Getting started with CartoWeb

Creating and customizing a new project

Isabelle KIENER

Claude PHILIPONA

CartoWeb Introduction

- www.cartoweb.org :
  - CartoWeb is a ready-to-use Web-GIS
  - CartoWeb is a framework for building advanced and customized applications
  - It is based on Mapserver

- Easy to configure
  - .ini files
  - Smarty templates
  - Mapserver mapfiles
- Extensible
  - Adding new functionalities using plugins
  - Separating generic and specific development using projects

Summary

1. Getting started
2. Configuration files (.ini)
3. Templates and resources customization
4. Layers definition and hierarchy
5. Queries and hilight
6. Annotations

---

7. Print
8. Authentification and access control
9. Table rules
10. Debugging

Starting point

- Installation on Windows : see http://cartoweb.org/doc/cw3.3/xhtml/user.install.html#user.install.win32
- Downloads at http://cartoweb.org/downloads.html
- Steps
  - Install MS4W 2.2.4 or higher
  - Launch cartoweb-setup-3.4.0-RC1-win32.exe, with Gettext and demo data as options
  - Restart Windows
- Results
  - Folder **`C:\ms4w\apps\cartoweb3`**
  - http://localhost/cartoweb3/htdocs : web root of CartoWeb
  - http://localhost/cartoweb3/htdocs/client.php : raw development interface
  - http://localhost/cartoweb3/htdocs/demoCW3.php : working demo
- For the next step (lab project installation), the necessary files are in the folder "Step 1" of the archive located under C:\foss4g2007\lab-07.

Creating the project foss4g

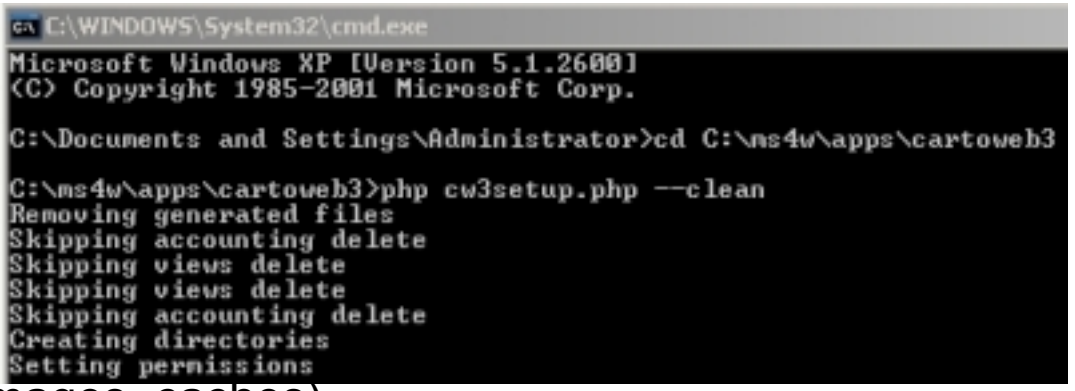
- Geodata installation
  - Unzip the archive data.zip into **`C:\FOSS4G07\Lab-07`**
- Project installation
  - Copy the folder foss4g into **`C:\ms4w\apps\cartoweb3\projects`**
  - Go to **`C:\ms4w\apps\cartoweb3\htdocs`**
  - Make a copy of demoCW3.php with name foss4g.php.
  - Edit it and change the project name.

```
<?php
$_ENV['CW3_PROJECT'] = 'foss4g';
require_once('client.php');
?>
```

- In a production environment, you'd have to configure your web server so that only the folder htdocs is externally visible.
- You still have to launch the setup script.

Setup script cw3setup.php

- See http://cartoweb.org/doc/cw3.3/xhtml/user.install.html#user.install.main.setup
- Open a command window.
- **`cd C:\ms4w\apps\cartoweb3`**
- **`php cw3setup.php`** + options
- Most current options
  - --help : name and use of all options
  - --clean : deletes all generated files (images, caches)
  - --install : installs CartoWeb
  - --base-url : in conjunction with --install; url giving access to the web root of CartoWeb
  - --project : in conjunction with --install; restricts the action to a project

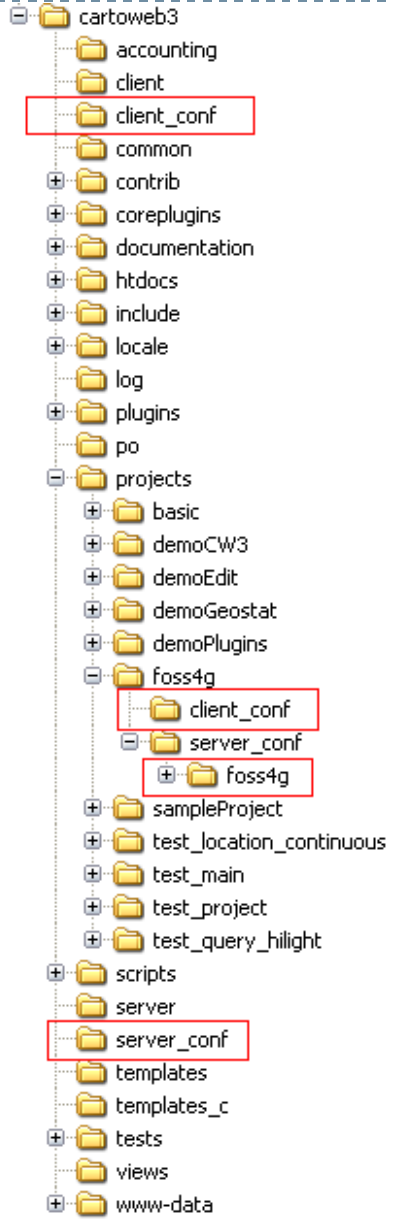


- In our case

```
php cw3setup.php --install --base-url
  http://localhost/cartoweb3/htdocs --project foss4g
```

- You can now access http://localhost/cartoweb3/htdocs/foss4g.php

.ini configuration files



- Locations
  - Upstream .ini files are in the folders client_conf and server_conf.
  - Project .ini files are in the folders foss4g/client_conf (client-side configuration) and foss4g/server_conf/foss4g (server-side configuration).
- How it works
  - If the value of a parameter is given in a project, this value overrides the default value given in the upstream CW configuration files.
  - Otherwise, the upstream value is used.
- Documentation
  - The files and the parameters within are documented in the user manual : http://cartoweb.org/doc/cw3.3/xhtml/cartoweb.user.html

Simple parametrization

- images.ini | client-side

  http://cartoweb.org/doc/cw3.3/xhtml/user.images.html

  - Modify allowed mapsizes, and default mapsize.

- location.ini | client-side

  http://cartoweb.org/doc/cw3.3/xhtml/user.location.html

  - Modify panRatio.
  - Hide "recentering on coordinates".

- location.ini | server-side

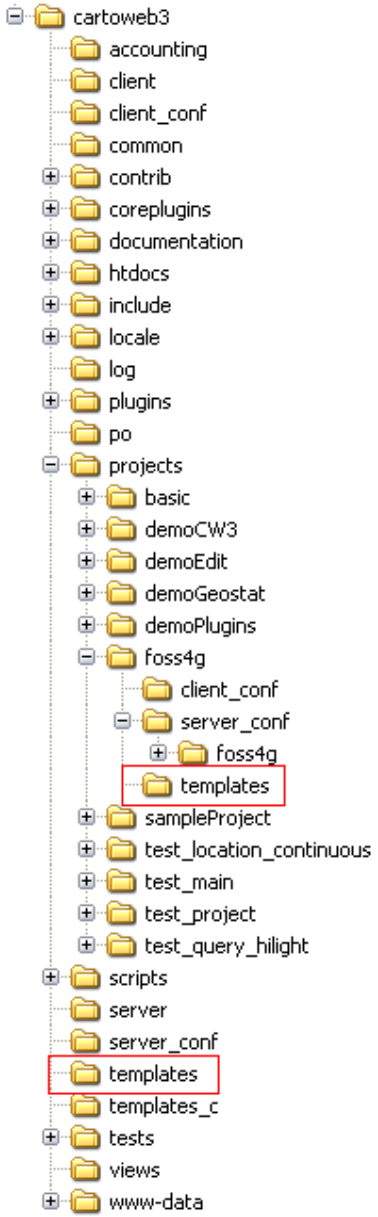
  - Modify allowed scales, and default scale.
  - Add a new shortcut for Austria.

- Don't forget

```
php cw3setup.php --clean
```

and the button reset_session or type ?reset_session at the end of the url so that your modifications are taken into account.

Templates customization



- Locations
  - Upstream templates are in the folder *templates*.
  - Project templates are in the folder *foss4g/templates*.
  - The main template is the file *cartoclient.tpl*.
  - Bits of templates may be handled by the relevant plugins; see e.g. *coreplugins/layers/templates*. More examples later.
- How it works
  - A project template replaces the corresponding upstream template.
- Documentation
  - The handling of the CW Smarty templates is documented in the user manual : http://cartoweb.org/doc/cw3.3/xhtml/user.template.html

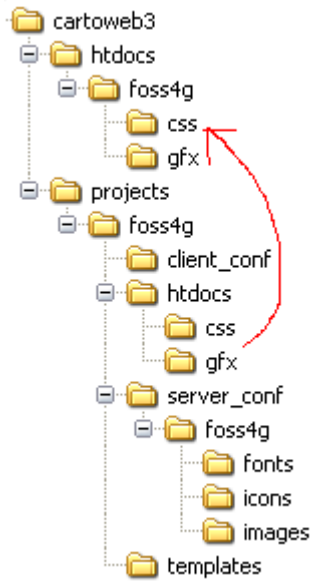
Customizing cartoclient.tpl

- In the project foss4g, create a folder *templates*.
- Copy the upstream main template (*templates/cartoclient.tpl*) into this new folder.
- Edit this file and make your modifications. For example, change the title and remove the debug messages (around line 150).
- You can edit a .tpl file like a simple html, considering the Smarty variables as constants.
- The handling of external resources (images, js, css) is described later.
- Empty the CW caches:

```
php cw3setup.php --clean
```

- If necessary, empty your browser's cache (usually with F5).

Adding resources



- Locations
  - Upstream resources are in the folders *htdocs/gfx* (for images), *htdocs/css* (style sheets) and *htdocs/js* (javascripts).
  - Project resources mirror the upstream hierarchy.
  - Some resources are directly available in the relevant plugins; for instance the icon of the zoom-in tool is to be found at *coreplugins/location/htdocs/gfx/zoomin.gif*.
- How it works
  - Project resources replace the corresponding upstream resources.
  - Resources have to be externally visible (through http), i.e. they must be under the upstream htdocs; the setup script (with the option --install) makes the necessary copies.

Adding resources to cartoclient.tpl

- In the project foss4g, create a folder *htdocs*.
- In this folder, create a folder *gfx* and a folder *css*
- Copy the files *logofoss4g.png* and *logofoss4g.css* into their respective folder
- Edit cartoclient.tpl
  - link the new css (in the head)

```
<link rel="stylesheet" type="text/css" href="{r type=css}foss4g.css{/r}" title="stylesheet" />
```

  - integrate the new image somewhere

```
<img src="{r type=gfx}logofoss4g.png{/r}" alt="foss4g" border="0"/>
```

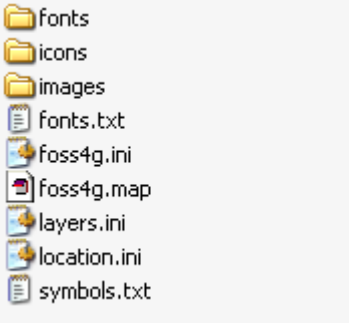
- These examples demonstrate the use of the resource tags {r}.
- Launch the install script and empty the CW caches

```
php cw3setup.php --install --base-url
  http://localhost/cartoweb3/htdocs --project foss4g
php cw3setup.php --clean
```

- If necessary, empty your browser's cache (usually with F5).

Layers configuration

- Location
  - The layer configuration files are in folder server_conf/foss4g.
  - These files are
    - the mapfile *foss4g.map* and its annexes (symbols, fonts...),
    - *layers.ini*, defining the hierarchy,
    - *foss4g.ini*, defining the initial state of the application.



- Documentation
  - Mapserver deserves a few workshops for its own sake. http://mapserver.gis.umn.edu/docs should be in your bookmarks' list.
  - For the CartoWeb part of the configuration, see http://cartoweb.org/doc/cw3.3/xhtml/user.layers.html.

Layers tree

- The layers hierarchy is defined in *layers.ini*.
- Two types of CW layers :
  - Layers : they correspond 1-to-1 to Mapserver layers, defined in the mapfile.
  - LayerGroups : they contain individual Layers or other LayerGroups.
- The notion of LayerGroup enable a hierarchy with infinite depth (only two levels with Mapserver).
- At the top, there is always a LayerGroup called root.
- Automatic generation of legends: autoClassLegend = true.

Parameters for a Layer

- If you don't need special parameters (label, icon or link) for your mapfile layer, then you can avoid defining it in the layers.ini
- For specifications, here are the mandatory parameters :

```
layers.LAYER_ID.className = Layer
layers.LAYER_ID.msLayer = mapserver_layer
```

- and additional parameters :

```
layers.LAYER_ID.label = label
layers.LAYER_ID.icon = image file
layers.LAYER_ID.link = url
```

[must be stored in folder icons]

Parameters for a LayerGroup

- Mandatory :

```
layers.LAYER_ID.className = LayerGroup
layers.LAYER_ID.children = layerId1, layerId2, layerId3
```

Rem: layerId1 can be a Layer specified in the mapfile only or in the layers.ini file, or another LayerGroup

- Optional :

```
layers.LAYER_ID.label = label
layers.LAYER_ID.icon = image file
layers.LAYER_ID.link = url
layers.LAYER_ID.aggregate = true|false
layers.LAYER_ID.rendering = tree|block|radio|dropdown
```

Example of layers.ini

```
layers.root.className = LayerGroup
layers.root.children =  background, contour, physical, human
layers.root.rendering = block

layers.background.className = LayerGroup
layers.background.children = raster, borders
layers.background.rendering = radio
layers.background.label = Background

layers.raster.className = Layer
layers.raster.label = Relief
layers.raster.msLayer = raster

layers.borders.className = Layer
layers.borders.label = Borders
layers.borders.msLayer = borders
.....
```

Initial map state

- Configuration of the initial state of the application (selected layers, location)
- Defined in *foss4g.ini*

  http://cartoweb.org/doc/cw3.3/xhtml/user.config.html#user.config.server.maps_config.initial
- Possible properties for Layers and LayerGroups
  - selected
  - hidden
  - frozen
- Only for LayerGroups
  - unfolded
- Initial location given by a bbox "xmin, ymin, xmax, ymax"

```
mapInfo.initialMapStates.default.location.bbox = "72705, 1620431, 1197822, 2677441"
mapInfo.initialMapStates.default.layers.raster.selected = true
```

Practical exercise

Using the ready-to-use Mapserver layers in the file *layers for mapfile.txt*, build the layers.ini file corresponding to the layers hierarchy described in *layers tree.pdf*.

Structure	Visible label
root	
background	
raster	*Relief*
borders	*Borders*
contour	*Contour lines*
physical	*Physical geography*
hydrography	*Hydrography*
sea	*Sea*
lakes	*Lakes*
rivers	*Rivers*
mountains	*Mountains*
summits	*Summits*
glaciers	*Glaciers*
human	*Human geography*
populated_places	*Populated places*
built_up	*Built-up areas*
towns	*Towns*
transport	*Transports*
railways	*Railways*
airports	*Airports*

Layer group
Layer

Enabling a plugin

- Core plugins are always enabled:
  - static tools, tables, images, layers, location, map query, ...
- Extension plugins must be explicitly enabled:
  - hilight, outline, mapOverlay, exportHtml, exportPdf, exportRtf, csv export, auth, ToolTips, layerReorder, views, edit, accounting, locate, Geostat, ...
- Client plugins are enabled in *client_conf/client.ini*.

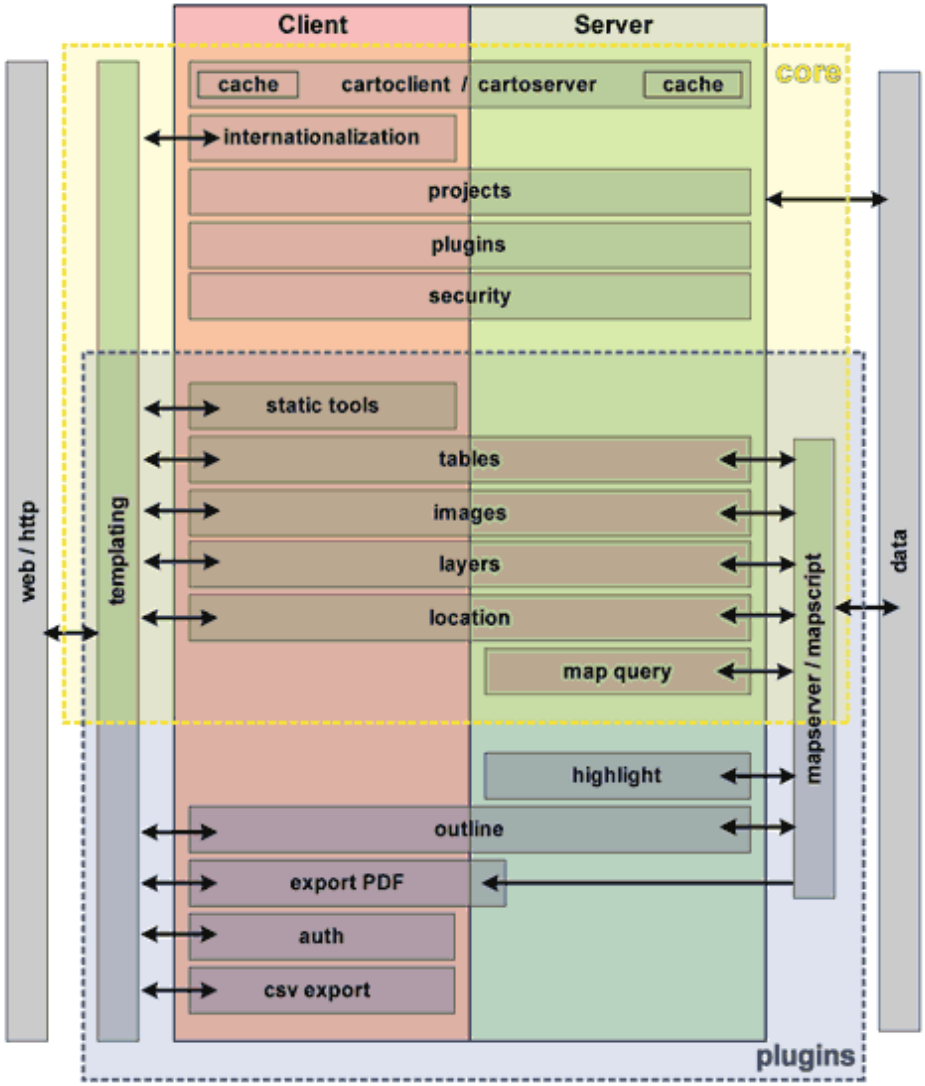
```
loadPlugins = auth, exportPdf
```

- Server plugins are enabled in *server_conf/foss4g/foss4g.ini*.

```
mapInfo.loadPlugins = hilight, exportPdf
```

- Some plugins are both client-side and server-side.

Plugins



Functional modules

Core vs optional plugins

Client and/or server plugins

camptocamp

Making a layer queryable

- In mapfile foss4g.map, insert

```
TEMPLATE "ttt"
```

, ttt being a dummy string

  into every queryable layer.

- This enables the standard Mapserver queries and hilight.
- To set which attributes are to be displayed, add a metadata

```
METADATA
  "query_returned_attributes" "spaces separated list"
END
```

- Make the layers included in the list *foss4g queries.pdf* queryable, and set the query_returned_attributes values.
- Documentation

  http://cartoweb.org/doc/cw3.3/xhtml/user.query.html

CartoWeb queries and hilight

- CartoWeb supports persistent queries as well as independent hilighting options for every layer.
- Enable the server plugin hilight.
- Add a *query.ini* file in the server-side configuration, and set

```
drawQueryUsingHilight = true
```

- In the mapfile *foss4g.map*, insert into every layer the metadata

```
"id_attribute_string" "OGC_FID"
```

- The data must contain a real ID attribute.
- For persistent queries, check additionally the client-side *query.ini*,

```
persistentQueries = true
```

- and a clear_query button is defined in cartoclient.tpl:

```
{if $query_result|default:''}
    {$query_result}
{/if}
```

CartoWeb queries and hilight

- You can now define a hilight layer for every queryable layer.
- It is a normal Mapserver layer; it must be named *abc_hilight*, where *abc* is the name of the non-hilighted layer.
- It is not included in the layers hierarchy (layers.ini).
- Depending on the hilight effect you want, it can be included before or after the initial layer.
- You can find ready-to-use hilight layers in the file *hilight layers.txt*.
- Add the new symbol to *symbols.txt*. It is used in the layer airports_hilight.
- Documentation

  http://cartoweb.org/doc/cw3.3/xhtml/user.query.html#user.query.mapfile.hilight

Enabling the outline plugin

- Enable the plugin outline in *client_conf/client.ini.*
- Enable the plugin outline in *server_conf/foss4g/foss4g.ini*.
- Enable the plugin mapOverlay in *server_conf/foss4g/foss4g.ini*.
- Insert the config file outline.ini into *server_conf/foss4g*.
- This file sets the Mapserver layers to be used by the plugin, for points, lines and polygons.
- The corresponding layers (ready-to-use in *outline layers.txt*) must exist in the mapfile.
- You can customize them.
- Try to add new symbols for point features.
- Documentation

  http://cartoweb.org/doc/cw3.3/xhtml/user.annotate.html

Customizing a plugin template

- As an example, we'll remove the hexadecimal color values in the outline tab.
- Copy the upstream outline template (*cartoweb3/plugins/outline/templates/outline.tpl*) in the project. The spelling and the path must be identical.
- Edit the template.
- Empty the caches.

Enabling the PDF export

- Enable the plugin exportPdf in *client_conf/client.ini*.
- Enable the plugin exportPdf in *server_conf/foss4g/foss4g.ini*.
- You need an *exportPdf.ini* (client-side).
- An example is available.
- Starting from this example, try playing around with the blocks, the formats...
- Be sure to test the mode pdfRotate.
- Documentation

  http://cartoweb.org/doc/cw3.3/xhtml/user.pdf.html

Enabling access control

- A security mechanism implementing the concepts of users, roles and permissions is available.
- Enable the plugin auth in *client_conf/client.ini*.
- You need a *auth.ini* file to define the users and their roles.
- An example is provided.
- Try adding new users and new roles.
- The special roles *anonymous*, *loggedIn*, and *all* are pre-built.
- To generate the md5sum of the passwords, this site may come in handy :

  http://pajhome.org.uk/crypt/md5/

- Documentation

  http://cartoweb.org/doc/cw3.3/xhtml/user.security.html

Global access control

- To restrict access to the application to certain users, you have to explicitly give the list of the allowed roles.
- In *client_conf/client.ini*, add a parameter

```
securityAllowedRoles = loggedIn
```

[default is all]

- With this setting, only authenticated users are allowed.

Access control to layers

- It is possible to make some layers available only to some roles.
- You need a *layers.ini* config file on the client-side, with the parameter

```
applySecurity = true
```

- Then go to the mapfile, and, for each protected layer, add the following metadata :

```
METADATA
  "exported_values" "security_view"
  "security_view" "roles list"
END
```

- For a LayerGroup, edit *layers.ini* (server-side), and add

```
layers.LAYER_ID.metadata.security_view = roles list
```

Access control to printing

- Printing may be completely restricted to some users.
- In *exportPdf.ini*, edit the parameter :

```
general.allowedRoles = roles list
```

- You can also restrict the use of some print formats to some users.
- In *exportPdf.ini*, edit the parameters :

```
formats.FORMAT_ID.allowedRoles = roles list
```

Modifying the query results table

- So-called tableRules plugins allow you to modify the content of the query result tables. For example, you can generate hyperlinks, include images, or even make a request to a distant database to display more info about the selected features.
- It's slightly more complex than configuring standard plugins, since you have to write some php code.
- Documentation
  http://cartoweb.org/doc/cw3.3/xhtml/dev.newplugin.html#dev.newplugin.special.tables
- We show here an example on the layer airports, by making an hyperlink with the content of the column NAM.
- Copy the folder *foss4gTableRules* in the *plugins* of the project.
- Enable the plugin foss4gTableRules in *client_conf/client.ini*.

Tools for debugging

- Development profile: no cache activated, jsTrace window, stack trace display through failure.tpl
  - client_conf/client.ini: profile = development
  - server_conf/foss4g.ini: profile = development
- Enable Firebug
- Logging framework: Log4php
  http://www.cartoweb.org/doc/cw3.3/xhtml/dev.debug.html

Contacts

Camptocamp SA
PSE A – Parc Scientifique EPFL
1015 Lausanne
Switzerland

Tel. +41 21 619 10 10

www.camptocamp.com / www.cartoweb.org

isabelle.kiener@camptocamp.com
claude.philipona@camptocamp.com