkol

Mar 09, 2019

Contents

1 Achievements **3**
1.1 Index 3

2 Indices and tables **13**

Configurable, pluggable and more user friendly map widgets for Django PostGIS fields.

Project Home Page: https://github.com/erdem/django-map-widgets

CHAPTER 1

Achievements

The aim of the Django map widgets is to make all Geo Django widgets more user friendly and configurable. Map widgets are currently supporting only Google Map services, but we are planning to add other major map services.

1.1 Index

1.1.1 Installation

Note: The library has been tested against Python 2.7 and 3.4-3.6. It is compatible with Django 1.6-2.1.

Installing from PyPi

Note: This is the preferred installation method.

```
$ pip install django-map-widgets
```

Installing from source

Alternatively, install the package from github

```
$ pip install git+git://github.com/erdem/django-map-widgets.git
```

Add 'map_widgets' to your *INSTALLED_APPS* in settings.py

```
INSTALLED_APPS = [
     ...
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'mapwidgets',
]
```

Collects the static files into `STATIC_ROOT`.

```
python manage.py collectstatic
```

If you are using `mapwidgets` in your regular django views, you need to add `{{ form.media }}` template variable to the `<head>` or end of the `<body>` HTML section of your templates:

```
<!DOCTYPE html>
<html>
<head>
  ...
  {{ form.media }}
</head>
<body>
...
</body>
</html>
```

Note: If you are using `mapwidgets` on the Django Admin, you don't need to add `{{ form.media }}` any template files. The media variable already added in default Django Admin templates.

Requirements

Django Map Widgets needs Jquery dependency to work in your regular views. In Django Admin case, you don't need to provide the jQuery just because it's already available on `django.jQuery` namespace.

1.1.2 Map Widgets

Google Map Point Field Widget

Preview

Tip: The widget has a Google Place Autocomplete widget as a default. You can find a specific address with it.

Tip: The widget has built-in geocoding support. The autocomplete input will be filled by google geocoding service when the user adds a marker manually.

Settings

- **GOOGLE_MAP_API_KEY**: Put your Google API key (required)
- **GOOGLE_MAP_API_SIGNATURE**: You can give Google Static Map API signature key (optional). Check out this page.
- **LANGUAGE**: Google Map language (optional, default value is `en`).
- **mapCenterLocationName**: You can give a specific location name for center of the map. Map widget will find this location coordinates using Google Place Autocomplete. (Optional)
- **mapCenterLocation**: You can give specific coordinates for center of the map. Coordinates must be list type. ([latitude, longitude]) (Optional)
- **zoom** : Default zoom value for maps (optional, default value is 6).
- **markerFitZoom** : When the marker is initialized google's default zoom is set to Max. This method sets the zoom level a reasonable distance and center the marker on the map.

Usage

Settings

In your `settings.py` file, add your `MAP_WIDGETS` config:

```
MAP_WIDGETS = {
    "GooglePointFieldWidget": (
        ("zoom", 15),
        ("mapCenterLocationName", "london"),
        ("GooglePlaceAutocompleteOptions", {'componentRestrictions': {'country': 'uk'}
↪}),
        ("markerFitZoom", 12),
    ),
    "GOOGLE_MAP_API_KEY": "<google-api-key>"
}
```

If you want to give specific location name or coordinates for center of the map, you can update your settings like that.

```
MAP_WIDGETS = {
    "GooglePointFieldWidget": (
        ("zoom", 15),
        ("mapCenterLocation", [57.7177013, -16.6300491]),
    ),
    "GOOGLE_MAP_API_KEY": "<google-map-api-key>"
}
```

```
MAP_WIDGETS = {
    "GooglePointFieldWidget": (
        ("zoom", 15),
        ("mapCenterLocationName", 'Canada'),
    ),
    "GOOGLE_MAP_API_KEY": "<google-map-api-key>"
}
```

Tip: If there is no specific value set for the map center for `mapCenterLocationName`, `mapCenterLocation` the widget will be centred by the timezone setting of the project Check out these links.

- Timezone Center Locations
- countries.json

You can also give specific *settings* as a parameter for each widget.

Note: Google Map is using SRID (Spatial Reference System Identifier) as *4326* as same as Django's default SRID value for postgis fields. If you are set SRID parameter on a postgis field, the coordinates will store as your SRID format on your database but the widget always converting coordinates to *4326* format when it rendering. Because, the Google Map Javascript API using *4326* format. So, you can see different coordinates values on frontend from your DB but the point will always some location. You reach more information on this Wikipedia page.

```
from django.contrib.gis import forms
from mapwidgets.widgets import GooglePointFieldWidget

CUSTOM_MAP_SETTINGS = {
    "GooglePointFieldWidget": (
        ("zoom", 15),
        ("mapCenterLocation", [60.7177013, -22.6300491]),
    ),
}

class CityAdmin(admin.ModelAdmin):
    formfield_overrides = {
        models.PointField: {"widget": GooglePointFieldWidget(settings=CUSTOM_MAP_
↪SETTINGS)}
    }
```

Note: *GOOGLE_MAP_API_KEY* must be set in the project Django settings file for custom settings usage.

Django Admin

```
from mapwidgets.widgets import GooglePointFieldWidget

class CityAdmin(admin.ModelAdmin):
    formfield_overrides = {
        models.PointField: {"widget": GooglePointFieldWidget}
    }
```

Django Forms

```
from mapwidgets.widgets import GooglePointFieldWidget

class CityAdminForm(forms.ModelForm):
    class Meta:
        model = City
        fields = ("coordinates", "city_hall")
        widgets = {
            'coordinates': GooglePointFieldWidget,
            'city_hall': GooglePointFieldWidget,
        }
```

Javascript Triggers

If you need to develop your map UI on front-end side, you can use map widget jQuery triggers.

- **google_point_map_widget:marker_create**: Triggered when user create marker on map. (callback params: place, lat, lng, locationInputElem, mapWrapID)
- **google_point_map_widget:marker_change**: Triggered when user change marker position on map. (callback params: place, lat, lng, locationInputElem, mapWrapID)
- **google_point_map_widget:marker_delete**: Triggered when user delete marker on map. (callback params: lat, lng, locationInputElem, mapWrapID)

```
$(document).on("google_point_map_widget:marker_create", function (e, place, lat, lng,␣
↪locationInputElem, mapWrapID) {
    console.log(place); // Google geocoding response object
    console.log(locationInputElem); // django widget textarea widget (hidden)
    console.log(lat, lng); // created marker coordinates
    console.log(mapWrapID); // map widget wrapper element ID
});

$(document).on("google_point_map_widget:marker_change", function (e, place, lat, lng,␣
↪locationInputElem, mapWrapID) {
    console.log(place); // Google geocoding response object
    console.log(locationInputElem); // django widget textarea widget (hidden)
    console.log(lat, lng);  // changed marker coordinates
    console.log(mapWrapID); // map widget wrapper element ID
});

$(document).on("google_point_map_widget:marker_delete", function (e, lat, lng,␣
↪locationInputElem, mapWrapID) {
    console.log(locationInputElem); // django widget textarea widget (hidden)
    console.log(lat, lng);  // deleted marker coordinates
    console.log(mapWrapID); // map widget wrapper element ID
})
```

Reach Javascript Objects

The widget JS objects associated to the map HTML elements with jQuery *$.data* method. You can reach Google Map object and the widget class object instance.

Google Map Widget for Django Admin Inlines

Preview

As you know, Django Admin has an inline feature where you can add an inline row dynamically. In this case, Django default map widget doesn't initialize widget when created a new inline row.

If you want to use Google Map Widget on admin inlines with no issue, you just need to use `GooglePointFieldInlineWidget` class.

Note: This widget working with *Google Map Point Field Widget* settings.

Usage

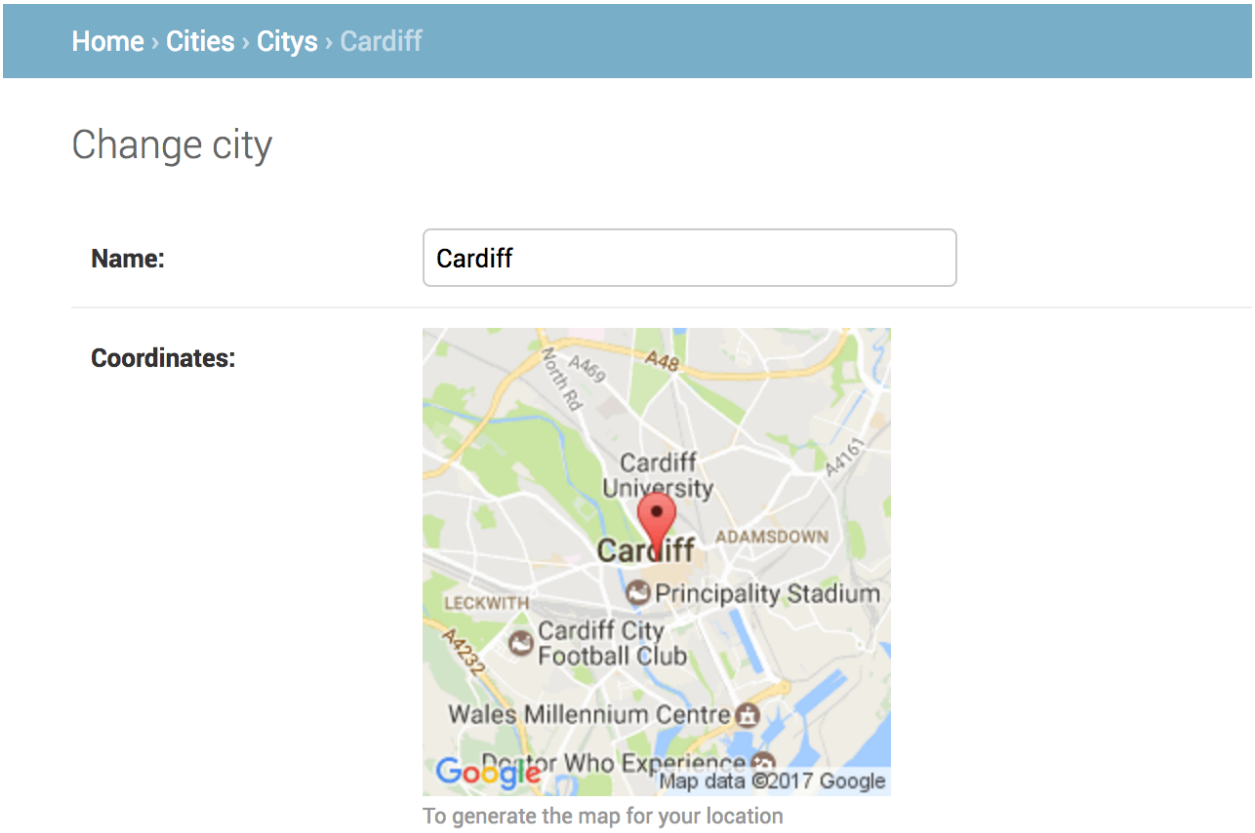
```
from mapwidgets.widgets import GooglePointFieldInlineWidget

class DistrictAdminInline(admin.TabularInline):
    model = District
    extra = 3
    formfield_overrides = {
        models.PointField: {"widget": GooglePointFieldInlineWidget}
    }

class CityAdmin(admin.ModelAdmin):
    inlines = (DistrictAdminInline,)
```

Google Map Static Widget

Preview



Django map widgets provide all Google Static Map API features. Check out this link for the google static map API features.

Here is the all default settings attribute for google static map widget.

```
MAP_WIDGETS = {
    "GoogleStaticMapWidget": (
        ("zoom", 15),
        ("size", "480x480"),
        ("scale", ""),
        ("format", ""),
```

(continues on next page)

(continued from previous page)

```
        ("maptype", ""),
        ("path", ""),
        ("visible", ""),
        ("style", ""),
        ("language", ""),
        ("region", "")
    ),

    "GoogleStaticMapMarkerSettings": (
        ("size", "normal"),
        ("color", ""),
        ("icon", ""),
    ),
    "LANGUAGE": "en",
    "GOOGLE_MAP_API_SIGNATURE": "",
    "GOOGLE_MAP_API_KEY": "",
}
```

Usage

If you are not using specific features on Google Static Map API, you just need to update GOOGLE_MAP_API_KEY value in your Django settings file. If you also need individual size map images, you can pass *size* and *zoom* parameter for each GoogleStaticMapWidget class.

Settings

In your `settings.py` file, add your `MAP_WIDGETS` config:

```
MAP_WIDGETS = {
    "GoogleStaticMapWidget": (
        ("zoom", 15),
        ("size", "320x320"),
    ),
    "GoogleStaticMapMarkerSettings": (
        ("color", "green"),
    ),
    "GOOGLE_MAP_API_KEY": "<google-map-api-key>"
}
```

Django Admin

```
from mapwidgets.widgets import GoogleStaticMapWidget

class CityAdmin(admin.ModelAdmin):
    formfield_overrides = {
        models.PointField: {"widget": GoogleStaticMapWidget}
    }
```

Django Forms

```
from mapwidgets.widgets import GoogleStaticMapWidget

class CityDetailForm(forms.ModelForm):

    class Meta:
        model = City
        fields = ("name", "coordinates", "city_hall")
```

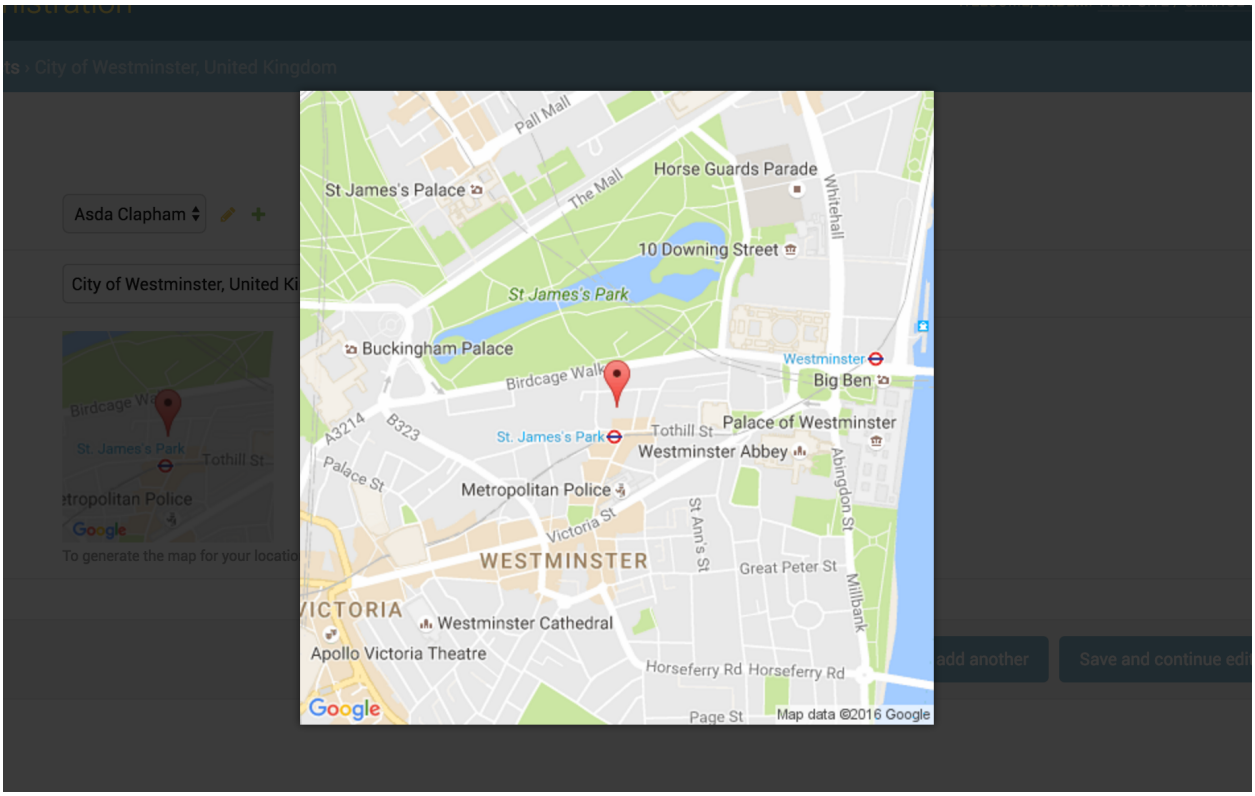
(continues on next page)

(continued from previous page)

```
        widgets = {
            'coordinates': GoogleStaticMapWidget,
            'city_hall': GoogleStaticMapWidget(zoom=12, size="240x240"),
        }
```

Google Map Static Overlay Widget

Preview



This widget is working with Magnific Popup jQuery plugin. The plugin javascript file load with the widget static files.

Usage

You can also use all static map features in this widget. Besides you can give a `thumbnail_size` value.

Here is the all default settings attribute for google static overlay map widget.

```
MAP_WIDGETS = {
    "GoogleStaticMapMarkerSettings": (
        ("size", "normal"),
        ("color", ""),
        ("icon", "")
    ),

    "GoogleStaticOverlayMapWidget": (
        ("zoom", 15),
        ("size", "480x480"),
        ("thumbnail_size", "160x160"),
        ("scale", ""),
```

(continues on next page)

(continued from previous page)

```
        ("format", ""),
        ("maptype", ""),
        ("path", ""),
        ("visible", ""),
        ("style", ""),
        ("language", ""),
        ("region", "")
    ),

    "GOOGLE_MAP_API_SIGNATURE": "",
    "GOOGLE_MAP_API_KEY": "",
}
```

Settings

In your django `settings.py` file, add your `MAP_WIDGETS` config:

```
MAP_WIDGETS = {
    "GoogleStaticMapWidget": (
        ("zoom", 15),
        ("size", "320x320"),
        ("thumbnail_size", "100x100"),
    ),
    "GoogleStaticMapMarkerSettings": (
        ("color", "green"),
    ),
    "GOOGLE_MAP_API_KEY": "<google-map-api-key>"
}
```

Django Admin

```
from mapwidgets.widgets import GoogleStaticOverlayMapWidget

class CityAdmin(admin.ModelAdmin):
    formfield_overrides = {
        models.PointField: {"widget": GoogleStaticOverlayMapWidget}
    }
```

Django Forms

```
from mapwidgets.widgets import GoogleStaticOverlayMapWidget

class CityDetailForm(forms.ModelForm):

    class Meta:
        model = City
        fields = ("name", "coordinates", "city_hall")
        widgets = {
            'coordinates': GoogleStaticOverlayMapWidget,
            'city_hall': GoogleStaticOverlayMapWidget(zoom=12, size="240x240"),
        }
```

1.1.3 Release Notes

v0.1.8

- Full documentation integrated to readthedocs.org.
- Fixed Google Map static widget issues.
- Added Russian localisation support.
- Added Google Places Autocomplete options support.
- Fixed CSS issues.

v0.1.9

- Google Place Autocomplete object binding to jQuery triggers.
- Implemented Google Geocoding support for the marker coordinates.
- Added custom widget settings feature for each widget.
- Added Portuguese localisation support.
- Fixed Google Place Autocomplete widget bugs in Django Admin Inlines.
- Fixed Python 3.6 errors.
- Fixed Javascript bugs.
- The GitHub repository Integrated with Travis CI.
- Implemented unit tests for backend code. (%100 code coverage)
- Change development environment from Vagrant to Docker.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search