Table of Contents

p.mapper User Manual

Installation

see Quick install instructions for the main set up instructions.

some info about MapServer installation

Additional PHP settings

PHP.INI

- **Optional**: If you want to make use of the functionality to join database tables to layers (for details see below) you have to install the PEAR DB or the MDB2 library (you need to define the library in the INI file). Typically you will define the path to the installation directory of PEAR in the key *include_path* (under [PHP] - Paths and Directories)

Directory and File Structure

config

The directory for the configuration files. Contains config.ini and subdirectory *default*, which is the default location for custom.js, js_config.php, php_config.php, search.xml, and the mapfile. In the future XML files for detailed configuration could be added.

doc

Contains this documentation.

images

Contains all images used in the user interface. Subdirectory legend contains the legend icons. These are automatically created and updated.

incphp

This directory contains the PHP files with the overall functions.

- common.php: Common functions and variables used in several php files
- custom.php: Customizing settings, at the moment mainly the definition of tools in the toolbar
- js_init.php: Initialize various JS variables from PHP
- /init/init.php & initmap.php: Initialization of the application

- initgroups.php: Initialization of GROUP and GLAYER objects
- /locale/language_*.php: localization files for multilingual GUI
- map.php: Functions for rendering map
- /query/...: Functions for queries (identify, select, search)
- legend.php: Functions for creating legend
- group.php: Class functions for GROUP and GLAYER objects
- util.php: Auxiliary functions
- /print/print.php: General print functions
- /print/pdfprint.php: Special print functions for PDF creation
- /print/tcpdf.php: PDF creation functions (Author: Nicola Asuni)
- /xajax/...: PHP parts for XMLHttp/AJAX calls

javascript

The releases contain all JavaScript files compressed to a single file: 1 for standard files `pm_cjs.js` and 1 for all jQuery files `jquery/jquery_merged.js`. All uncompressed source files are under the subdirectory `/src/`. If you want to make modifications to an existing JS file it is recommended to do this in a separate JS file (like `custom.js`) under the `/config/.../` directory adding the modified function to this file.

All `JavaScript` files:

- common.js: commonly used functions
- custom.js: customized functions, like hyperlink calls, to be adapted to user needs
- forms.js: Functions for forms and scale selection list
- geometry.js: library for measurements/digitizing (Author: Federico Nieri)
- layout.js: set layout parameters (width/height, left/top) of the GUI
- mapserver.js: Main functions used for map creation, query, etc.
- pmapper.js: main p.mapper related functions, resize, GUI, etc.
- pmdraw.js: Map measuring/digitizing functions
- pminit.js: init functions
- pmjson.js: Functions for rendering JSON output of queries
- pmquery.js: generic functions for queries
- search.js: functions for attribute search
- sorttable.js: Sort result table functions (Author: Stuart Langridge)
- toc.js: TOC/legend definition
- wz_jsgraphics.js: Draw measure lines (Author: Walter Zorn)
- xmlhttp.js: XMLHttp/AJAX functions
- zslider.js: Create zoom slider (Author: Mark Wilton-Jones)
- zoombox.js: DHTML zoom/pan interface; based on DOM
- jquery/*.js: jQuery libraries

templates

Cascading Style sheets. Most formatting options in p.mapper are done via CSS.

Basic Design

Application Structure

The application is set up as a single PHP page (by default named 'map.phtml') with elements set in DIV's.

PHP and Javascript

The application uses a DHTML zoom/pan interface for its navigation. The functions of the zoombox.js file call as central counterpart the zoombox_apply() function in mapserver.js. This function spreads the calls to different `JavaScript` functions that then pass the necessary variables to the PHP files. In p.mapper 2 most calls to PHP files from JavaScript are performed via XMLHttp (AJAX) requests and call PHP scripts named x_abcdef.php.

Start

The application should be started via a JavaScript 'window.open' command. The file index.html contains sample links how to start the application from within a standard HTML page. Additional parameters are the size of the application window like

```
http://server/map.phtml?winwidth=700&winheight=500
```

and the language (see section below).

Multilingual User Interface

The application is designed with a multilingual approach. Nearly all text annotation in the user interface is written dynamically with PHP. The definition of the annotation strings is made in the files below /incphp/locale/language_xx.php where xx is the country id of the language. It uses a function emulating the GNU Gettext and could be substituted with the latter. The currently predefined languages are

- English
- German
- Italian (with corrections by Luca Marletta)
- French (provided by Paul Hasenohr)
- Dutch (provided by Wim Devos)
- Czech (provided by Jachym Cepicky)
- Brazilian Portuguese (provided by Rodrigo Gaete)
- Slovak (provided by Ivan Mincik)
- Spanish (provided by Nuria González)
- Russian (provided by Anna Kostikova)
- Hungarian (provided by Zoltan Siki)
- Chinese (simplified) (provided by Xiaobao Zang)
- Japanese (provided by Takashi Ota)

Adapt the locale files or add new ones for additional languages.

p.mapper also contains an alternative function "`__p()`" in common.php that uses a SQLite 3 database ('localedb.db') with UTF-8 encoding. The database can be edited with a CLI or more comfortable with a SQLite GUI. Substitute the file-based function "_p()" with this one if you want to use a DB driven solution. You can also maintain all language settings in the SQLite file 'localedb.db' and write out the 'language_xx.php' files with the PHP script 'extract_locales.php'.

NEW since version 1.9.4: the default character encoding is set to UTF-8. Use a text editor that supports UTF-8 encoding if editing the locales files (eg. Scite). If your map file includes non-ASCII characters (eg for layer DESCRIPTION or CLASS names), set the map2unicode entry in the config.ini to 1 (if the map file is in an encoding other than ISO-8859-1 add a METADATA entry like "MAPFILE_ENCODING" "ISO-8859-2" to the MAP tag).

To call the application in a certain language start the map.phtml file with the URL containing language=xx in the search portion of the URL, e.g.

```
http://server/map.phtml?language=en
```

where predefined settings are: en English, de German, it Italian, fr French, se Swedish, nl Dutch, cz Czech, br Brazilian, sk Slovak.

The default character encoding of the pages is UTF-8. In order to ensure correct display of query results from shapefiles the result strings are converted to UTF-8. If the layers are in a different encoding than ISO-8859-1 or WIN1252 (CP1252) - eg. DOS (CP850), UTF-8 encoding of a PostGIS layer, or other ISO encodings - you should specify this as a layer METADATA tag "LAYER_ENCODING" like

```
METADATA
  ...
  "LAYER_ENCODING" "UTF-8"
END
```

A correct functioning of this conversion requires the ICONV library available in PHP (usually included in standard builds).

Miscellaneous

Legend The legend icons are created every time when there had been changes to the map file. The init.php file checks the modification date of the map file and compares it with a log file out of the last legend icon creation. If the map file is newer, then the icons will be created newly. On the one hand this keeps legend up-to-date, on the other hand it avoids new creation of the icons and transmission to the user every time he requests the legend.

Printing For the print function you can choose between HTML and PDF output. The PDF output uses the free library FPDF by Olivier PLATHEY. You can increase the map image quality for the PDF print by increasing the setting pdfres in the config.ini file. Note that higher resolution increases file size of retrieved PDF files and map labels can become difficult to read.

Configuration

» GUI layout configuration

Configuration Options of the INI File

INI File /config/config_xyz.ini settings:

For p.mapper version up to 3.2: see the description in the ini file itself.

For version 3.3: » see INI File Settings with descriptions

Selection of the INI file at start up

It is possible to select the INI file to use for the configuration via the start up URL parameter &config=.... This setting requires to have the config file named like config_name-of-your-configuration.ini. E.g. &config=test will use the file config_test.ini in the /config/ directory. This allows flexible presentations based on the same common application code. Because config files can use different map files, one can also choose a different content via the config parameter. See the file start.html for examples how to start an application with different configurations.

Configuration options of the php_config.php file

- *Definition of categories for Legend/TOC:* Define arrays of groups/layers for each category displayed in Legend/TOC
- *Tools:* Arrays of the tool buttons (including tool tips and javascript functions), and spacer/separators.
- *List for menu:* Print, Download, and Help tools.
- *Toolbar theme:* Define one of the themes here.
- *Title and heading:* Heading may include a graphics file.
- *Tool mouseovers:* Whether tool images will swap between *_on, *_off, *_over.
- *PDF Print Settings:* Including title, font, header height/color, and logo.

Additional JavaScript related configuration options: `js_config.php`

Some settings for various JavaScript related options can to be set in the file

```
/config/default/js_config.php
```

For most settings: if not set, reasonable default values are taken.

The **layout** of the application is set in this file. See more details here

Also all **context-menu** settings for the TOC are set here

```
  var pmContextMenuList = [
       ...
  ]
```

See the short explanations there. . One of the more important settings in this file

Configuration Options of the MAP File

General Settings

RESOLUTION

In order to allow correct PDF printing and to have correct scales for printing set RESOLUTION tag below MAP to 96. This is the default screen resolution Windows calculates with. With regard to a vast majority of users using Windows (and in its default configuration) this should be an appropriate setting.

Symbols

p.mapper uses a symbol called 'circle' for highlighting selected features. This symbol definition has to be present either in the map file or the symbol file referenced in the map file. The default definition of this symbol is

```
SYMBOL
  NAME 'circle'
  TYPE ELLIPSE
  POINTS 1 1 END
  FILLED TRUE
END
```

Groups and Layers

p.mapper supports grouping of layers just like the cgi-interface. To achieve this, a secondary group and layer definition is done via special group/layer objects that include also the support for multi-lingual applications. It is recommended to avoid spaces in group or layer names.

In the ini file you can set the default order how layers/groups are displayed in the table of contents (allGroups) and which layers are set visible at start (defGroups). If a layer belong to a group specify the **group name**, otherwise use the **layer name**.

p.mapper supports the definition of **thematic categories**, thus allowing to group layers in the table of contents together (see description of INI parameter *useCategories*). If categories are used, all layers/groups have to be referenced in the *$categories* array in */config/php_config.php*. Categories are named in the appropriate language file in the incphp/locale folder.

Layer TOLERANCE settings for Queries

The query function of p.mapper uses the TOLERANCE settings in the map file. This means you should define tolerances in the map file for each layer you want to be queryable. Recommended are TOLERANCEUNITS pixels and settings for TOLERANCE between 2 and 5 for vector layers, and 0 for raster layers.

Layer/Group Descriptions and Fields for Queries

The configuration in the map file for description (i.e. the name displayed in the TOC or legend) of the layers/groups is done via the METADATA tag DESCRIPTION for each layer. If more than one layer belongs to a common group the name of the first layer is taken for the group name.

The fields used for displaying query results are defined in the METADATA tag RESULT_FIELDS. For shapefiles these fields have to be defined completely in upper case. For PostGIS layers the field names will typically have to be defined in lower case. The field names displayed in the result table are similarly defined as RESULT_HEADERS

```
"RESULT_FIELDS" "NAME_IT,TYPE_EN"
"RESULT_HEADERS" "Name,Street Type"
```

If no definitions are made for these METATAGS then the default values will be taken from the layer sources.

Notabene: All layers that shall be queryable need a **TEMPLATE** tag set for either the layer or the classes. Otherwise the query returns no result for this layer. This is a well known MapServer issue. So just define something like

```
LAYER
  ...
  TEMPLATE "void"
  ...
END
```

Joining Database tables to layers

This functionality makes use of the PEAR database abstraction layer (module DB). It uses PEAR syntax for the connection string to the database. For installation and detailed information, especially the connection string syntax, see the PEAR documentation, especially the section DB. Joins for a layer are

defined in the map file METADATA tag for the layer as RESULT_JOIN

Examples:

"RESULT_JOIN" followed by the database string:

PostgreSQL:

```
<----- PEAR Connection String ------->||<---- DB Parameters (join field, result fields ---->||<Join
"pgsql://postgres:passwd@host/Database||DB_table@DB_join_field@0@field1,field2,field3,field4||SHAPE_
```

ODBC (e.g. MS Access):

```
"odbc://'':''/host@odbc_connection_dsn||DB_table@DB_join_field@0@field1,field2,field3||SHAPE_join_fi
```

Because of limitations of the PEAR DB module a join of dBase tables is not working. On MS Windows systems you can use an ODBC connection instead, pointing to an MS Access file that links the dBase file (this solutions is also typically faster). On Unix systems you can only use the built-in functions of Mapserver for DBF joins or use the OGR Virtual Layer definition. A join works also for RASTER layers, but not for PostGIS layers. PostGIS layers can have a join defined directly in the map file DATA tag (see PostGIS manual).

Hyperlinks

In order to make the results of queries more flexible the application offers the possibility of hyperlinks for selected fields. This works both for layer fields and fields out of a joined DB table. The hyperlinks are defined in the METADATA tag for every layer like the following example:

```
"RESULT_HYPERLINK" "FIELD1||Link on Detail, FIELD2"
```

In between the quotes separate the hyperlink fields with commas. If you don't want to have the field value being displayed as the hyperlink text, you can add an ALIAS name after the field name separated with a double pipe sign like set for FIELD1 in the example above.

The hyperlink definition has its corresponding JavaScript function openHyperlink() (in the custom.js file) that is added to every link as the action. The function receives layer, field name, and field value. For every layer and field you can define how to process this information, e.g. include them in other JavaScript functions.

XY Layers

One can use point layers based on DBMS tables that have x/y fields for the coordinates. The data can be in any DB supported by PEAR. The definition in the map file has to be as "XYLAYER_PROPERTIES" in the METADATA tag, like the following example:

```
|--- IN METADATA --" |--- PEAR DB definition string ----------||- DB table -@- Filter -||coordinate
"XYLAYER_PROPERTIES" "sqlite://'':''/home/data/sqlite/gisdb.db||cities_cntrs@inh>100000||x,y"
```

Currently the data need to be in the same projection system as the map.

Miscellaneous Functions

Definition of Attribute Queries (Search Function)

p.mapper offers a simplified definition of attribute queries (search function). Since version 3.1 the attribute queries are defined in */config/default/search.xml*. See **here** for more details.

The query output is no longer HTML but a JSON string (see www.json.org for details). The output is then formatted to HTML via Javascript functions parsing the JSON objects. One could also use PHP instead. This makes it much easier to customize the display of the query results in different layouts. See the file */incphp/query/json_query_doc.txt* for a description of the JSON structure.

Zoom to pre-defined extent

You can start the application already zoomed to a specified extent. The default extent can be defined either directly with x/y min/max values or read from a feature of a map layer. The default extent is defined by an additional key/value pair for the starting URL. See the file 'index.html' for an example.

Manually defined extent:

Comma-separated list of coordinates: xmin, ymin, xmax, ymax; like

```
&me=3678982,2134585,4721175,3092410
```

Extent read from layer feature:

```
&zoomLayer=countries&zoomQuery=NAME@1@Italy@0@1
```

zoomQuery parameters (separated by @):

```
NAME: name of the field to search (upper case for shapefiles)
1: field type, 0 = numerical, 1 = alphanumerical
Italy: search string (case sensitive)
0: highlight feature, 0 = no, 1 = yes
1: set maximum map extent to feature extent, 0 = no, 1 = yes
```

Note: the slash character "/", for the URI needs to be properly encoded using `%2F`, otherwise you will get a wrong interpretation of the URI and subsequent errors.

Defining the active layers:

Additionally, it is possible to specify the layers/groups that shall be activated (visible) on startup via the *&dg=* URL key, with layers comma-separated

```
&dg=countries,cities
```

Utilities

If you are using ESRI ArcView 3 you can use the extension AV.p.mapper. This extension helps you to create a map file out of a given ArcView project. It also adds the necessary information, e.g. the fields to be used in query functions.

Credits

I would like to thank all who provided contributions to the p.mapper development:

- Federico Nieri and Alessandro Radaelli from Comune di Prato:
    - ♦ Contributions in quite a few files (Javascript and PHP) in order to make p.mapper 2 compliant with W3C XHTML standards.
    - ♦ Re-structured and enhanced measure functions, including geometry.js library.
- Luca Marletta from beOpen.it: For regularly testing of development versions and reporting bugs.

All projects and products that p.mapper is based on or uses code snippets from or that gave ideas for the development are listed below. See here for detailed information about their licenses.

> MAPSERVER: Copyright (c) 1996-2001 Regents of the University of Minnesota PHP/MapScript & GMAP75: Copyright (c) 2000-2002, DM Solutions Group TCPDF: Copyright (c) Nicola Asuni, based on FPDF Copyright (c) Olivier Plathey PHP PEAR: Copyright (c) 1997-2002 The PHP Group SLIDER.JS: Copyright (c) by Mark Wilton-Jones 12-13/10/2002 WZ_JSGRAPHICS.JS: Copyright (c) 2002-2004 Walter Zorn ka-map: Copyright (c) 2005, DM Solutions Group Inc. SORTTABLE.JS: Copyright (c) 1997-date Stuart Langridge jQuery JavaScript framework, (c) 2006 John Resig jQuery Dimensions plugin (c) Brandon Aaron jQuery plugin Treeview 1.2 Copyright (c) 2006 Jörn Zaefferer, Myles Angell jQuery plugin qModal Copyright (c) 2007 Brice Burgess