Flower

Kivy

May 20, 2020

CONTENTS:

`MapView` is a Kivy widget specialized into tiles-based map rendering.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

1.1.1 Supported versions

- Python 3.5+

1.1.2 Dependencies

1. Kivy
2. concurrent.futures
3. requests

1.1.3 Installation

Install it from PyPI via:

```
python -m pip install kivy_garden.mapview
```

Or even shorter:

```
python -m pip install mapview
```

Alternatively, you can install mapview develop directly from github with:

```
python -m pip install https://github.com/kivy-garden/mapview/archive/master.zip
```

Look under the releases tab if you'd like to install a specific release or a pre-compiled wheel, if the flower has any. Then use the url with *pip*.

Or you can automatically install it using garden's pypi server with:

```
python -m pip install kivy_garden.mapview --extra-index-url https://kivy-garden.
↪github.io/simple/
```

To permanently add our garden server to your pip configuration so that you don't have to specify it with *–extra-index-url*, add:

```
[global]
timeout = 60
index-url = https://kivy-garden.github.io/simple/
```

to your pip.conf.

Please see the garden docs for further installation instructions.

1.2 Usage

1.2.1 Basic Usage

If you use Kivy garden, you can import the widget like this:

```
from kivy.garden.mapview import MapView, MarkerMap
map = MapView()
```

You can customize the default zoom and center the view on Lille by:

```
map = MapView(zoom=9, lon=50.6394, lat=3.057)
```

Then, you can create marker and place them on the map. Normally, anything that goes on a map should go on a `MapLayer`. Hopefully, the `MapView` give an API for adding marker directly, and creates a `MarkerMapLayer` if you did'nt created one yet:

```
m1 = MapMarker(lon=50.6394, lat=3.057)  # Lille
m2 = MapMarker(lon=-33.867, lat=151.206)  # Sydney
map.add_marker(m1)
map.add_marker(m2)
```

You can also change the providers by:

1. using a provider key:

   ```
   map.map_source = "mapquest-osm"
   ```

2. using a new MapSource object:

   ```
   source = MapSource(url="http://my-custom-map.source.com/{z}/{x}/{y}.png",
                      cache_key="my-custom-map", tile_size=512,
                      image_ext="png", attribution="@ Myself")
   map.map_source = source
   ```

1.3 Examples

1.3.1 Basic Example

Add your examples here

CHAPTER

TWO

THE MAPVIEW API

2.1 Mapview

2.1.1 kivy_garden.mapview

class mapview.**Coordinate**(*lon*, *lat*)

Named tuple that represent a geographic coordinate with latitude/longitude

Parameters

- **lon** (*float*) – Longitude
- **lat** (*float*) – Latitude

class mapview.**MapSource**(*url*, *cache_key*, *min_zoom*, *max_zoom*, *tile_size*, *image_ext*, *attribution*, *subdomains*)

Class that represent a map source. All the transformations from X/Y/Z to longitude, latitude, zoom, and limitations of the providers goes are stored here.

Parameters

- **url** (*str*) – Tile's url of the providers. Defaults to *http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png*
- **cache_key** (*str*) – Key for storing the tiles. Must be unique and not colliding with another providers, otherwise tiles will not be downloaded again. Defaults to "osm"
- **min_zoom** (*int*) – Minimum zoom value acceptable for this provider. Defaults to 0.
- **max_zoom** (*int*) – Maximum zoom value acceptable for this provider. Defaults to 19.
- **tile_size** (*int*) – Size of a image tile returned by the provider. Defaults to 256.
- **attribution** (*str*) – Attribution for this provider. Defaults to empty string
- **subdomains** (*str*) – Domains substitutions for the {s} in the url. Defaults to "abc"

get_x(*zoom*, *lon*)

Get the x position to the longitude in the map source's projection

Parameters

- **zoom** (*int*) – Zoom level to look at
- **lon** (*float*) – Longitude

Returns X position

Return type float

get_y(*zoom*, *lat*)
Get the y position to the latitude in the map source's projection

Parameters

- **zoom** (*int*) – Zoom level to look at
- **lat** (*float*) – Latitude

Returns Y position

Return type float

get_lon(*zoom*, *x*)
Get the longitude to the x position in the map source's projection

Parameters

- **zoom** (*int*) – Zoom level to look at
- **x** (*float*) – X position in the map

Returns Longitude

Return type float

get_lat(*zoom*, *y*)
Get the latitude to the y position in the map source's projection

Parameters

- **zoom** (*int*) – Zoom level to look at
- **y** (*float*) – Y position in the map

Returns Latitude

Return type float

get_col_count(*zoom*)
Return the number of column for this provider at this zoom level.

Parameters **zoom** (*int*) – Zoom level to look at

Returns Number of column

Return type int

get_row_count(*zoom*)
Return the number of row for this provider at this zoom level.

Parameters **zoom** (*int*) – Zoom level to look at

Returns Number of rows

Return type int

get_max_zoom()
Return the maximum zoom of this source

Returns Maximum zoom

Return type int

get_min_zoom()
Return the minimum zoom of this source

Returns Minimum zoom

Return type int

class mapview.**MapMarker**

A marker on the map, that must be used on a *MapMarker*, or with *MapView.add_marker()* or with MapView.add_widget()

Events *on_press*: Fired when the MapMarker is pressed *on_release*: Fired when the MapMarker is release

anchor_x

Anchor of the Marker on the X axis. Defaults to 0.5, means the anchor will be at the X center of the image

anchor_y

Anchor of the marker on the Y axis. Defaults to 0, means the anchor will be at the Y bottom of the image

lat

Latitude of the marker

lon

Longitude of the marker

source

Image source of the marker, defaults to *marker.png* within the mapview package.

class mapview.**MapView**

MapView is a widget that control the map displaying, navigation and layers management.

Available events *on_map_relocated*: called everytime the MapView change location

lon

Longitude at the center of the widget, read-only.

lat

Latitude at the center of the widget, read-only.

zoom

Zoom of the MapView. Must be between *MapSource.get_min_zoom()* and *MapSource.get_max_zoom()*. Default to 0

map_source

Provider of the map, default to an empty *MapSource*

double_tap_zoom

If True, this will activate the double-tap to zoom.

Defaults to False.

pause_on_action

Pause on any loading / tiles loading when an action is done. This allow better performance on mobile, but can be safely deactivated on desktop.

Defaults to True.

scale

Current scale of the internal scatter, read-only. This is usually not used in user-side unless you're hacking mapview.

snap_to_zoom

When the user initiate a zoom, it will snap to the closest zoom for better graphics. The map can be blur if the map is scaled between 2 zoom.

Defaults to True, even if it doesn't fully working yet.

add_layer(*layer*)
Add a new layer to update at the same time than the base tile layer

Parameters **layer** (MapLayer) – Map layer to add

add_marker(*marker*, *layer=None*)
Add a marker into a *layer*. If *layer* is None, it will be added in the default marker layer. If there is no default marker layer, a new one will be automatically created.

Parameters

- **marker** (MapMarker) – The marker to add
- **layer** (MarkerMapLayer) – The layer to use

center_on(*lat*, *lon*)
Center the map on the coordinate (lat, lon)

Parameters

- **lat** (*float*) – Latitude
- **lon** (*float*) – Longitude

get_bbox(*margin=0*)
Returns the bounding box from the bottom-left to the top-right.

Parameters **margin** (*float*) – Optionnal margin to extend the Bbox bounds

Returns Bounding box

Return type Bbox

get_latlon_at(x, y, zoom=None):
Return the current coordinate (lat, lon) at the (x, y) widget coordinate

Parameters

- **x** (*float*) – X widget coordinate
- **y** (*float*) – Y widget coordinate

Returns lat/lon Coordinate

Return type *Coordinate*

remove_layer(*layer*)
Remove a previously added *MapLayer*

Parameters **layer** (MapLayer) – A map layer

remove_marker(*marker*)
Remove a previously added MarkerMap

Parameters **marker** (*MarkerMap*) – The marker

set_zoom_at(*zoom*, *x*, *y*, *scale=None*)
Sets the zoom level, leaving the (x, y) at the exact same point in the view.

Parameters

- **zoom** (*float*) – New zoom
- **x** (*float*) – X coordinate to zoom at
- **y** (*float*) – Y coordinate to zoom at
- **scale** (*float*) – (internal) Scale to set on the scatter

unload()
Unload the view and all the layers. It also cancel all the remaining downloads. The map should not be used after this.

class mapview.**MapLayer**
A map layer. It is repositioned everytime the *MapView* is moved.

reposition()
Function called when the *MapView* is moved. You must recalculate the position of your children, and handle the visibility.

unload()
Called when the view want to completely unload the layer.

class mapview.**MarkerMapLayer**(*MapLayer*)
A map layer specialized for handling *MapMarker*.

class mapview.mbtsource.**MBTilesMapSource**(*MapSource*)
Use a Mbtiles as a source for a MapView

class mapview.geojson.**GeoJsonMapLayer**(*MapLayer*)
A Geojson MapLayer.

Experimental, only Polygon and LineString feature are supported. Marker are not yet implemented, due to lack of API for wiring Marker selection back to you.

source

A Geojson filename to load, defaults to None.

geojson

A dictionary structured as a Geojson. This attribute contain the content of a *source* if passed.

class mapview.clustered_marker_layer.**ClusteredMarkerLayer**(*MapLayer*)
Experimental Layout that implement marker clustering. It implement its own version of Super Cluster, based itself on a KD-tree.

Aka you can load like 2000 markers without issues. The cluster index is immutable, so if you add a new marker, it will be rebuild from scratch.

Please note that the widget creation is done on the fly by the layer, not by you.

DONT use *add_widget*, use *add_marker()*

Example:

```
layer = ClusteredMarkerLayer()
for i in range(2000):
    lon = random() * 360 - 180
    lat = random() * 180 - 90
    layer.add_marker(lon=lon, lat=lat, cls=MapMarker)

# then you can add the layer to your mapview
mapview = MapView()
mapview.add_widget(layer)
```

cluster_cls

Reference to the class widget for creating a cluster widget. Defaults to *ClusterMapMarker*

cluster_min_zoom

Minimum zoom level at which clusters are generated. Defaults to 0

cluster_max_zoom

Maximum zoom level at which clusters are generated. Defaults to 16

cluster_radius

Cluster radius, in pixels. Defaults to 40dp

cluster_extent

Tile extent. Radius is calculated relative to this value. Defaults to 512.

cluster_node_size

Size of the KD-tree leaf node. Affects performance. Defaults to 64.

add_marker(*lon*, *lat*, *cls=MapMarker*, *options=None*)

Method to add a marker to the layer.

Parameters

- **lon** (*float*) – Longitude
- **lat** (*float*) – Latitude
- **cls** (*object*) – Widget class to use for creating this marker. Defaults to `MapMarker`
- **options** (*dict*) – Options to pass to the widget at instanciation. Defaults to an empty dict.

Returns The instance of a Marker (internal class, not the widget)

Method to call for building the cluster. It is done automatically at the first rendering. If you missed it, or need to rebuild after readding marker, just call this function.

class mapview.clustered_marker_layer.**ClusterMapMarker**(*MapMarker*)

Widget created for displaying a Cluster.

cluster

Reference to the Cluster used for this widget

num_points

Number of marker that the cluster contain.

text_color

Color used for the text, defaults to [.1, .1, .1, 1]. If you want others options, best is to do your own cluster widget including the label you want (font, size, etc) and customizing the background color.

CHAPTER

THREE

CURRENT LIMITATIONS

- The API is still moving, it may contain errors.
- Some providers can be slow or timeout. This is not an issue from MapView.
- If a tile is not correctly downloaded or missing from the provider, the error will be showed on the console, but nothing happen on the map itself. This can lead to a defect user experience.
- When leaving, *concurrent.futures* are joining all the threads created. It can stuck the application at a maximum time of 5 seconds (requests timeout). More if the network is unstable. There is no way to force it yet.
- The cache is not controlable, if the user move the map a lot, it can fill the disk easily. More control will be given later.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

R

S

T

U

Z