anycluster Documentation

Release 1.0

biodiv

Sep 09, 2019

Contents

1 Installation and Configuration **3**
- 1.1 Prequisities . . . 3
- 1.2 Installation and Configuration . . . 3

2 Usage **5**
- 2.1 Initialization . . . 5
- 2.2 Properties of the anycluster instance . . . 7
- 2.3 Methods . . . 7

3 Filters **9**
- 3.1 filterObject . . . 9
- 3.2 Operators . . . 9

4 Cusotmization **11**
- 4.1 Single Pin Images . . . 11

5 Indices and tables **13**

Contents:

CHAPTER 1

Installation and Configuration

1.1 Prequisities

- Django 2.x
- python3
- Geodjango
- PostGis 2.x
- (for kmeans clustering, recommended) kmeans PostgreSQL extension: https://github.com/umitanuki/kmeans-postgresql

1.2 Installation and Configuration

Installing the kmeans PostgreSQL extension (optional, needed for kmeans clustering)

1. Download and unzip https://github.com/umitanuki/kmeans-postgresql on your server.
2. make sure you have the development packages for you postgresql server package installed (e.g. sudo apt-get install libpq-dev postgresql-server-dev-10)
3. In your unzipped kmeans folder run the following (e.g. on ubuntu)

```
make
sudo make install
psql -f /usr/share/postgresql/10/extension/kmeans.sql -d YOURGEODJANGODATABASE
```

   The latter needs to be processed as a postgresql superuser, e.g. the user postgres. You now have access to the kmeans functions which are necessary for the pin-based clustering.

Install anycluster with your Django installation

1. use `pip install anycluster` OR unzip the folder anycluster into your project directory

2. add 'anycluster' to your INSTALLED_APPS
3. required SETTINGS (settings.py)

```
ANYCLUSTER_GEODJANGO_MODEL = "yourapp.models.your_geodjango_model"
ANYCLUSTER_COORDINATES_COLUMN = "your_geometric_column"
```

4. urls.py

```
path('anycluster/', include('anycluster.urls')),
```

Load the needed javascript modules and css styles

1. Load google maps api. Add the following inside the <head> tag of your website:

```
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?&
↪key=YOURKEY&sensor=false"></script>
```

   Replace YOURKEY with the key you obtained from google.

2. Load anycluster modules. Add the following inside the <head> tag of your website:

```
<script type="text/javascript" src="{% static 'anycluster/anycluster.js' %}"></
↪script>
<script type="text/javascript" src="{% static 'anycluster/django_ajax_csrf.js' %}
↪"></script>
<script type="text/javascript" src="{% static 'anycluster/anycluster_marker.js' %}
↪"></script>
<link rel="stylesheet" href="{% static 'anycluster/anycluster.css' %}">
```

That's it! you are now ready to cluster your map markers!

CHAPTER 2

Usage

2.1 Initialization

First you have to initialize your map.

```
var anyclusterSettings = {
    mapType : "leaflet"
}

anycluster = new Anycluster("div_id", anyclusterSettings);
```

2.1.1 anyclusterSettings Properties

mapType *string* currently only supports "google". Support for "openlayers" and "leaflet" is planned. This is the only required property.

autostart *boolean* true or false. Defaults to true. Defines if the map is automatically clustered when loaded

center *list* defaults to [0,0]. Set the center your map should begin at in [longitude,latitude].

zoom *integer* defaults to 3. Set the initial zoom level.

clusterMethod *string* "grid" or "kmeans". Defaults to "grid".

iconType *string* "exact" or "rounded". Affects kmeans clustering only, defaults to "exact". Determines if the counts on the clusters are displayed as an exact or rounded number.

gridSize *integer* defaults to 256. Defines the size of the grid for both kmeans and grid clustering.

mapTypeId *string* defaults to "HYBRID". Possible values are "ROADMAP", "SATELLITE", "HYBRID" or "TERRAIN".

filters *object* defaults to {}. See filter documentation.

singlePinImages *object* defaults to {}.

onFinalClick *function* function fired when the clusters cannot be declustered anymore.

loadEnd *function* callback function fired after every clustering event.

loadStart *function* callback function fired before every clustering event.

Example (google):

```
var anyclusterSettings = {
              mapType : "google",
              google : {
                      mapTypeId: "HYBRID",
              },
              gridSize: 256,
              zoom: 3,
              center: [30,30],
              clusterMethod : "kmeans",
              iconType: "exact",
              singlePinImages: {
                      'blue':'/static/anycluster/pin_blue.png',
                      'red':'/static/anycluster/pin_red.png',
              },
              onFinalClick : function(entries){
                      var dialog = $("#dialog");
                      dialog.html(entries);
                      dialog.dialog('open');
              },
              loadEnd: function(){
                      $("#totalcount").text(anycluster.viewportMarkerCount);
              },
              autostart: true,
              filters: { "is_faved": {"values" : true} }
}

anycluster = new Anycluster("mymap", anyclusterSettings);
```

Example (leaflet):

```

```

var anyclusterSettings = { mapType : “leaflet”, gridSize: 256, zoom: 3, center: [30,30], clusterMethod : “kmeans”, iconType: “exact”, singlePinImages: {

‘blue’:’/static/anycluster/pin_blue.png’, ‘red’:’/static/anycluster/pin_red.png’,

}, onFinalClick : function(entries){

var dialog = $(“#dialog”); dialog.html(entries); dialog.dialog(‘open’);

}, loadEnd: function(){

$(“#totalcount”).text(anycluster.viewportMarkerCount);

}, autostart: true, filters: { “is_faved”: {“values” : true} }

}

anycluster = new Anycluster(“mymap”, anyclusterSettings);

2.2 Properties of the anycluster instance

anyclusterInstance.viewportMarkerCount contains the count of all items that are currently on the viewport

2.3 Methods

anyclusterInstance.filter(filterObject) applies all filters of the filterObject and reclusters the map. See filter documentation for filterObjects.

CHAPTER 3

Filters

3.1 filterObject

A filterObject looks like this:

```
var filterObj = [
   {"db_column_name" : { "values": value, "operator": operator_string }}
]
```

value *list* or *string*

operator_string *string* that determines the database query

Example:

```
var filters = [
   {"color": {"values" : "red", "operator":"!=" }},
   {"number": {"values": [2,3], "operator": "either_="}}
]

anyclusterInstance.filter(filters);
```

This will result in the following query:

```
WHERE color != red AND (number=2 OR number=3)
```

Note: filterObjects are ANDed together. For OR lookups use the either_ operator.

3.2 Operators

- = (default)

- !=
- >=
- <=
- >
- <
- contains
- startswith
- either_[operator]

If the value is a list, the operator can be prefixed with “either_” to apply to each item of the list.

CHAPTER 4

Cusotmization

4.1 Single Pin Images

The pin of markers with count 1 can be customized according to the value of a database column. To enable this you have to add the following to your settings.py:

settings.py

```
ANYCLUSTER_PINCOLUMN = "db_column"
```

with db_column being the column that determines the image for pins with count 1.

anyclusterSettings

```
var anyclusterSettings = {
            mapType : "google",
            singlePinImages: {
                    'db_value1':'/static/path/to/pin_value1.png',
                    'db_value2':'/static/path/to/pin_value2.png',
            }
}
```

Example:

```
ANYCLUSTER_PINCOLUMN = "color"
```

```
var anyclusterSettings = {
            mapType : "google",
            singlePinImages: {
                    'blue':'/static/path/to/pin_blue.png',
                    'red':'/static/path/to/pin_red.png',
            }
}
```

In this example, the database column that determines the pin images for single pins is “color”. If the column has the value “blue” for the pin, the image will be “/static/path/to/pin_blue.png”

CHAPTER 5

Indices and tables

- genindex
- modindex
- search