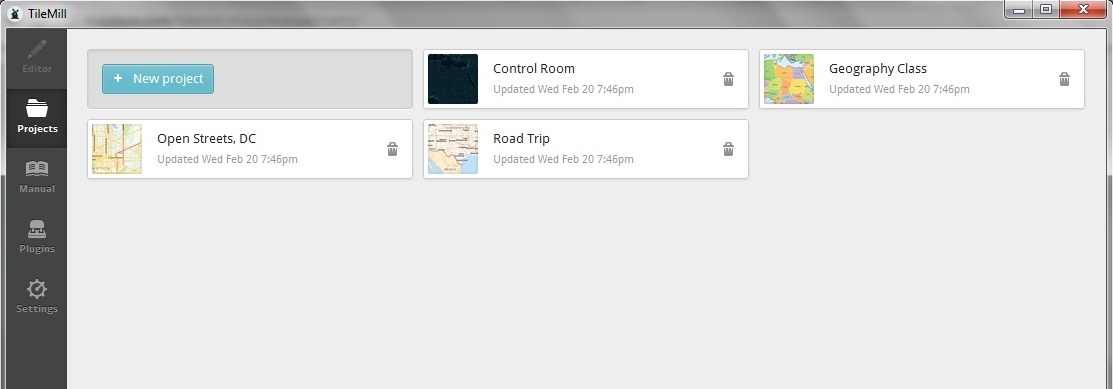
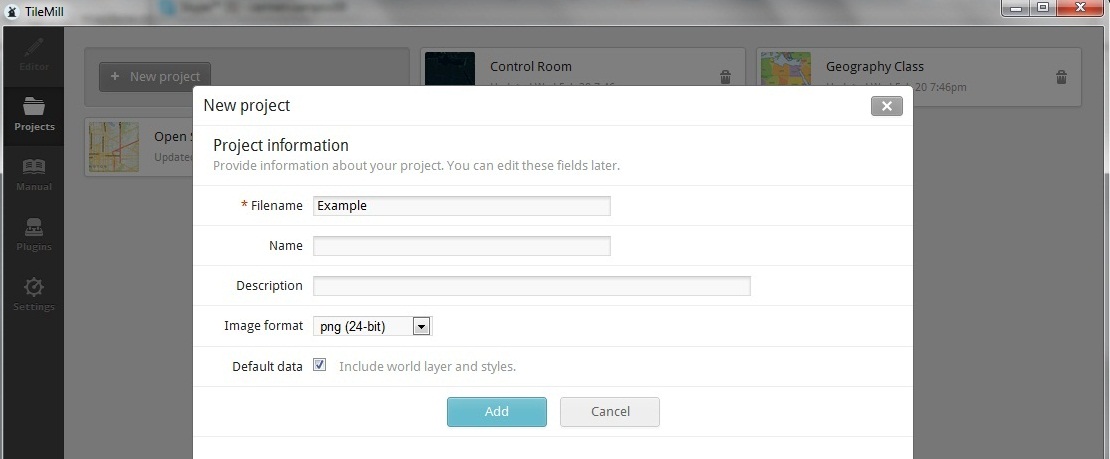
## TileMill

Using TileMill is easy once you have done your first project because everything is intuitive. Once we have open the program, to start creating a map, we have to press the button “New project”.

We need to provide some information about the project. The only require data is the name of the file. But we can also add a name for the project, a description and the image format.

There is an option “Default data” to indicate that we want an initial world layer to appear in our project with an style:

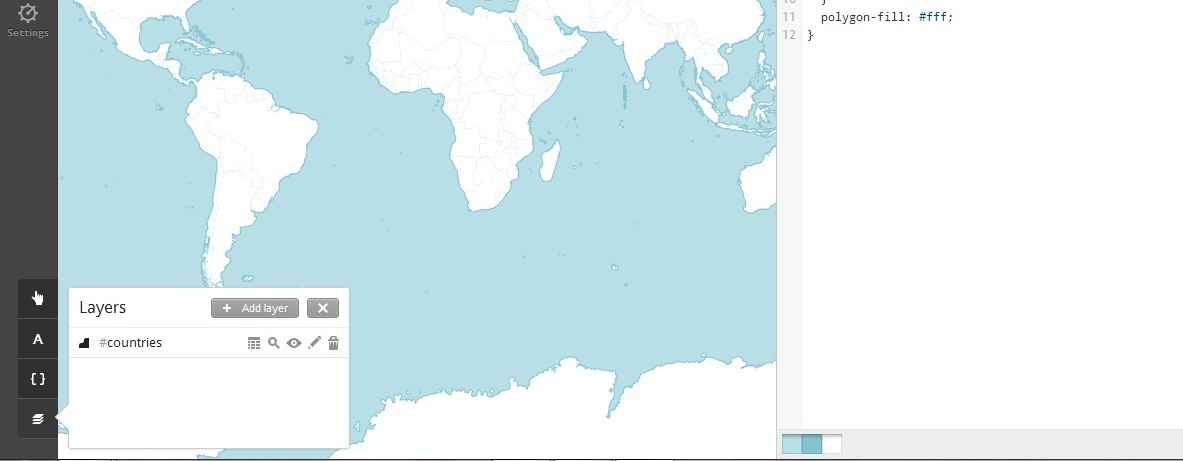
* background-color: #b8dee6; background color.
* line-color: #85c5d3; color of the line that separates the countries.
* line-width: 2; width of the line that separate the countries.
* line-join: round; kind of line that separate the countries.
* polygon-fill: #fff; color to fill the countries.

Then we press “Add”.

It returns to the main page in TileMill, and we can observe that our project has been created there. To open it, we have to click in its image.

### Layers

To begin with, we should add a layer. To do that, we have to select the appropriate image down on the left and press the button “Add layer”.

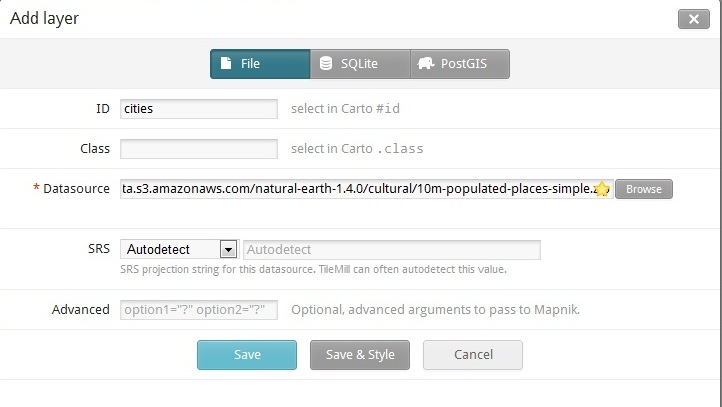


As we can see, there are other options for every layer:

1. To see the features of the layer. This features can be used, for example in this case, to consider only a specific country, or a group of them which satisfy a concrete property; to show the value of a concrete feature in the map.
2. To zoom in the map.
3. To not consider the layer. It is like if we delete the layer. But it is still there, in case we need to consider it again, we do not have to import it.
4. To edit the layer.
5. To delete the layer.

When we add a layer, we have to indicate the datasource address where the file is located; it could be a file in our computer or be located in MapBox. When can also select the file as “favorite”, so when we do another project, we do not need to look for the file.

We could assign an ID and a class for the layer, so we can apply a css style to the layer. The ID is necessary, so in the we do not choose any, one will be automatically assigned.

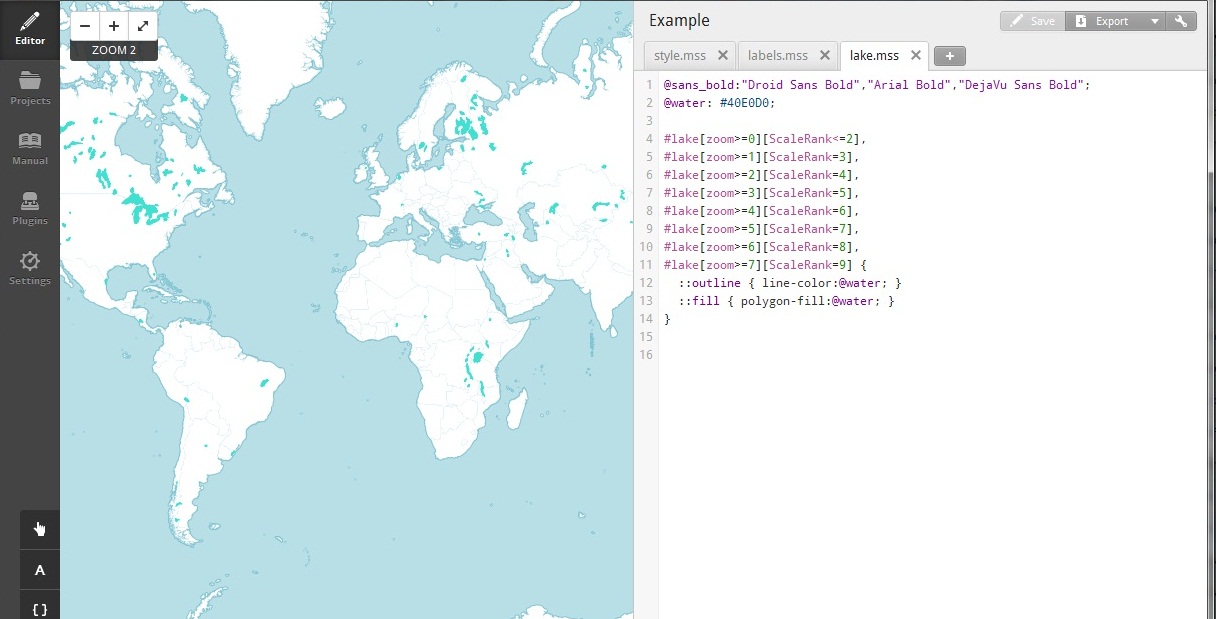


We just have to look for the different layers we want to add. Some options are:

* http://mapbox-geodata.s3.amazonaws.com/natural-earth-1.4.0/cultural/10m-populated-places-simple.zip show the different cities.
* http://mapbox-geodata.s3.amazonaws.com/natural-earth-1.4.0/cultural/10m-admin-0-country-points.zip show the names of the countries.

We could add the lakes in the world, which datasource is http://mapbox-geodata.s3.amazonaws.com/natural-earth-1.3.0/physical/10m-lakes.zip, with the ID lake and apply a style:

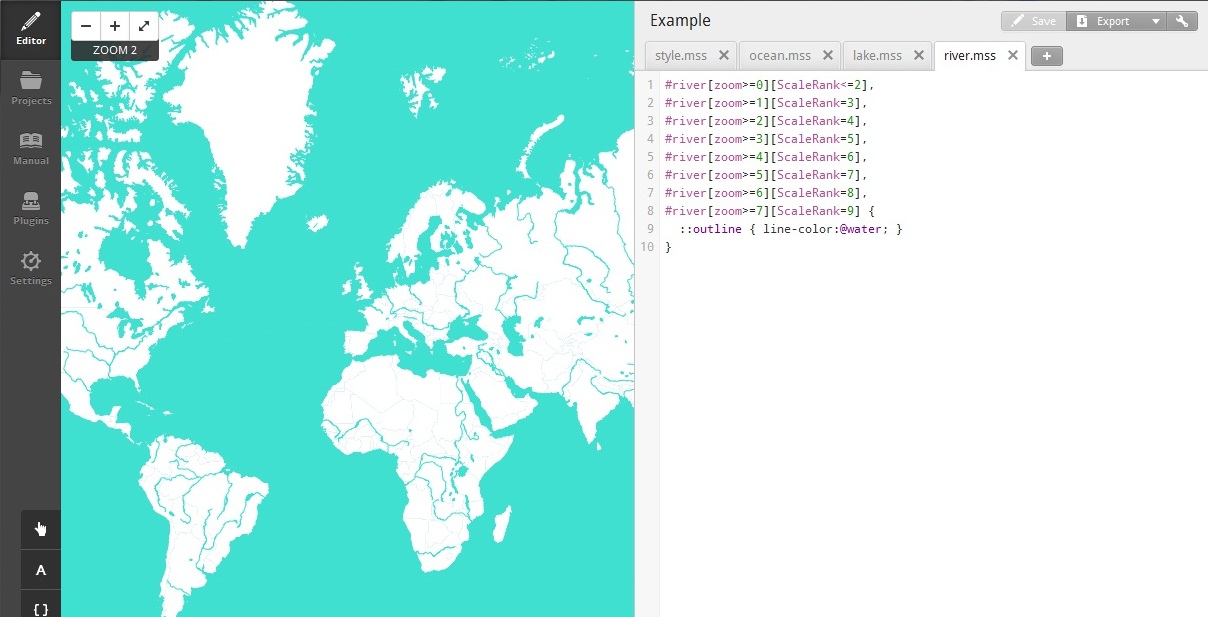
* @sans\_bold:"Droid Sans Bold","Arial Bold","DejaVu Sans Bold"; select the kind of letter, we can only put a name, or several, in case one of the options is not available.
* @water: #40E0D0; we can define a name for an specific color, in case we want to use the same several times. Of course, we could also write the code of the color every time
* #lake[zoom>=0][ScaleRank<=2], depending on the zoom we are applying, we will only show determinate lakes. We can see the ScaleRank of every element in the layers image.
* line-color: @water; line of the lake color
* polygon-fill: @water; color inside the line



We could also add the oceans and the rivers, which are respectively in:

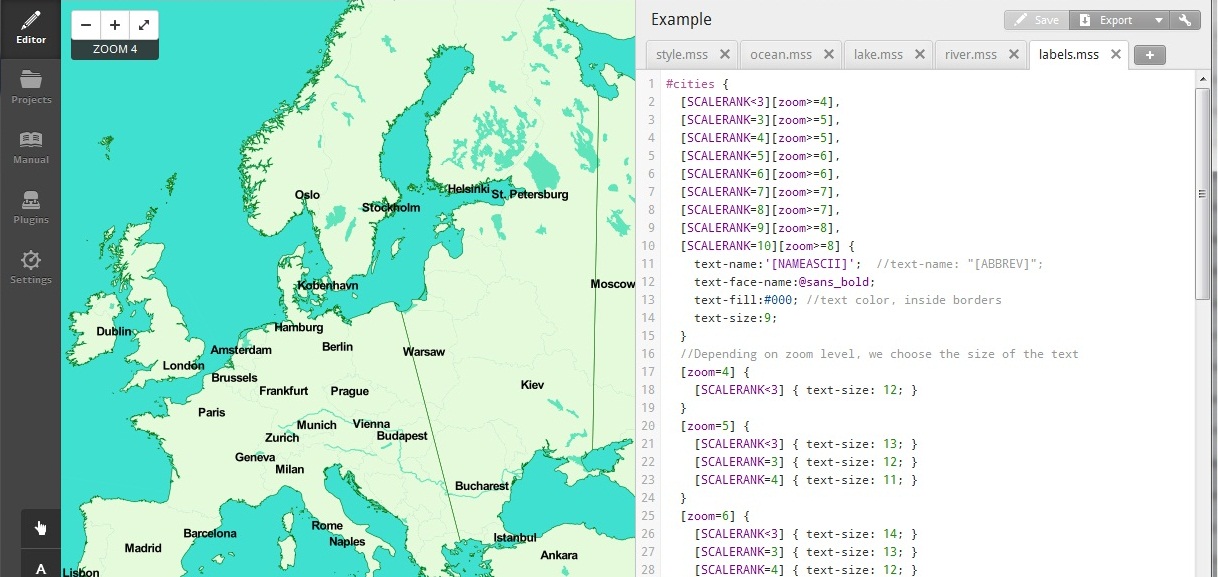
http://mapbox-geodata.s3.amazonaws.com/natural-earth-1.3.0/physical/10m-ocean.zip

http://mapbox-geodata.s3.amazonaws.com/natural-earth-1.3.0/physical/10m-rivers-lake-centerlines.zip

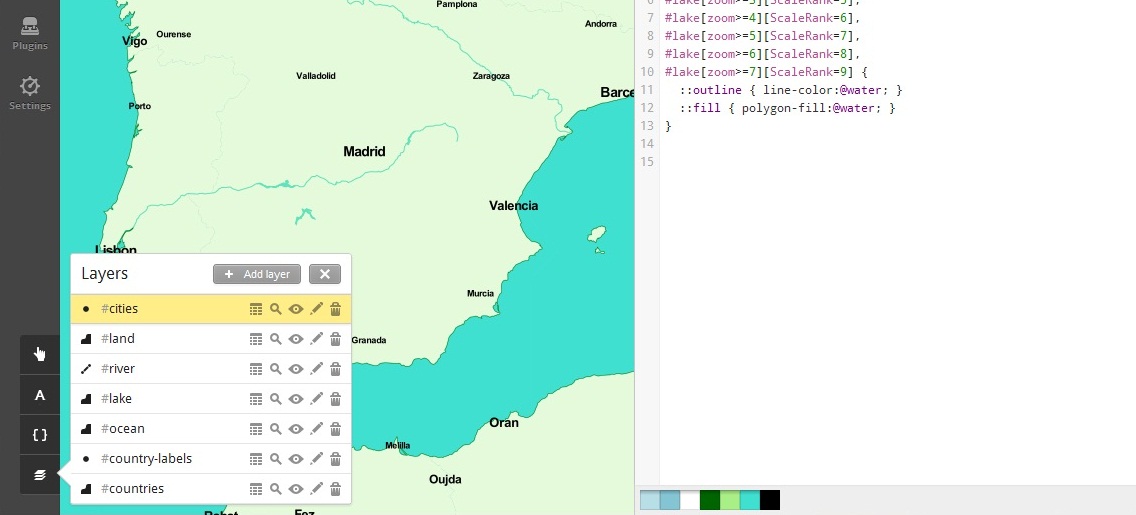


In order to show the names of the cities, we need to add this layer http://mapbox-geodata.s3.amazonaws.com/natural-earth-1.4.0/cultural/10m-admin-0-country-points.zip

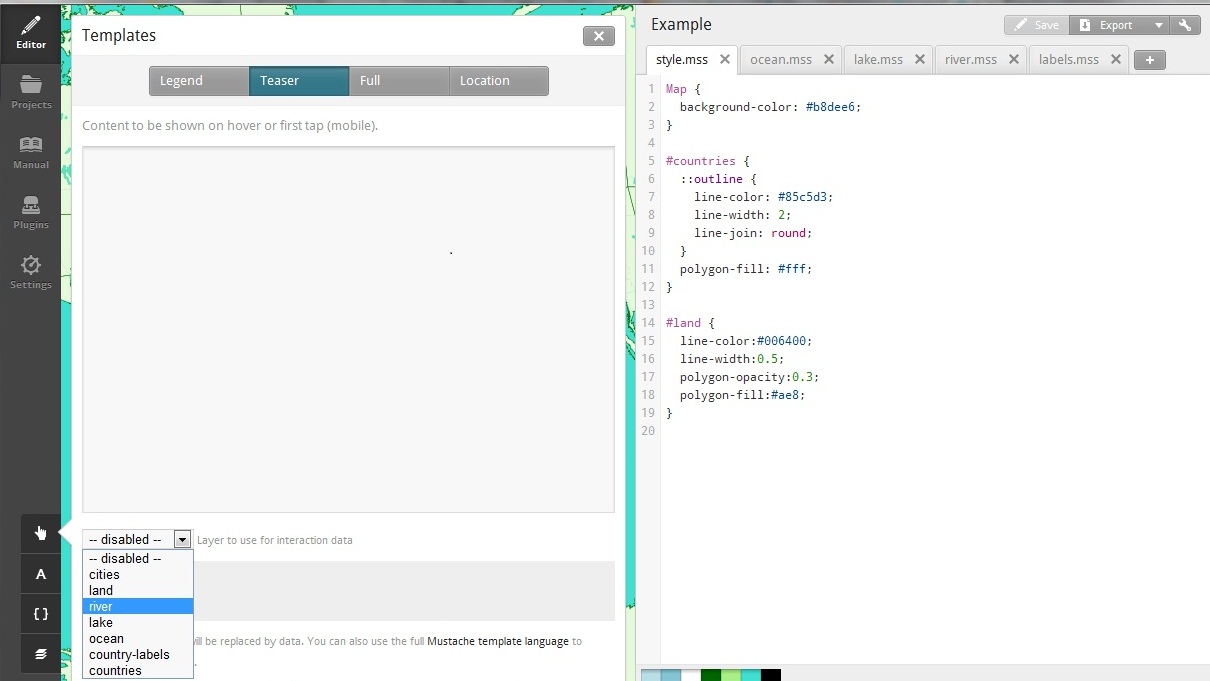
* text-name:'[NAMEASCII]'; the name of the city, we could have used NAME, NAME\_ALT, LS\_NAME… In every particular case, we can choose one or another, and we can consult what we are going to show in the features of the layer.
* text-face-name:@sans\_bold; type of letter.
* text-fill:#000; text color inside borders.
* text-size:9; size of the letter.
* [zoom=10] {
* [SCALERANK<3] { text-size: 16; text-character-spacing:2; }
* [SCALERANK=3] { text-size: 16; text-character-spacing:2; }
* [SCALERANK=4] { text-size: 15; text-character-spacing:1; }
* [SCALERANK=5] { text-size: 15; text-character-spacing:1; }
* [SCALERANK=6] { text-size: 15; text-character-spacing:1; }
* [SCALERANK=7] { text-size: 14; }
* [SCALERANK=8] { text-size: 14; }
* [SCALERANK=9] { text-size: 13; }
* [SCALERANK=10] { text-size: 13; }
* } depending on zoom level, we choose the size of the text and the space between characters



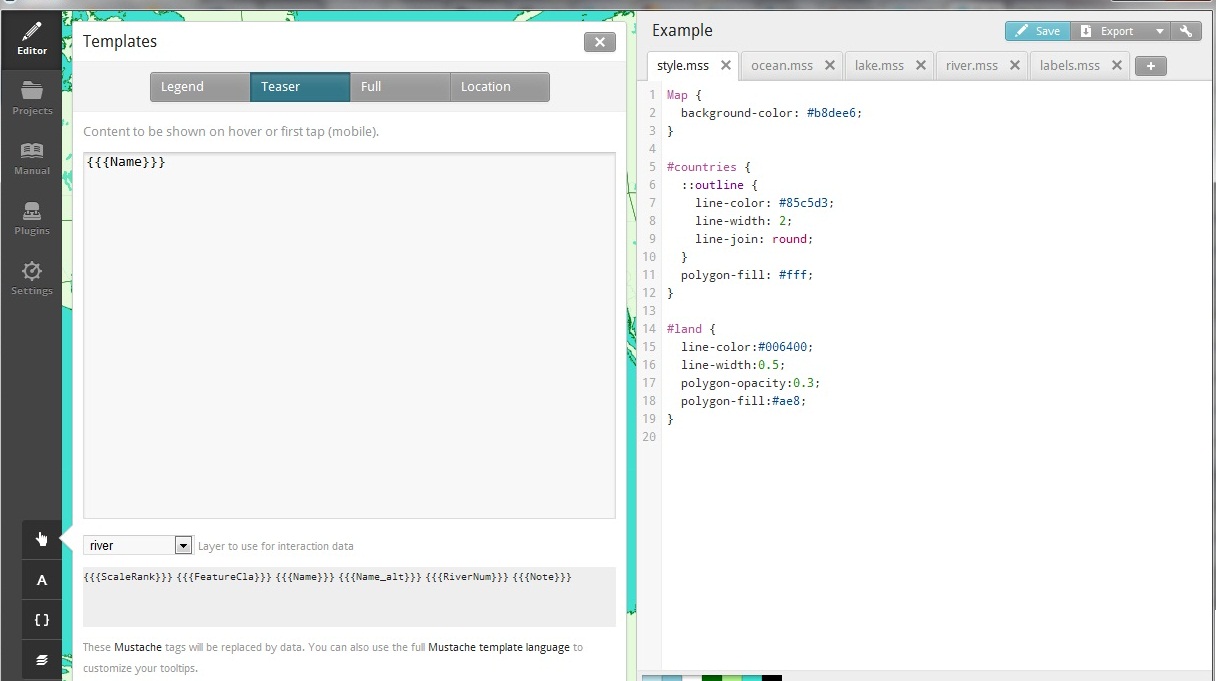
If we want a layer to be on top of another, for example the names, so they are not half cover, we just have to select it and move it to the position where we want it.



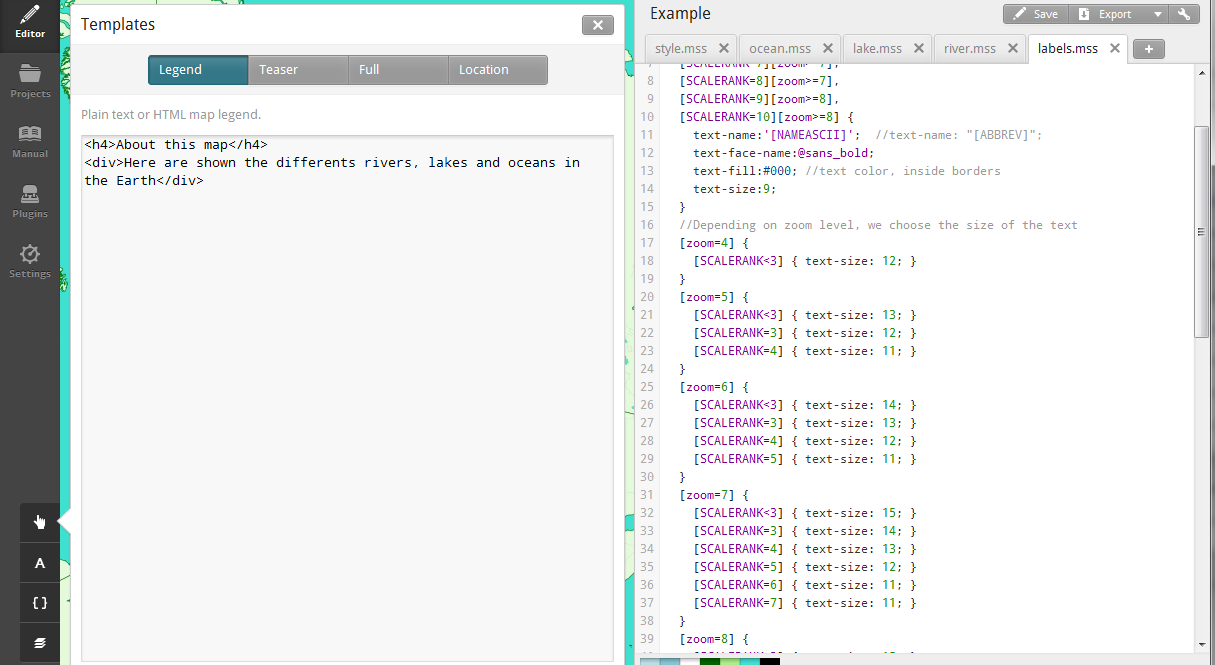
To add a teaser, we press the image and we choose “teaser” in the templates. In the combox, we have to choose for what layer we want to do the teaser.



Once we select one, the different attributes we can use appear in the second text area. And in the first one, we can write the text we want to show in the map when the mouse rolls a concrete element. We can show one or more element, and a normal sentence.

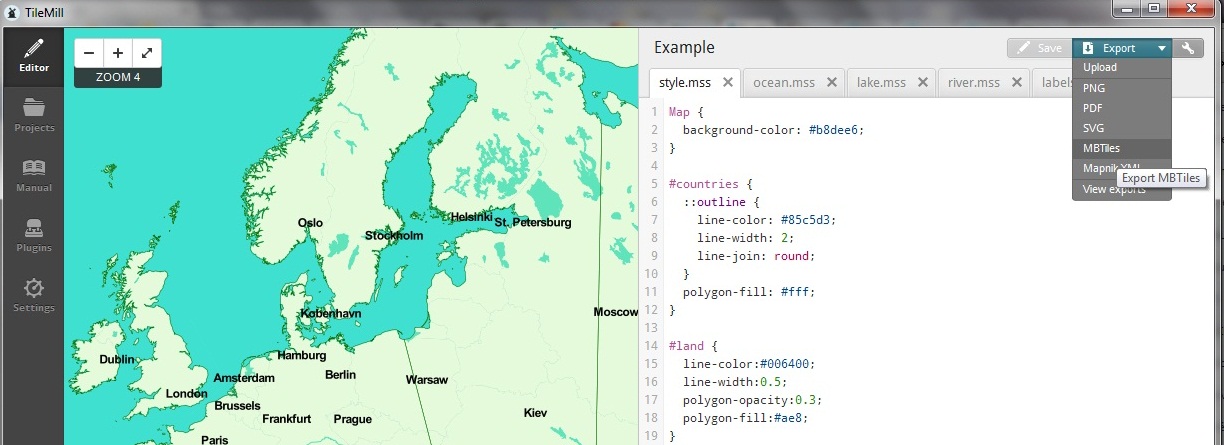


To create a legend for the map, we press the image and we choose “legend” in the templates. We can add normal text or html formatted text. Te legend will appear down in the right.

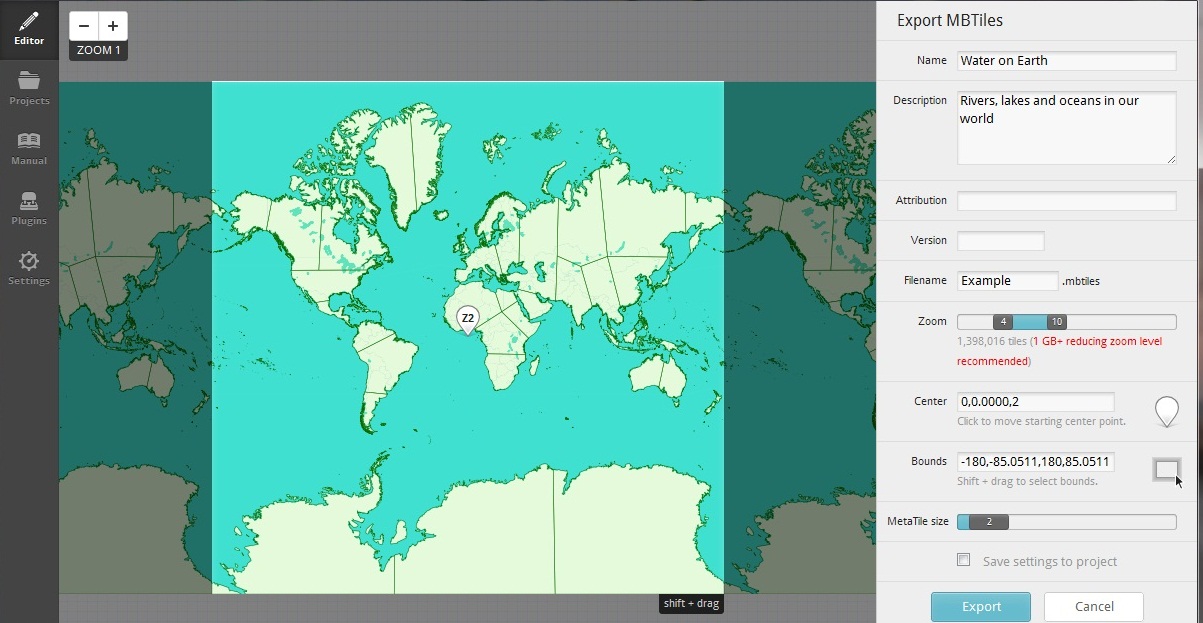


### Export as MBTiles

To export the map, we press the button and select the kind of file we want to get. In this case, MBTiles.



We can choose a name, a description, the attribution and the version. And the filename is the one we set in the beginning. An important thing here is the zoom. By default, the zoom is the complete range of values, which could take years in generate the map. We can change the minimum and the maximum, and we can observe how many tiles will be generated and how much space will be needed. We can also change the part of the map we want to see when we open it. And then we press “Export”.

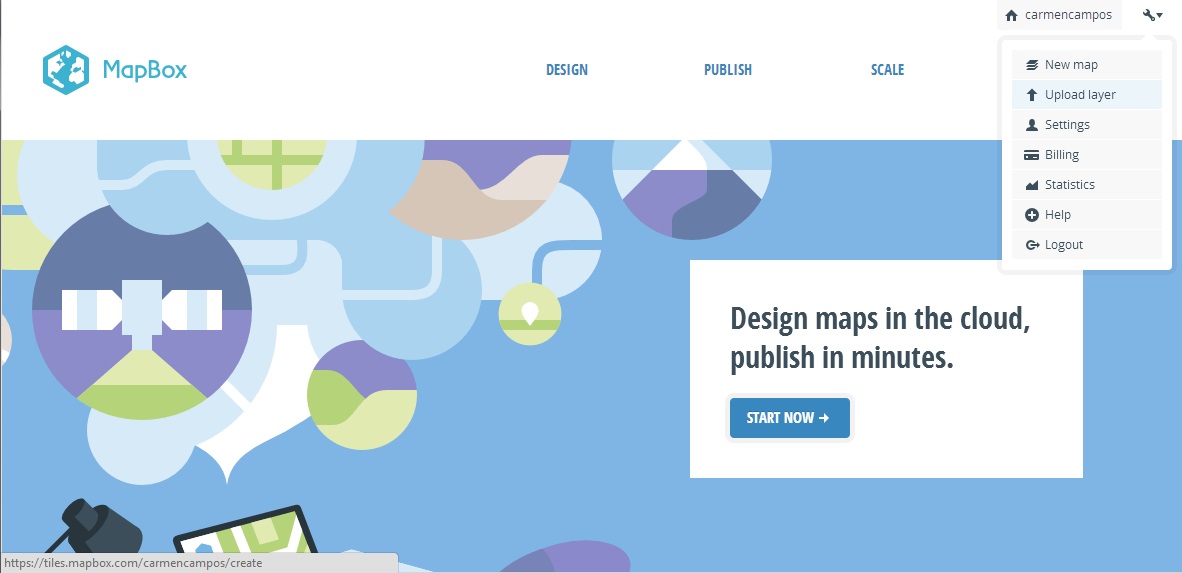


Once it is finish, we press the button “Save MBTiles”.

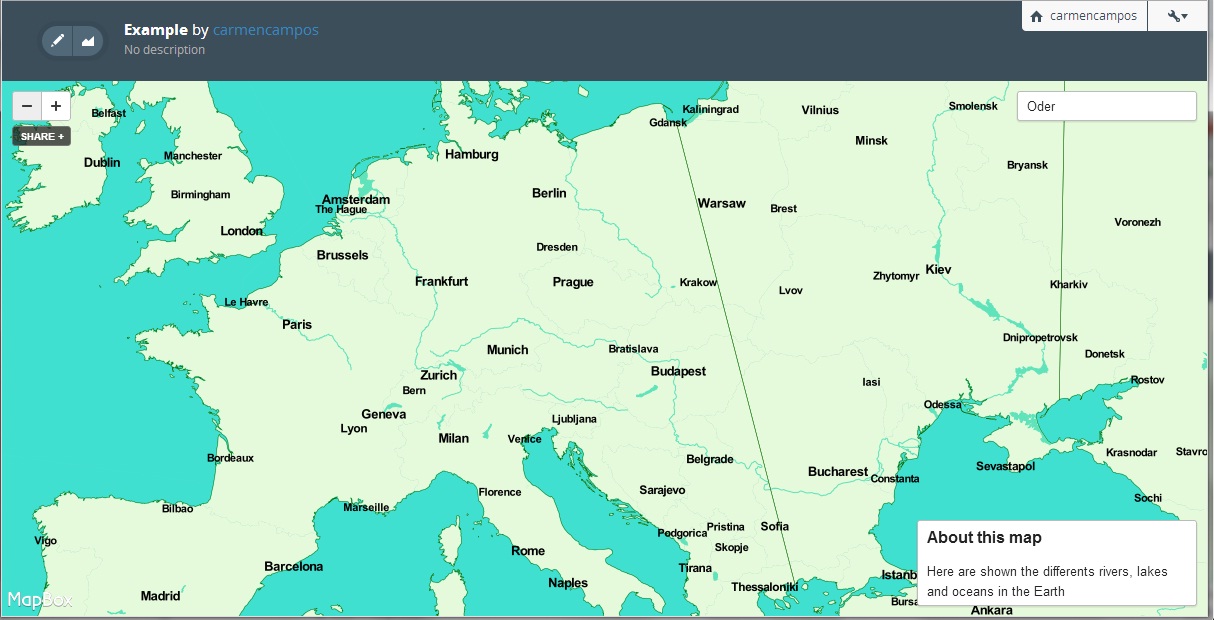
## Publish to Mapbox

To publish our map in Mapbox (http://mapbox.com/) we need to create an account and to log in.

We have to press the tool image and select “Update layer” option.



Then we just select a file in our computer and press “Upload file” button. The file size must be less than 5GB, but also less than the capacity we have available in our account.



## Publish map using leaflet

1. We need to have a server installed in our system.
2. We create a folder to work with leaflet in the appropriate folder in the server, the one where the server accesses to show the files.
3. We add the server code to read MBTiles to this folder.
4. We add our index.html page to this folder.
5. We add our {name-of-the-map}.mbtiles to this folde.
6. We run the server.
7. We go to localhost/{name-of-the-folder}

Our index.html page would be like this:

<!DOCTYPE html>

<html>

<head>

<title>Leaflet PHP/MBTiles Example</title>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- file to use leaflet -->

<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.4.5/leaflet.css" />

<!--[if lte IE 8]><link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.4.5/leaflet.ie.css" /><![endif]-->

</head>

<body>

<!-- space where we are going to place the map -->

<div id="map" style="width: 600px; height: 400px"></div>

<!-- file to use leaflet -->

<script src="http://cdn.leafletjs.com/leaflet-0.4.5/leaflet.js"></script>

<script>

var map;

var mapquestOSM = new L.TileLayer("http://{s}.mqcdn.com/tiles/1.0.0/osm/{z}/{x}/{y}.png", {

maxZoom: 17,

subdomains: ["otile1", "otile2", "otile3", "otile4"],

});

var mapquestOAM = new L.TileLayer("http://{s}.mqcdn.com/tiles/1.0.0/sat/{z}/{x}/{y}.jpg", {

maxZoom: 17,

subdomains: ["oatile1", "oatile2", "oatile3", "oatile4"],

});

// this part is to show the layer using the map hosted in mapbox

// http://{s}.tiles.mapbox.com/v3/carmencampos.example/{z}/{x}/{y}.png is the url where our map is hosted in mapbox

// in general would be http://{s}.tiles.mapbox.com/v3/{user}.{name-of-the-map}/{z}/{x}/{y}.png

var hostedTiles = new L.tileLayer('http://{s}.tiles.mapbox.com/v3/carmencampos.example/{z}/{x}/{y}.png', {

//indicate that it is not show by default

tms: false,

subdomains: ["a", "b", "c", "d"],

opacity: 0.7

});

// this part is to show the layer using the local map

// mbtiles.php?db=example.mbtiles&z={z}&x={x}&y={y} is the local url

// in general would be {name-of-the-file-to-extract-the-map}.php?db={name-of-the-map}.mbtiles&z={z}&x={x}&y={y}

var mbTiles = new L.tileLayer('mbtiles.php?db=example.mbtiles&z={z}&x={x}&y={y}', {

//indicate that it is show by default

tms: true,

opacity: 0.7

});

map = new L.Map("map",{

layers: [mapquestOSM, mbTiles]

}).fitWorld();

// layers which are going to be always in the map

var baseLayers = {

"MapQuest Streets": mapquestOSM,

"MapQuest Aerial": mapquestOAM

};

// we can decide when we want to show the layers

var overlays = {

"Local MBTiles File": mbTiles,

"Hosted Tiles": hostedTiles

};

layersControl = new L.Control.Layers(baseLayers, overlays, {

collapsed: false

});

map.addControl(layersControl);

</script>

</body>

</html>

Our mbtiles.php file to extract .mbtiles would be like this:

<?php

$zoom = $\_GET['z'];

$column = $\_GET['x'];

$row = $\_GET['y'];

$db = $\_GET['db'];

try

{

// Open the database

$conn = new PDO("sqlite:$db");

// Query the tiles view and echo out the returned image

$sql = "SELECT \* FROM tiles WHERE zoom\_level = $zoom AND tile\_column = $column AND tile\_row = $row";

$q = $conn->prepare($sql);

$q->execute();

$q->bindColumn(1, $zoom\_level);

$q->bindColumn(2, $tile\_column);

$q->bindColumn(3, $tile\_row);

$q->bindColumn(4, $tile\_data, PDO::PARAM\_LOB);

while($q->fetch())

{

header("Content-Type: image/png");

echo $tile\_data;

}

}

catch(PDOException $e)

{

print 'Exception : '.$e->getMessage();

}

?>

## Publish map using openlayers

1. We need to have a server installed in our system.
2. We create a folder to work with leaflet in the appropriate folder in the server, the one where the server accesses to show the files.
3. We add the server code to read MBTiles to this folder.
4. We add our index.html page to this folder.
5. We add our {name-of-the-map}.mbtiles to this folde.
6. We run the server.
7. We go to localhost/{name-of-the-folder}

Our index.html page would be like this:

<html>

<head>

<title>OpenLayers PHP/MBTiles Example</title>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- file to use openlayers -->

<link rel="stylesheet" href="http://openlayers.org/dev/theme/default/style.css" type="text/css">

<style>

.olControlAttribution {

font-size: smaller;

right: 2px;

bottom: 2px;

position: absolute;

display: block;

}

</style>

</head>

<body>

<!-- space where we are going to place the map -->

<div id="map" style="width: 600px; height: 400px"></div>

<!-- file to use openlayers -->

<script src="http://openlayers.org/dev/OpenLayers.js"></script>

<script>

var map;

var mapquestOSM = new OpenLayers.Layer.XYZ("MapQuest Streets", ["http://otile1.mqcdn.com/tiles/1.0.0/osm/${z}/${x}/${y}.png", "http://otile2.mqcdn.com/tiles/1.0.0/osm/${z}/${x}/${y}.png", "http://otile3.mqcdn.com/tiles/1.0.0/osm/${z}/${x}/${y}.png", "http://otile4.mqcdn.com/tiles/1.0.0/osm/${z}/${x}/${y}.png"], {

transitionEffect: "resize"

});

// // this part is to show the layer using the map hosted in mapbox

// http://a.tiles.mapbox.com/v3/carmencampos.example/${z}/${x}/${y}.png is the url where our map is hosted in mapbox

// in general would behttp://a.tiles.mapbox.com/v3/{user}.{name-of-the-map}/${z}/${x}/${y}.png

var hostedTiles = new OpenLayers.Layer.XYZ("Hosted Tiles", ["http://a.tiles.mapbox.com/v3/carmencampos.example/${z}/${x}/${y}.png", "http://b.tiles.mapbox.com/v3/carmencampos.example/${z}/${x}/${y}.png", "http://c.tiles.mapbox.com/v3/carmencampos.example/${z}/${x}/${y}.png", "http://d.tiles.mapbox.com/v3/carmencampos.example/${z}/${x}/${y}.png"], {

transitionEffect: "resize",

isBaseLayer: false,

opacity: 0.7,

visibility: true

});

// create TMS layer using MBTiles sqlite database

// this part is to show the layer using the local map

// mbtiles.php is the name of the file we use to extract our map from the .mbtiles file

var mbTiles = new OpenLayers.Layer.TMS("Local MBTiles File", "mbtiles.php", {

getURL: mbtilesURL,

transitionEffect: "resize",

isBaseLayer: false,

opacity: 0.7

});

// See: http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection

// example.mbtiles name of the file where our map is stored

function mbtilesURL (bounds) {

var db = "example.mbtiles";

var res = this.map.getResolution();

var x = Math.round ((bounds.left - this.maxExtent.left) / (res \* this.tileSize.w));

var y = Math.round ((this.maxExtent.top - bounds.top) / (res \* this.tileSize.h));

var z = this.map.getZoom();

return this.url+"?db="+db+"&z="+z+"&x="+x+"&y="+((1 << z) - y - 1);

}

map = new OpenLayers.Map('map', {

projection: "EPSG:900913",

layers: [mapquestOSM, mbTiles, hostedTiles]

});

map.zoomToMaxExtent();

var switcherControl = new OpenLayers.Control.LayerSwitcher();

map.addControl(switcherControl);

switcherControl.maximizeControl();

</script>

</body>

</html>

Our mbtiles.php file to extract .mbtiles would be the same than in the previous case.