GABBSPython Maps Library User Manual

Scientific Solutions Group

Rosen Center for Advanced Computing

Purdue University

2015-10-20

Content

1. What is GABBS Python Maps Library?

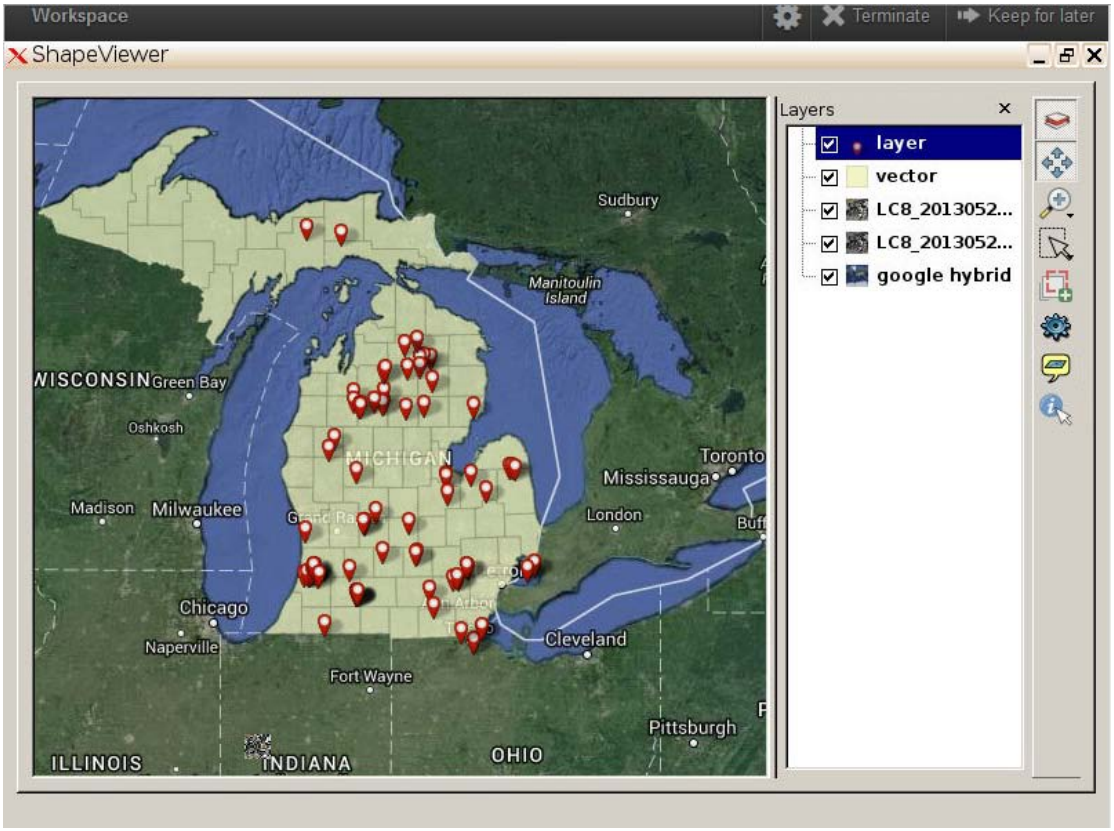
GABBS Python Maps Library allows MyGeoHub users and scientists to display maps, images, vectors and other geospatial information within their Hubzero Tools. It includes the GABBS Python Maps API and GABBS Map Widgets

GABBSPython Maps API enables developers to customize maps and the information on maps.GABBS MapWidgets provide a container for displaying maps without programming, which is an easy way to integrate maps into an application.

2. Basics

In this section we will show how to create a simple map using the GABBS Python Maps Library. We will start with a simple example and then go through the code step by step.

2.1 Create a simple map



This example creates a map show some vector of Michigan, U.S, and raster image of Purdue, West Lafayette, Indiana, U.S.

Example:

```
# Create a Map Container
        canvasProp = {'panControl':    True,
                      'zoomControl': True,
                      'selectControl': True,
                      'layerControl':    True }
        self.mapContainer = gabbs.maps.MapContainer(canvasProp)
        # Add canvas as a widget to the layout
        self.layout.addWidget(self.mapContainer)
        # Create a Map object
        mapProp = {'center':
                        {'lon': -86.9080556,
                         'lat': 40.4258333},
                   'zoom':       8,
                   'maxZoom': 18,
                   'minZoom': 1,
                   'mapTypeId': 'OSM'}
        self.map = gabbs.maps.Map(mapProp)
        self.mapContainer.addLayer(self.map)
        # Create a Raster object
        filePath = os.path.join("/home/mygeohub/wanwei/geo", "data", "LC8_20130524.tif")
        rasterProp = {"layerName":          "raster"}
        rasterProp["fileName"] = filePath
        self.raster = gabbs.maps.Raster(rasterProp)
        self.mapContainer.addLayer(self.raster)
        # Create a Vector object
        filePath = os.path.join(".", "data", "Counties", "Counties.shp")
        polygonProp = {"layerName":          "vector",
                       "strokeColor":      "#0000FF",
                       "strokeOpacity":    0.8,
                       "strokeWeight":      2,
                       "fillColor":          (40, 200, 160),
          "fillOpacity":        0.5}
        polygonProp["fileName"] = filePath
        self.polygon = gabbs.maps.Vector(polygonProp)
        self.mapContainer.addLayer(self.polygon)
        # Create a Point object
        filePath = os.path.join("/home/mygeohub/wanwei/geo", "data", "Burn_Point_Locations.csv")
        markerProp = {"layerName":          "layer",
                      "delimiter":          ",",
                      "xField":               "Longitude",
                      "yField":               "Latitude",
                      "useSystemStyle": True,
```

```
                    "styleName":        "SVG_MARKER_RED_MARKER"}
        markerProp["fileName"] = filePath
        self.point = gabbs.maps.DelimitedText(markerProp)
        self.mapContainer.addLayer(self.point)
```

The rest of this section describes the above example step by step.

2.2. Load the GABBS MapsAPI

The GABBS Maps API is a Python library. It can be added to a Python main programas following:

```
import gabbs.maps
# Initialize gabbs maps library
gabbs.maps.gbsLoadLibrary()
#adding your codes here
…
# Unload gabbs maps library
gabbs.maps.gbsUnloadLibrary()
```

2.3. Create a map container

Create a mapContainerobject to contain the map. mapContaineris a widget in PyQT. Use layout to size the element.

Note: The map will always "inherit" its size from its container layout.

In the initialize function, create adict object (canvasProp) to define the properties of the mapContainer, such as enabling control panels on the toolbar. The following shows example code that initializes the map container:

```
# Create a Map Container, aka map canvas
        canvasProp = {'panControl':    True,
                      'panControlOptions':
                      'zoomControl': True,
                      'selectControl': True,
                      'layerControl':    True,
                      'overviewControl':True}
        self.mapContainer = gabbs.maps.MapContainer(canvasProp)
        # Add canvas as a widget to the layout
        self.layout.addWidget(self.mapContainer)
```

2.4. Set map properties

The syntax for setting map properties follows the style of Google Maps API. All properties are stored in a Python dictionary, where property nameis the key and property value is the value. And the value is using native Python data types, such as Boolean, Integer, Float, and String.

In the initialize function, create a dict object (mapProp) to define the properties for the map. Here are the steps to initialize a map:

```
# Create a Map object
```

```
            mapProp = {'center':
                                {'lon': -85.508742,
                                 'lat': 46.120850},
                            'zoom':       6,
                            'maxZoom': 18,
                            'minZoom': 1,
   'mapTypeId': 'OSM'}
```

The center property specifies where to center the map. Create a Lat and Lonkey to center the map on a specific point. Pass the coordinates in the order: longitude, latitude.The zoom property specifies the zoom level for the map. A value of zero shows a map of the Earth fully zoomed out. Higher zoom levels zoom in at a higher resolution.The mapTypeId property specifies the map type to display. The following map types are supported:

- OSM (Open Street Map)

2.5. Create a map object

The following code creates a new map inside the main window class with member variable name "self.map ", using the parameters that are passed through mapProp.

```
self.map = gabbs.maps.Map(mapProp)
self.mapContainer.addLayer(self.map)
```

3. GABBS Maps Overlays

Overlays are objects on the map that are bound to latitude/longitude coordinates. Scientists can use overlay to display result images. GABBS Maps supports several types of overlays:

Vector:

This includes (1) polyline which includes series of straight lines on a map,(2) polygon which includes series of straight lines on a map and the shape is "closed",(3)circle, and (4) rectangle.

Raster:

This includes any GDAL supported image file format, such as tiff, geoTiff. The file may be single band or multiband.

DelimitedText:

This includes CSV, or other text file. This is often symbolized as markerson a map which can also display custom icon images.

In the future, GABBS Maps will support info windows, displaying content within a popup balloon on top of a map, and other custom overlays.

3.1.Vector

A vector polygon is similar to a polyline in that it consists of a series of coordinates in an ordered

sequence. However, polygons are designed to define regions within a closed loop.Polygons are made of straight lines, and the shape is "closed" (all the lines connect up).

A polygon overlay supports the following properties:
strokeColor - specifies a hexadecimal color for the line (format: "#FFFFFF")
strokeOpacity - specifies the opacity of the line (a value between 0.0 and 1.0)
strokeWeight - specifies the weight of the line's stroke in pixels
fillColor - specifies a hexadecimal color for the area within the enclosed region (format: "#FFFFFF")
fillOpacity - specifies the opacity of the fill color (a value between 0 and 100)
visible- defines whether the layer is visible by users (true/false)

Here is an example:

```
# Create a Vector object
        filePath = os.path.join(".", "data", "Counties", "Counties.shp")
        polygonProp = {"layerName":         "vector",
                                "strokeColor":    "#0000FF",
                                "strokeOpacity": 0.8,
                                "strokeWeight":    2,
                                "fillColor":       (40, 200, 160),
                                "fillOpacity":      0.5}
        polygonProp["fileName"] = filePath
        self.polygon = gabbs.maps.Vector(polygonProp)
        self.mapContainer.addLayer(self.polygon)
```

3.2 Raster

A rasteroverlay supports the following properties:
opacity - specifies the opacity of the fill color (a value between 0.0 and 1.0)
visible- defines whether the layer is visible by users (true/false)

Here is an example:

```
filePath = os.path.join("/home/mygeohub/wanwei/geo", "data", "LC8_20130524.tif")
        rasterProp = {"layerName":         "raster"}
        rasterProp["fileName"] = filePath
        self.raster = gabbs.maps.Raster(rasterProp)
        self.mapContainer.addLayer(self.raster)Try it yourself
```

3.3 DelimitedText

The DelimitedTextoverlay constructor creates a point layer. (Note that the position property must be set for the marker to display).Add the layer to the map by using the *addLayer()* method of *mapContainer* object.

Here is an example:

```
# Create a Point object
```

```
        filePath = os.path.join("/home/mygeohub/wanwei/geo", "data", "Burn_Point_Locations.csv")
        markerProp = {"layerName": "layer",
                      "delimiter": ",",
                      "xField": "Longitude",
                      "yField": "Latitude",
                      "strokeColor": "#0000FF",
                      "strokeOpacity": 0.8,
                      "strokeWeight": 2,
                      "fillColor": (255, 0, 0),
                      "fillOpacity": 1}
        markerProp["fileName"] = filePath
        self.point = gabbs.maps.DelimitedText(markerProp)
        self.mapContainer.addLayer(self.point)
```

4. GABBS Maps Events and Actions

4.1.Add an action to the map

Add an action that will execute a script function once a mouse click event is triggered when using map tools. It is also possible to link the action to a specific layer of the maps on demand. It provides an easy and safe way to add user defined functions to map events. In addition, user can get map attributes by using built-in key words in the action. Please see the GABBS Maps Action Reference for details.

The gbsAddAction function creates a GABBS Maps API action. The action script to perform when an event is triggered is added as a parameter to the action.

Here is an example that shows a message box of clicked mouse coordinates when user clicks on the map. It defines a “python” type of action to call the QMessageBox function. In the call to gbsAddAction(), self.polygon is the target layer, "python" is the action type, "massage" is the action name, “action1” is a string that contains thescript to be executed.

```
action1 = """
from PyQt4.QtCore import *
from PyQt4.QtGui import *
QMessageBox.information(None, "Info", "left button clicked x:[% $clickx %], y: [% $clicky %]
")
"""
gabbs.maps.gbsAddAction(self.polygon, "python", "massage", action1)
```

Below is another example that echoes an attribute value when a feature is clicked. It defines a

“generic” type of action to call the system echo command:

```
action2 = """ echo "[% "fpa_name" %]" """
gabbs.maps.addAction(self.point, "generic", "name", action2)
```

4.2 GABBS Maps Action Reference

Action Type:

Generic“generic”

tr("Echo attribute's value"), "echo \"[% \"MY_FIELD\" %]\"", "", true)

tr("Run an application"), "ogr2ogr -f \"ESRI Shapefile\" \"[% \"OUTPUT_PATH\" %]\" \"[% \"INPUT_FILE\" %]\"", "", true);

Python“python”

Python, tr("Get feature id"), "QtGui.QMessageBox.information(None, \"Feature id\", \"feature id is [% $id %]\")", "", false);

Python, tr("Selected field's value (Identify features tool)"), "QtGui.QMessageBox.information(None, \"Current field's value\", \"[% $currentfield %]\")", "", false);

Python, tr("Clicked coordinates (Run feature actions tool)"), "QtGui.QMessageBox.information(None, \"Clicked coords\", \"layer: [% $layerid %]\\ncoords: ([% $clickx %],[% $clicky %])\")", "", false);

OpenUrl“open”

OpenUrl, tr("Open file"), "[% \"PATH\" %]", "", false);

OpenUrl, tr("Search on web based on attribute's value"), "http://www.google.com/search?q=[% \"ATTRIBUTE\" %]", "", false);

Action Keywords:

[%$clickx %],[% $clicky%]

$clickx, $clicky keywords will be automatically replaced by the mouse cursor position when a click event is triggered.

[% \"ATTRIBUTE\" %]

ATTRIBUTE keyword need to be the name of anattribute field, and the value of the attribute field of certain feature will be required by the action tool.

4.3.Add an Event Listener to the Map

Add an event listener that will execute the user function on certain signal from the map object. It is also possible to load the GABBS Maps API on demand.

This feature is currently under development.

5. GABBS Maps API Reference

GABBS Python Maps API gabbs.maps.			
No	API	Return Value	Usage
1.	gbsLoadLibrary()	void	Initialize GABBS maps library
2.	gbsUnloadLibrary()	void	Exit GABBS maps library
3.	MapContainer(dict canvasProp)	MapContainer	Create a Map Container, aka map canvas
4.	Map(dict mapProp)	Map	Create a Map layer
5.	Raster(dict rasterProp)	Raster	Create a Rasterlayer
6.	Vector(dict vectorProp)	Vector	Create a Vectorlayer
7.	DelimitedText(dict pointProp)	DelimitedText	Create a DelimitedText, CSV or other WKTlayer
8.	MapContainer.addLayer(Layer)	void	Add layer to map canvas
9.	MapContainer.removeLayer(Layer)	void	Remove layer to map canvas
10.	Layer.show()	void	Show layer
11.	Layer.hide()	void	Hide layer
12.	Layer.setProperty()	void	Update layer's properties and styles
13.	gbsGetSelectedAttributes()	List[]	Get all attributes in selected features of current layer
14.	gbsGetSelectedBounds()	QRectF	Get the bounding box's coordinates of all selected features of current layer