mappiamo Documentation

Release latest

Oct 05, 2017

English documentation

#mappiamo - EN

This is the #mappiamo Ensglish documentation.

CHAPTER 1

Introduction

#mappiamo is a CMS that allows you to create and leverage content through the use of OpenData, the geo-location and microformats. It can be used for processing the data produced by public administrations, collecting content (crowdsourcing), civic hacking and providing a basis for the portal of a smart city.

CHAPTER 2

Installation

Download #mappiamo package from GIT, and copy all files to your web host by FTP. Copy files to subdirectory if required, or to public_html root. Login to your control panel or phpMyadmin to create database user with password and add database to user. Give all access rights to your database user. When you copied all files to your host, access to the #mappiamo root by your browser by http. Setup process will be started. Fill all required fields. If the process done without error, you can access to the content manager on the URL: http://{{}your_host{}}/manager/

If something changed later (for example database password) edit settings.php from the root of installed #mappiamo.

1. Row 7: Fill or modify your sitename
2. Row 8: Fill or modify your domain

The database access storef from row 14 to 19:

1. Row 14: Database name
2. Row 15: Database type
3. Row 16: Database hostname
4. Row 17: Database prefix
5. Row 18: Database username
6. Row 19: Database password

If you need e-mail service, setup your SMTP provider:

1. Row 21: Your e-mail
2. Row 22: Username for SMTP service
3. Row 23: Password for SMTP service
4. Row 24: Hostname for SMTP service

Using content manager

CHAPTER 3

Create new content as admin or editor

You can create several type of content.

1. Post: this is a simple text based blog content (with marker on the map if required)
2. Place: content for the place selected on the map
3. Event: event is like the place, but you can define start and end dates on extra fields
4. Route: this content contains longer route on the map for long distance travels between cities

All content types can create marker on the map and can be used draw-on-the-map functions, excluding route. Route can contains only route plan.

CHAPTER 4

About semantic web

The Semantic Web is an extension of the Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats and exchange protocols on the Web, most fundamentally the Resource Description Framework (RDF).

According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries". The term was coined by Tim Berners-Lee for a web of data that can be processed by machines. While its critics have questioned its feasibility, proponents argue that applications in industry, biology and human sciences research have already proven the validity of the original concept. (wikipedia)

Note: Mappiamo automatically support semantic web for better result on google searches by content and meta data. The standard semantic data generated automatically by the content module.

CHAPTER 5

Create automatic meta data by form

Semantic data generated by meta data and content. The content editor or administrator can insert metadata when modify saved content step-by-step, but metadata generator form is available on the frame “Use meta data wizzard” on “Meta” tab. Editor can select form theme, and the selected form can be filled and savaed. These forms are folows the standard rules of semantic web data.

Note: Always read the rules before fill the metadata generator form. For example, if the content meta assigned to organization, the instrictions can be found here: https://schema.org/Organization. Currently we offer four pre-generated form for automatic metadata creation.

CHAPTER 6

Insert contents to category

Create category, and group several types of content to selected category. The selected category will group contents and markers on the map if required. The groped contents can be listed on one page, and the content markers will be displayed at one time on the map.

CHAPTER 7

Create pages

Create content type “pages” if you want to display the collection of content by menu. Pages can contains category (with any contents), one content, module generated service or page, and events with date filter on the top of page.

Note: If you choose “Add event” when adding page content, you will found a new dialog with several parameters, not only the content selection list. Here you can insert more than one events groupped by categories, and you can setup the sort order and filter functions for this page by input fields. Only this one type of page, the users can re-define event filter if enebled on the top.

CHAPTER 8

Create custom menus

Create menu with name on the content manager. When menu named and created, use it on “Page”. Select manager’s menu “Page”, click on previously created page-content (with document, category, modul or events) and insert selected page content to any menu. Select more than one times and add if required.

Note: Very important, that the created menu must be inserted to the template of content type by name or menu ID.

Widgets on your template

You can insert several widgets to your own #mappiamo template. You have to edit tamplete files only with your favorite IDE / text editor. Widgets are the part of ducument front-end with several functions. Some of them can be inserted to the content, some of then can be inserted to the sidebar on left or right.

Note: If the widger name contains word “Box”, the widget primary created for the sidebar, not the column of main content. but because the template can be modified with several tricks, these widget can be used under or within the main content text.

Note: New widgets required new CSS classes for correct display. Check the HTML source code to get widget’s class names.

CHAPTER 9

Address

- Usage code example:

```
<?php M_Template::widget('address'); ?>
```

This widget have no parameters, creating search box for map, the widget centering map for the search address. The search string must be real name (for example city name) to get real latitude and longitude.

Note: This widget is the part of Leaflet panel widget, this widger required to show address search function.

CHAPTER 10

Bottom menu

- Usage code example:

```
<?php M_Template::widget('bottommenu', array($ID)); ?>
```

Display bottom menu items. This widget have 1 parameter, the menu ID.

Note: Menu must be created by manager, you can insert any menu of them by ID.

CHAPTER 11

Allmeta box

- Usage code example:

```
<?php $this->widget('box_allmeta'); ?>
```

This widget have no parameters, creating list (table) of all meta data of content. This widget is ideal for right column, but van be used unser the main content. The disabled meta names is on the row 13 on the code.

CHAPTER 12

Box

- Usage code example:

```
<?php M_Template::widget('box', array($image, $title, $desc, $link)); ?>
```

This widget display image box, using four parameters.

1. $image -> image path
2. $title -> title text on image (positioned by customizable CSS!)
3. $desc -> description of image (positioned by customizable CSS!)
4. $link -> link if user click on the image

CHAPTER 13

Collabrators box

- Usage code example:

```
<?php $this->widget('box_collabrators' array(n)); ?>
```

This widget have one parameters “n”, what is the maximum number of collabotators article based on the selected content. The collaborator’s e-mail must be saved to the meta value with name “collaborator”.

Note: This widget have no parameter about collaborators name ir e-mail, because these names depending on the document. This is the reason why the collaborator’s identifier definied by meta data of selected document not by the template.

CHAPTER 14

Cookie box

- Usage code example:

```
<?PHP $this->widget('box_cookie'); ?>
```

This widget have no parameters, creating alert box for cookie usage.

CHAPTER 15

Distance box

- Usage code example:

```
<?PHP $this->widget('box_distance'); ?>
```

This widget have no parameters, creating list (table) of related articles not far from the current content.

Note: The distance is fixed on code, the radius is 1 km.

CHAPTER 16

Events box

- Usage code example:

```
<?PHP $this->widget('box_events'); ?>
```

This widget have no parameters, creating list (table) of events not far from the current content.

Note: The distance is fixed on code, the radius is 1 km.

CHAPTER 17

Instagram box

- Usage code example:

```
<?PHP $this->widget('box_instagram', NULL); ?>
```

This widget have one parameter what is the hashtag for images. If this parameter missing or NULL, the default hashtag is ‘tourism’. With meta name ‘hashtag-instagram’ can be overwite the deafult hashtag to anything else.

Note: If you use meta to define instagram hashtag instead of template, you can get images several hashtags on all documents where ‘hashtag-instagram’ have value.

CHAPTER 18

Onemeta box

- Usage code example:

```
<?PHP $this->widget('box_onemeta', '[meta_name]'); ?>
```

This widget have one parameter what is the meta name to get the value of only oane meta data.

Note: This widget can be used on the column of main content.

CHAPTER 19

Youtube box

- Usage code example:

```
<?php $this->widget('box_youtube', array('[developer key]', '[channel id]',␣
↪[maximum content])); ?>
```

This widget have 3 parameters. Developer key, youtube channel id, and the maximum number of youtube content.

Note: This widget can be inserted to the left or right sidebar column, and creating scrollable carousel of selected channel content.

CHAPTER 20

Allmeta

- Usage code example:

```
<?PHP $this->widget('content_allmeta'); ?>
```

This widget have no parameters, creating list (table) of meta data from the current content.

Note: This widget created for list or table of standard schemantic data if available.

CHAPTER 21

Slideshow

- Usage code example:

```
<?PHP $this->widget('content_slideshow'); ?>
```

This widget have no parameters, creating slideshow on the content column from all images included to the current content.

Note: If more than one images inserted to the content, the widget will be show the gallery where you insert. The best place is under the content column.

CHAPTER 22

Divided menu

- Usage code example:

```
<?php M_Template::widget('dividedmenu', array($ID)); ?>
```

Display divided menu. This widget have 1 parameter, the menu ID.

CHAPTER 23

Dropdown menu

- Usage code example:

```
<?php M_Template::widget('dropdownmenu', array($ID)); ?>
```

Display dropdown menu. This widget have 1 parameter, the menu ID.

CHAPTER 24

Intro

- Usage code example:

```
<?PHP $this->widget('intro'); ?>
```

This widget have no parameters, display intro image.

CHAPTER 25

Headline

- Usage code example:

```
<?PHP $this->widget('content_headline'); ?>
```

This widget have no parameters, creating group of some data and metadata which are rewired on content column between title and content text.

CHAPTER 26

Flickr

- Usage code example:

```
<?PHP $this->widget('flickr'); ?>
```

This widget have no parameters, creating flickr image groups on the map by visible box of map.

CHAPTER 27

Form contact

- Usage code example:

```
<?PHP $this->widget('form_contact', array('[registered username]')); ?>
```

This widget have one parameter, the parameter must be the username of registered Mappiamo user. This widget creating form with input fields for sending simple message with ajax validation.

CHAPTER 28

Gravatar

This widget included to the content module, cannot use on the template or MVC view. The widget fetching gravatar icon by the content creator's e-mail address, if the editor registered on this service.

CHAPTER 29

Jplayer

- Usage code example:

```
<?PHP $this->widget('jplayer'); ?>
```

This widget have no parameters, creating javascript player for audio (or video) content. The required meta name is 'audio' and the meta value must be the full url of audio or video file.

Note: The meta data value is the full URL of audio file, but the correct encoding is very important. Plase refer to the officiel JPlayer page to inform about usable audio formats.

CHAPTER 30

Leaflet panel widget

- Usage code example:

```
$Panel_names = array([panel_name_1], [panel_name_2], ....., [[panel_name_n]]);
    $Panel_icons = array([icon_name_1], [icon_name_1], ....., [icon_name_n]);
    $this->widget('leaflet_panel', array($Panel_names, $Panel_icons));
```

This widget have two required parameteres, booth have to be arrays. The array of Panel_names listed the names of buttons on will be isplayed on the map. On the template directory must be created .php files with same name. For example, if the panel_name_1 is “SearchBox”, SearchBox.php must be created to the template directory. This file can contains any required code, for example widgets.

- Usage code example of SearchBox.php:

```
<div id="SearchBox" class="PanelOnTheMAP">
        <?php M_Template::widget('address'); ?>
</div>
```

- **Rules:**
  - The panel code must be included between .
  - The div ID must be same as the panel name.
  - The class “PanelOnTheMAP” required.
  - Between can be inserted any widget or code.
- The panel icon array contains the name of bootstrap icon. For exammple if the bootstrap icon name is glyphicon-search, the panel icon name is only “search”.

CHAPTER 31

Map

- Usage code example:

```
<?PHP $this->widget('map' array($zoom)); ?>
```

This widget have 1 parameter, the default zoom. This widget display map anywhere on the content page. This widget display map (with markers, draw or route) on the visitor's interface.

CHAPTER 32

Menu

- Usage code example:

```
<?PHP $this->widget('menu' array($ID)); ?>
```

This widget have 1 parameter, the menu id. This widget display menu item.

CHAPTER 33

Video box

- Usage code example:

```
<?PHP $this->widget('videobox'); ?>
```

This widget have no parameters, creating embedd iframe player for youtube content by full url. The required meta name is ‘videobox’ and the meta value must be the full url of youtube video.

CHAPTER 34

Lastcontent

- Usage code examples:

```
<?php $this->widget('lastcontent', array(5)); ?>
<?php $this->widget('lastcontent', array(5, 'event', 'start', 'from_now')); ?>
<?php $this->widget('lastcontent', array(5, 'post', 'created')); ?>
```

This widget have parameters. The first is the maximum number of content, this is required.

All other paramteres are optional (not required because default values are available): [content type], [ordering column name], and if the content type is ‘event’, the last parameter ‘from_now’ shows only current and future events. If the last parameter is “now” for event type, the list will be displayed event only if the event currently running.

CHAPTER 35

Full featured menu

- Usage code example:

```
<?php M_Template::widget('menu_full', array('[category name]', '[treemenu|popmenu]
↪', '09', 'check')); ?>
```

This widget have parameters. Creating custom menu system by Mappiamo “pages” and “menus”, and display selected categories on the map.

- Parameters:

1. The category name
2. Menu type: ‘treemenu’ or ‘popmenu’
3. Template number of menu only. Menus have 15 templates, the menu template number can be 1 to 15.
4. How menu display the selected catorgory contents:

- ‘link’ - the category opens new page with content list
- ‘check’ - the category displays as marker on the map

CHAPTER 36

Owl image

- Usage code example:

```
<?PHP $this->widget('owl_image', array('category', 4, 60)); ?>
<?PHP $this->widget('owl_image', array('path', 6, 'templates/soccorso/images/
↪partners', 'index.php?module=category&object=59')); ?>
```

This widget have parameters, creating image carousel to the main content column. The source images can get from two different source: ‘category’ or ‘path’. This is the first parameter.

If the image source is ‘path’, the 3rd parameter must be the relative path to the directory contains images.

If the image source is ‘category’, the 3rd parameter must be the id number of category where the widget reads all images from content. This category must be created and filled with grouop of contents.

The 2nd parameter is the maximum number of items to show.

The 4th parameter is the link to open when user click on image. This is optional. If the source is ‘category’, the link will open the document contains clicked image.

CHAPTER 37

Owl video

- Usage code example:

```
$TubeID = array('jkovdYV0qm0', 'dw6wZQkfsn0', 'CqdSzVXkhmY', 'km3JiaPqWMI',
↪'NyCwOdyhZco', 'YJTxnhjpF3U', 'HOVYTZkvjH8', '2Tlou1Vdg6Y', '0_rtwI_nUlI',
↪'LCtp7D0uCjA');
$this->widget('owl_video', array($TubeID, 3));
```

This widget have parameters, creating video carousel to the main content column. The first parameter must be an array on the separated variable, contains all youtube video id required for the carousel.

The second parameter is how many videos display at once by the scrollable carousel.

CHAPTER 38

Share

- Usage code example:

```
<?PHP $this->widget('share', array($site_id)); ?>
```

Share content on social networks.

CHAPTER 39

Slider

- Usage code example:

```
<?PHP $this->widget('slider', array($content_id)); ?>
```

This widget creating image slider from the content by content ID.

CHAPTER 40

Weather

- Usage code example:

```
<?PHP $this->widget('weather'); ?>
```

This widget have no parameters, creating weather report on the map.

CHAPTER 41

Disqus

- Usage code example:

```
$Types = array('post', 'event');
<?php M_Template::widget('disqus', array($Types)); ?>
```

This widget have parameter as array named $Types. Creating comment section on content page. Disqus account and disqus site name required. On the parameter '$Types' must be listed all content types (post, event, place, route) where the disqus comment service will be available. Insert this widget under the main content page.

Note: You must register your installed #mappiamo on the Disqus service page as site administrator to get your unique Disqus site name. If you have this name, you must define it on the manager -> preferences.

The API module

By API module, you can get data for external application for example mobile app or another #mappiamo

Note: If 'auth' parameter required to get data by API module, the auth key must be storen in the preferences table with name 'api_auth'.

Note: The 'lang' parameter is always optional. Without this parameter API uses internal auto language.

CHAPTER 42

Get all places

http://{[}site_uri{]}/index.php?module=api&task=allpois&auth={[}auth_key{]}&lang={[}language]

CHAPTER 43

Get all routes

http://{[}site_uri{]}/index.php?module=api&task=allroutes&auth={[}auth_key{]}&type=route&lang={[}language]

CHAPTER 44

Get all polygons

http://{[}site_uri{]}/index.php?module=api&task=allroutes&auth={[}auth_key{]}&type=polygon&lang={[}language]

CHAPTER 45

Get all markers by 1 km distance from route

http://{[}site_uri{]}/index.php?module=api&task=poisonroute&route={[}route_name{]}&auth={[}auth_key{]}&lang={[}language**]**

Note: The value of 'route' parameter must be same as the value of 'name' column on content table.

CHAPTER 46

Get all markers within polygon

http://{[}site_uri{]}/index.php?module=api&task=poisonroute&route={[}polygon_name{]}&auth={[}auth_key{]}&lang={[}language]

Note: The value of route parameter must be same as the value of column name on content table.

CHAPTER 47

Search by field content

http://{[}site_URI{]}/index.php?module=api&task=search&auth={[}auth_key{]}&field={[}col_name{]}&data={[}col_value]

CHAPTER 48

Get category contents by category ID

http://{[}site_URI{]}/index.php?module=api&task=category&object={[}category_id]

CHAPTER 49

Get one content by content ID

http://{[}site_URI{]}/index.php?module=api&task=content&object={[}content_id]

CHAPTER 50

Get marker data by distance from coordinates

http://{[}site_URI{]}/index.php?module=api&task=search&lat={[}latitude{]}&lng={[}longitude{]}&radius={[}distance_by_km]

CHAPTER 51

Get marker data by distance from coordinates filter by category ID

http://{[}site_URI{]}/index.php?module=api&task=search&lat={[}latitude{]}&lng={[}longitude{]}&radius={[}distance_by_km]

Importers

CHAPTER 52

SHP2GeoJson Importer

The data importers created to save exteral data set to mappiamo at one step. The GeoJson importer can be used for import data created from .SHP source files by QGIS desktop software.

For the import process you need .geojson file exported from QGIS, and you have to create .INI file.

.ini file must be contains rules how to save .geojson data to mappiamo. The importer can use two labels: [{database_table_name}] and [fixed_data].

Under the optional label [fixed_data] must be listed the database table, clumn, and the value. For example, if you need to insert value "place" to all imported rows on table "contents" and column "type", the corrent row under this label: contents[type]="place".

The [{database_table_name}] is required label. For example, if to the table "contents" column "address" have to be inserted something, you have to enter these rows to .ini:

```
[contents]
address[]="Residenza"
```

The rules of this .ini label:

```
[{database_table_name}]
{table_column}[]="{geojson_prperty_name}"
```

You can use more than one labels for table name, and if you want to store more than one geojson property to the column, you can duplicate the row with several values of geojson prperty names.

Example of tested .ini file:

```
[contents]
address[]="Residenza"
title[]="Tipologia"
title[]="Residenza"

[fixed_data]
contents[type]="place"
```

New Updates

CHAPTER 53

M_Module better templates generation

The class M_Module generate the html page and implements the pattern MVC. Now you can use the same view file all time you want. You can use view files as section of a more complex template.

CHAPTER 54

Admin Panel Widget_List error

If Admin Panel “Widget_list” crash the problem is in the naming of the widget or in the naming of the main widget functions. The main function name should be like: mwidget_<widgetName>(){}.

Admin Panel could crash also if the name of other functions of your widget are a duplicate of other widgets function name. So keep attention to the naming of your functions.

CHAPTER 55

Mappiamo custom content type managment

In Mappiamo you can have only 4 content type by default: post, place, route, event. If you want insert custom type you have to follow these steps: 1. Create the display function for new type in modules/content/models/content.php 2. Add the new type Manager in modules/content/view/default.php 3. Insert new type in Types array in bin/mbin.object.php 4. Create a new class with this name: class.<newtype>.php in bin/classes (name are case sensitivie) 5. Register the new class in binaries.php

example of class.<newtype>.php:

```
<?php
class M_Newtype extends M_Post {
protected $type = 'newtype';
protected $kind;
protected $start = NULL;
protected $end = NULL;
public function __construct($id = NULL) {if ($id) {$this->read($id);}       }
public function get_start() {return $this->start;}
public function get_end() {return $this->end;       }
public function set_start($value) {$this->start = strval($value);}
public function set_end($value) {$this->end = strval($value);}
}
?>
```

CHAPTER 56

Call a model from controller

use this function inside a controller:

```
$this-> model("name_model", $parameters)
```

$parameters should be an array. When you pass the array $parameters, Mappiamo split it in a list of parameters for the “name_model” function. For example if i have $parameters[a,b,c] when i pass throught $this-> model(“name_model”, $parameters) the function “name_model” will be like this:

```
function name_model (a, b, c){
//some stuff
}
```

The order of data in $parameters array corresponds to the order of function parameters.

CHAPTER 57

Call a view from controller

use this function inside a controller:

```
$this-> view("name_view", $data)
```

$data should be an array otherwise data are not passed. It's important that you use the variable name as "$data" otherwise it doesn't work.

CHAPTER 58

Mappiamo

This is the Mappiamo documentation.

CHAPTER 59

Introduction

Italian translation required .....

The first italian subtitle

Italian translation required .....

The second italian subtitle