# Plugin Map

Use this wiki plugin to display a map on a wiki page with a wide range of customisation options.

See also Geolocation for how other Tiki objects, e.g. wiki pages, blog posts etc. can be geolocated.

## Historical overview of Maps integration in Tiki

Maps have been supported in Tiki since 2003 (which is why some call it a GeoCMS). There is geo-related info in various places (users, trackers, image galleries, articles, blog posts, etc.). This was originally done using MapServer, an active and powerful FLOSS mapping solution. However, it requires a dedicated server and more importantly, access to map data (which is not easy).

Later on, Google Maps arrived, providing an easy to use map integration to regular web sites, even without having to manage mapping data. Thus, Google Maps specific code was added to Tiki, which was convenient for a lot of people.

Then, after a community discussion, starting in Tiki7, OpenLayers (another option was Mapstraction) was added as a native way to handle maps, which permits the use of tiles from Google Maps, Bing Maps, OpenStreetMap (which is like Wikipedia but for maps), MapQuest (which serves OpenStreetMap maps), etc.

The Cartograf project further improved maps in Tiki8, Tiki9, Tiki10 and Tiki11. Many features were added, including Street View support.

In Tiki12, all Google Maps specific code was removed in favor of using OpenLayers, so Google Maps is accessible via the OpenLayers Google Layer. In addition the Natural Access project (no longer active) added further new capabilities to upload any existing line and polygon data as files and to be able to further customise how data objects were shown on the underlying map layer.

In Tiki15 all MapServer-specific code has been removed. Also OpenLayers 2.x continues to be used for the map layer and integration with Tiki to allow editable map objects to be overlaid on the map layer but experimentation has started with the integration of OpenLayers 3.x.

In Tiki20 integration with OpenLayers 3.x and higher was improved, and more features were exposed through the corresponding PluginMap parameters, as well as adding new tilesets, some of them using vector tiles instead of just the usual raster tiles.

Map-related documentation, as of summer of 2019, still makes reference to the different historical approaches but as it continues to be improved the older methods that are no longer used will be deprecated/removed. Volunteers to help with documentation improvement : please contact marclaporte at tiki dot org

## Parameters

Display a map
*Introduced in Tiki 1.*
Go to the source code
*Preferences required:* wikiplugin_map, feature_search

| Parameters | Accepted Values | Description | Default | Since |
|---|---|---|---|---|
| (body of plugin) | | Instructions to load content | | |

| | | | | |
|---|---|---|---|---|
| `center` | text | Format: `x,y,zoom` where `x` is the longitude, and `y` is the latitude. `zoom` is between `0` (view Earth) and `19`. | | 9.0 |
| `height` | digits | Height of the map in pixels | | 1 |
| `width` | digits | Width of the map in pixels | | 1 |
| `extents` | text | Extents | | 1 |
| `mapfile` | url | MapServer file identifier. Only fill this in if you are using MapServer. | | 1 |
| `scope` | text | Display the geolocated items represented in the page (`all`, `center`, or `custom` as a CSS selector). Default: `center` | center | 8.0 |
| `size` | digits | Size of the map | | 1 |
| `cluster` | digits | Distance between features before they are "clustered", 0 (off) to 100. (requires Open Layers v3+, default is 0) | 0 | 20.0 |
| `clusterFillColor` | text | Cluster fill color in RGB. (requires Open Layers v3+, default is 86, 134, 200) | 86, 134, 200 | 20.1 |
| `clusterTextColor` | text | Cluster text and outline color in RGB. (requires Open Layers v3+, default is 255, 255, 255) | 255, 255, 255 | 20.1 |

| | | | | |
|---|---|---|---|---|
| `popupstyle` | (blank)<br>bubble<br>dialog | Alter the way the information is displayed when objects are loaded on the map. | bubble | 10.0 |
| `tilesets` | text | Tilesets to use for background layers, comma separated. Tileset groups can be added separated by a tilde character (requires Open Layers v3+, default is the geo_tilesets preference) | 86, 134, 200 | 20.1 |
| `clusterHover` | (blank)<br>features<br>none | Appearance of clusters on mouse over. (requires Open Layers v3+, default is features) | features | 20.1 |
| `controls` | controls, layers, search_location, levels, current_location, scale, streetview, navigation, coordinates, overview<br>*separator:* `,` | Comma-separated list of map controls will be displayed on the map and around it | controls,layers,search_location | 9.0 |
| `library` | (blank)<br>ol2<br>ol3 | OL2 or OL3+ so far (default ol2) | ol2 | 20.1 |
| `tooltips` | (blank)<br>y<br>n | Show item name in a tooltip on hover | n | 12.1 |

### More detail for the `controls` display options

`controls` : basic pan and zoom control positioned left top of map
`layers` : clickable tool right top of map that opens a pop-up where the different map options can be

selected, e.g., OpenStreetMaps, Google Street, etc.
`search_location` : adds a 'Search Location' link beneath the map on the left, to allow a post code or address to be entered to search for a specific location
`levels` : displays zoom levels, positioned below the basic zoom control on the left of the map
`current_location` : adds a 'To My Location' link beneath the map on the left
`scale` : adds the map scale at the bottom left of the map
`streetview` : https://en.wikipedia.org/wiki/Google_Street_View
`navigation` : adds two icons below the zoom levels on the left of the map to select a grab tool to move the map and a box tool to zoom down to the selected box area
`coordinates` : displays the cursor coordinates at the top right of the map
`overview` : displays the current location in the context of the wider geographic area in a box at the bottom right of the map where the box can be toggled to be open/closed

### More detail for the `center` parameter

Note that this (x,y,z) order means longitude and latitude are the opposite way round to some map applications, such as Google Maps and Earth.

## Examples

### Simple Example

This map shows all the tracker items from tracker #10. In its simplest form, the only thing a tracker item needs to appear on a map is a Location field with the "Use as item location" option set to "yes".

```
{MAP(controls="controls,layers,levels,navigation,scale,coordinates" width="350" height="350" center="-1.305,51.487,8" tooltips="y")} {searchlayer tracker_id="10"} {MAP}
```

### Slightly less simple real world example

This one combines the items from three trackers, #10 (for the points), #11 (paths and trails) and #12 (boundaries and zones. They each have the following fields:

- "theRidgewayMap" this is from type name, it contains a useful name for the displayed item
- "publishBoundaryZoneSeparately" is from type checkbox
- "publishPathsSeparately" also type checkbox
- "publishPOISeparately" also type checkbox

Another version: Use a "file"-field in the tracker, check the option "Index as MapLayer" is set to "geojson" and upload a geojson-file, created by http://geojson.io/ .

*This code:*

```
{MAP(controls="controls,layers,levels,navigation,scale,coordinates" width="630" height="500" center="-1.305,51.487,9" popupstyle="bubble" tooltips="y")} {searchlayer tracker_field_theRidgewayMap="y" tracker_field_publishBoundaryZoneSeparately="yes" maxRecords="200" popup_height="300" popup_width="480"} {searchlayer tracker_field_theRidgewayMap="y" tracker_field_publishPathsSeparately="yes" maxRecords="200" load_delay="2" popup_height="300" popup_width="300"} {searchlayer tracker_field_theRidgewayMap="y" tracker_field_publishPOISeparately="yes" maxRecords="200" load_delay="3" popup_height="300" popup_width="360"} {MAP}
```

*Would produce on this site:*

*Note: load_delay help with the order (what should be display in the back and what should be in the front) of the layers.*

You'll need several trackers to store the information you'll need (Boundaries, Point Of Interest and/or Path) with a few fields.
*Note: I used the information below from a different simpler project.*

- Boundaries: A name field, a files field (option index as maplayer GeoJSON) and a yes/no field (so the plugin know you want it displayed)

- POI (Point Of Interest): A name field, an icon, a field a location field and a yes/no field (so the plugin know you want it displayed)
- Path: A name field, a files field (option index as maplayer GeoJSON) and a yes/no field (so the plugin know you want it displayed)

---

{MAP(scope="Center" controls="yourselection" center="yourcoord,yourzoompref")} {searchlayer tracker_field_theRidgewayMap="y"} {searchlayer tracker_field_theRidgewayMap="y"} {searchlayer tracker_field_theRidgewayMap="y"} {MAP}

### Using `{searchlayer}`

*Since Tiki9*
The `{searchlayer}` function allows objects to be displayed on maps and for logic to be applied for when and how they are displayed. It should be noted however that this is a 'search' function so only objects that have been properly indexed with Unified Search can be 'worked on'.

Individual `{searchlayer}` items are put in the body of the map plugin where the generic format is:

---

{MAP(mapfile="" extents="" size="" width="" height="" etc)} {searchlayer parameter1="xx" parameter2="yy" layer="foo" etc } {searchlayer parameter1="aa" parameter2="bb" layer="bar" etc } etc {MAP}

### More detail on the {searchlayer} parameters:

`suffix` : will add a value from the incoming items as a suffix to the layer name, essentially creating multiple layers from a single search query. For example, in the CartoGraf demo case, it fetches the elements from the other students in the same workgroup and creates one layer per student, so other's results can be hidden or displayed.
`refresh` : is the refresh delay in seconds after which the query will be re-executed to update the map.
`fields` is the list of fields to take along with the results. The unified index indexes more data than the index returns with the results by default. This allows to provide a list of fields to be fetched before sending the result.
`maxRecords` : (Tiki 11+) Search results are **limited by the "Maximum number of records in listings"** setting (Look & Feel => Pagination) maxRecords bypass this setting and indicates how many results will be taken from the query. Be aware that it is important to give a reasonable value to this to make sure errors in a query won't take down your server and still the results you need to be displayed.
`tracker_field_XXXXX` : where XXXXX is the tracker field permanent name, allows conditional logic to be applied to the overall searchlayer request. If multiple tracker_field parameters are used then AND logic applies, e.g., if both `tracker_field_XXXX="y"` and `tracker_field_YYYY="y"` then both conditions will need to be true for the searchlayer action to be carried out
`sort_mode` : (Tiki 11+) order for the search results (e.g., `tracker_id_desc` )
`load_delay` : (Tiki 12+) delay in seconds to wait before loading the layer (can help in ordering layers with features loaded from files)
`popup_width` : (Tiki 12.2+) width of popups for features on this layer - plain numeric xx for pixels or xx% for percentage (percentage size only on dialog popups)
`popup_height` : (Tiki 12.2+) height of popups
`layer` : name of the layer
`tracker_status` : filtering results based on tracker status (ie: tracker_status="o")

### Layers usage example 1

---

```
{MAP()} {searchlayer tracker_field_fieldname="something" refresh="3600" maxRecords="50"
fields="tracker_field_fieldname01,tracker_field_filedname02"} {MAP}
```

```
{MAP()} {colorpicker colors="ff0,69c,c96,6c9,ffffff,black,f0f,0ff,96c"} {MAP}
```

### Layers usage example 2

Imagine that you want to allow an external web site (blogspot, or anywhere where iframes are accepted) to show a map generated in a tiki site, in a similar way to how you would add a google map. You could embed the external wiki page with the geolocated tracker items in a map through an iframe at the remote site.

This type content below added to the wiki page will display a map in the page without showing the list of items (from tracker 16 in this example) below the map. And since the PluginAppframe is also used, no side columns will be displayed either, so that you could make an iframe with just the map of the page:

```
{MAP(scope="center" width="600" height="400" center="1.7282503,41.2257581,14")}
{searchlayer geo_located="y" maxRecords="100" tracker_id="16" type="trackeritem"} {MAP}
{appframe min="0" hideleft="y" hideright="y" fullpage="n" absolute="n"}
```

Then, if the wiki page was called "MyMap", then the url to be used in the iframe could be this one: http://example.com/tiki-index_raw_p.php?page=MyPage

### Creating map objects with the File option

Map objects can be defined in Tracker records: a simple POI can be defined using the Icon field type and the CartoGraf project then introduced the Geographic feature Tracker Field that allows paths (LineString) and boundaries (Polygon) to be drawn on a map and saved.

Whilst the 'Geographic feature Tracker Field' method is very powerful it does have limitation so in 12.1 a new "Index As Map Layer" option (defaults to No) was added to the Files Tracker Field for an uploaded file (scroll to the end of the Options list in the tracker field set up screen to find this new one). This allows a file to be uploaded that is is then indexed and can therefore be added to a map. The file does have to very carefully conform to a standard XML-like format and a drop down list allows the selection of the file format to be either geoJSON or GPX - however the map projection must (at present) be EPSG:4326. This new capability overcomes the previous limitations with the Geographic feature Tracker Field. A tracker should either have the Files field or the Geographic feature Tracker Field. Note you may have to use "load_delay" to tweak the order of the "searchlayer" to display the elements that will be on the foreground and those on the background (IE: POI should be displayed over opaque or color filled boundaries).

Using this File method any of the standard OpenLayer object types, LineString, MultiLineString, Polygon, MultPolygon, etc. can be used.

### Worked example with map objects

*Since Tiki12*
This example displays a zoomed in section of the same map as the the 'Simple worked example' above, but uses 'searchlayer' logic to display different types of map objects (POIs, paths and boundaries) where POIs have been defined using the 'Icon' field and paths and boundaries have both been defined using the File field method described above.

Some detailed notes on how the above worked example is created

The following is the 'code' used:

```
{MAP(controls="controls,layers,levels,navigation,scale,coordinates", center="-1.305,51.487,9",
width="630" height="500" tooltips="y" )} {searchlayer tracker_field_theRidgewayMap="y"
tracker_field_publishBoundaryZoneSeparately="yes" maxRecords="200" popup_height="300"
popup_width="480" } {searchlayer tracker_field_theRidgewayMap="y"
tracker_field_publishPathsSeparately="yes" maxRecords="200" load_delay="2" popup_height="300"
popup_width="300"} {searchlayer tracker_field_theRidgewayMap="y"
tracker_field_publishPOISeparately="yes" maxRecords="200" load_delay="3" popup_height="300"
popup_width="360"} {MAP}
```

In addition the following custom javascript is used to 'adjust' how OpenLayers displays the path 'lines'

```
/* * this section of the custom js amends the way that OpenLayers 'spaces out' dot and dot/dash
markings for paths etc */ OpenLayers.Renderer.SVG.prototype.dashStyle = function(style,
widthFactor) { var w = style.strokeWidth * widthFactor; var str = style.strokeDashstyle; switch (str)
{ case 'solid': return 'none'; case 'dot': return [1, 4 * w].join(); case 'dash': return [4 * w, 2 * w].join();
case 'dashdot': return [4 * w, 2 * w, 1, 2 * w].join(); case 'longdash': return [8 * w, 2 * w].join(); case
'longdashdot': return [8 * w, 2 * w, 1, 2 * w].join(); default: return
OpenLayers.String.trim(str).replace(/\s+/g, ","); } }
```

Also:

- The tracker field "theRidgewayMap" allows the same tracker mapobjects to be displayed on a variety of different maps (or not)

- The tracker fields: "publishBoundaryZoneSeparately", "publishPathsSeparately", and "publishPOISeparately" - allow individual map objects to be displayed (or not) by changing their value in the appropriate tracker
- The 'popup_height' and 'popup_width' parameters allow the pop-ups for individual map objects to be resized dependent upon what tracker field data is being displayed (set in the main tracker parameters), and
- More customisation of the pop-up box can be achieved than is shown in this example by creating custom .tpl files:

`/templates/styles/yourstyle/object/infobox.tpl`
and
`/templates/styles/yourstyle/object/infobox/trackeritem.tpl`

### Some other tips

### Globally change stroke width

*For instance*

```
$(document).ready(function () { $(".map-container").on('initialized', function () { $(".map-container")[0].defaultStyleMap.styles.default.context.getStrokeWidth = function () { return 4; }; });
});
```

### Hide location field when location is not set

In some case you may have some items with a map location set and some items without.
Those without will display a map and you may prefer this not to happen.
You can hide the field using a Javascript plugin and some code on the page:

**hide location field when location is not set**

```
{JQ()} var map = $('input[name="location"]').val(); if (map == "0,0,2") { $('#openlayers1').hide(); }
{JQ}
```

### New OpenLayers Library Support (Tiki 20+)

In Tiki 20.1 support for newer OpenLayers versions has been added, now updated to ol5. A very experimental version was introduced in earlier Tikis but it wasn't reliable or usable. There is a preference for this on the maps control panel.

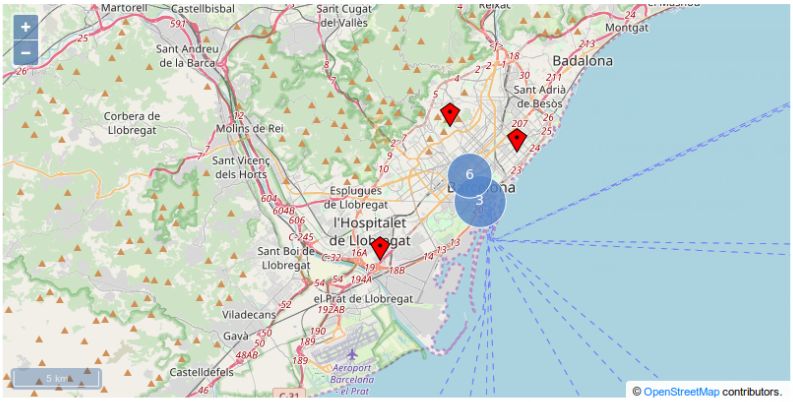
This introduced:

- new parameters `library` and `cluster` which can be seen:
  - in this page: PluginMap ol3 Demo
  - applying this profile in your tiki site: GeoCMS_Maps_18
- new `tilesets` allowed, coming from:
  - Stamen (no api key required). Tileset param names:
    - `tilesets="stamen_watercolor"`
    - `tilesets="stamen_terrain"`
    - `tilesets="stamen_toner-hybrid"`
      (see the full list at http://maps.stamen.com )
  - Nextzen (Vector Tiles, & free api key required): https://www.nextzen.org
    - `tilesets="nextzen"`

# All geolocated content



## 1.1. Map




Search Location

## Related pages

- PluginList
- Maps
- Geolocation
- PluginGoogleMap

## Aliases

Plugin Map | PluginMaps | Plugin Maps