Pmapper exercise

Mapfile editing

In your mapfile, insert inside the MAP object the EXTEND (minx miny maxx maxy, eg: 1473526 5056000 1537368 5087000) and UNITS tags. You can use the same values used for the MapServer exercise.

Then you can write the SHAPEPATH, SYMBOLSET and FONTSET tags (pay attention to write the correct paths!):

```
SHAPEPATH "../../data/shpf"
SYMBOLSET "../common/symbols/symbols-pmapper.sym"
FONTSET "../common/fonts/fontset.txt"
```

As in the previous mapfile, you can set the IMAGECOLOR (e.g.: 255 255 255). In addition in this case you have to add 2 tags:

```
IMAGETYPE png
FORMATOPTION “INTERLACE=OFF”
```

The first tag sets the output format to generate (gif, png, jpe, wbmp, gtiff, swf or user defined), the second is used to specify if the output images should be interlaced or not.

Now we have to declare the **PROJECTION** of the map:

```
PROJECTION
   "+proj=tmerc  +lat_0=0  +lon_0=9  +k=0.9996  +x_0=1500000  +y_0=0  +ellps=intl
+units=m +no_defs  no_defs"
END
```

(NOTE: write the text between “” in the same line)

(In the example the projection set is the Gauss Boaga Rome40 one. In case of UTM WGS84 you can use: #"+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs no_defs").

Inside the **WEB** object (remember to close it with the proper END!) insert the filename of the TEMPLATE file to use in presenting the results to the user in an interactive mode, the IMAGEPATH (path to the temporary directory fro writing temporary files and images. Must be writable by the user the web server is running as. Must end with a / or depending on your platform) and the IMAGEURL (base URL for IMAGEPATH. This is the URL that will take the web browser to IMAGEPATH to get the images):

```
WEB
   TEMPLATE "map.phtml"
   IMAGEPATH "/usr/local/geo_temp/"
   IMAGEURL "/tmp/"
END
```

The **REFERENCE** object has to be filled with the characteristics of the reference map that you want to use, as the extent (you should use the same extent of the map), the image file (e.g.: “ref.png”), the size and colors:

```
REFERENCE
   EXTENT minx miny maxx maxy     (NOTE: replace with your extent!)
   IMAGE "../../images/ref.png"
   SIZE 200 100
   COLOR -1 -1 -1
   OUTLINECOLOR  255  0 0
END
```

At this point you can start to insert the **LAYERS** you want to display. For the p.mapper example you are asked to insert these layers:

- **DTM_20m (raster)**

Set this layer on. Since this is a WMS layer (provided by the Lombardia Region) we have to insert the proper parameters for the connection to the service:

```
CONNECTION
"http://www.cartografia.regione.lombardia.it/ArcGIS93/services/wms/dtm20_wms/
MapServer/WMSServer"
CONNECTIONTYPE WMS
```

(NOTE: write the text between "" on the same line)

Then you have to insert the tag TEMPLATE "void" to allow the queries also when the layer is on. Then in the METADATA object you have to insert some parameters for the WMS service, as the reference system, the name of the layer you want to get, the server version and the format. You can set also the DESCRIPTION parameter that is the name that you will see in the legend (e.g.: "DTM 20m"):

```
TEMPLATE "void"
METADATA
        DESCRIPTION "DTM 20m"
        "wms_srs"           "EPSG:3003"
        "wms_name"           "0"
        "wms_server_version"  "1.1.1"
        "wms_format"         "image/png"
END                    (END of the METADATA object, then END also the layer!)
```

See http://mapserver.org/ogc/wms_client.html as a reference.

(http://www.cartografia.regione.lombardia.it/geoportale/DiscoveryServlet)

- **Ortophoto (raster)**

Also in this case it is a WMS layer:

```
CONNECTION
"http://www.cartografia.regione.lombardia.it/ArcGIS93/services/wms/ortofoto_wms/MapServer/WMSServer"
CONNECTIONTYPE WMS
TEMPLATE "void"
METADATA
        "wms_srs"            "EPSG:3003"
        "wms_name"           "0"
        "wms_server_version" "1.1.1"
        "wms_format"         "image/png"
END
```

- **CTR (raster)**

Again a WMS layer:

```
CONNECTION
"http://www.cartografia.regione.lombardia.it/ArcGIS93/services/wms/ctr_wms/MapServer/WMSServer"
CONNECTIONTYPE WMS
TEMPLATE "void"
METADATA
        DESCRIPTION "CTR - scale 1:10000"
        "wms_srs"            "EPSG:3003"
        "wms_name"           "0"
        "wms_server_version" "1.1.1"
        "wms_format"         "image/png"
END
```

- **DTM_20m_m (raster)**

In this case you are going to use a WMS layer provided by the Environmental Ministry, so the connections parameter become:

```
CONNECTION "http://wms.pcn.minambiente.it/cgi-bin/mapserv.exe?map=/ms_ogc/
service/dtm_20m_f32.map"
CONNECTIONTYPE WMS
TEMPLATE "void"
METADATA
        DESCRIPTION "DTM 20m"
        "wms_srs"            "EPSG:32632"
        "wms_name"           "DTM_20M_f32"
        "wms_server_version"  "1.1.1"
        "wms_format"         "image/png"
END
```

Notice that in this case the layer is provided in UTM WGS84 - 32N zone (EPSG code: 32632), so it is reprojected by MapServer. You can clearly see the errors due to this reprojection, that's why it is important to keep in mind the reference system of your project and check if it is provided by the WMS/WFS service.

(See: http://www.pcn.minambiente.it/PCNDYN/catalogowms.jsp?lan=en)

- **municipalities (polygon)**

The municipalities layer is a 'TYPE polygon' one. Since it is a shapefile that you can find in the default data folder, you can write just the name of the file (in this case "municipalities") with no extension in the DATA tag.

We want to display a label for each municipality that shows the name and we want to represent the municipalities with different colors according to the district.

As in the MapServer case, we can use:

```
LABELITEM "Name"
CLASSITEM "District"
LABELMAXSCALEDENOM 100000
```
(to show the labels only below a certain scale denom)

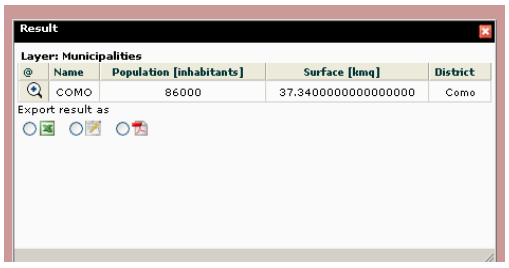
After those tags we can insert the CLASS objects:

```
CLASS
      NAME "District of Como"          or Varese
      EXPRESSION "Como"                or Varese
      LABEL
        TYPE bitmap
        SIZE small
        POSITION …      (e.g.: AUTO)
        COLOR …
      END
# Style to be used for the feautures of the layer.
      STYLE
        COLOR …
        OUTLINECOLOR …
     END
END
```

Then we have to insert the parameters related to the queries. We can use a “void” template and set the TOLERANCE in the selection equal to 6 (pixels).

In the METADATA object, besides the DESCRIPTION tag, you can indicate which fields you want to show in the query result and the headers of the table. E.g.:

```
TEMPLATE "void"
TOLERANCE 6
METADATA
      DESCRIPTION "Municipalities"
     RESULT_FIELDS "NAME,POPULATION,SURFACE,DISTRICT"
     RESULT_HEADERS "Name,Population [inhabitants],Surface [kmq],District"
END
```


Result

Layer: Municipalities

@	Name	Population [inhabitants]	Surface [kmq]	District
🔍	COMO	86000	37.3400000000000000	Como

Export result as

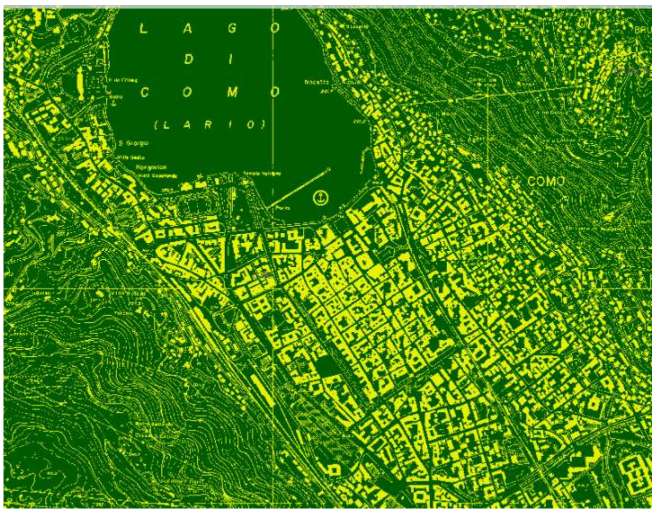
• Como_CTR (raster)

This is a 'TYPE raster' layer that wants to represent the tiff file stored in the 'raster' folder:

```
DATA "../../data/raster/B4A5.tif"
```

We want to show it just at a scale factor lower than 50000, so we write:

```
MAXSCALEDENOM 50000
```



We can recolor the map pixels from black/ white to green/yellow. Since it is a tiff file we can do it simply inserting those lines:

```
CLASSITEM "pixel"
CLASS
     EXPRESSION ([pixel]=0)
     STYLE
       COLOR 0 91 0
     END
END
```

```
CLASS
    EXPRESSION ([pixel]=1)
    STYLE
      COLOR 231 241 0
    END
END
```

As before, with the tag DESCRIPTION in the METADATA object we can write the name in the legend, for example: “Como CTR”.

- **Rivers (raster)**

Again a WMS layer from Environmental Ministry server. This time the url is:

```
CONNECTION "http://wms.pcn.minambiente.it/cgi-bin/mapserv.exe?map=/ms_ogc/
service/astefluviali_32.map"
```

and the METADATA object:

```
METADATA
        "wms_srs"            "EPSG:32632"
        "wms_name"           "aste_fluviali_f32"
        "wms_server_version"  "1.1.1"
        "wms_format"         "image/png"
END
```

- **Railways (line)**

This is a ‘TYPE line’ WFS layer provided by the Environmental Ministry server.
We specify the projection:

```
PROJECTION
        "init=epsg:32632"
END
```

And then the connection parameters:

```
CONNECTIONTYPE WFS
CONNECTION "http://wms.pcn.minambiente.it/cgi-bin/mapserv.exe?map=/ms_ogc/
wfs/ferrovie_wfs_f32.map"
TEMPLATE "void"
METADATA
          "wfs_typename"       "linee_ferroviarie_f32"
          "wfs_version"        "1.0.0"
          "wfs_request_method"  "GET"
          "wfs_connectiontimeout" "60"
          "wfs_latlongboundingbox" "486000 5055905 524999 5087904"

END
```

(See as a reference: http://mapserver.org/ogc/wfs_client.html)

In case of this WFS service, you can set the "wfs_latlongboundingbox" parameter with the bounding box coordinates in the reference system of the service (e.g.: UTM WGS84 - 32N) because it is a service provided for the entire national territory so the layer cover a very large area and it could be heavy to load.

We can decide to represent it with a style that reminds the user a railway, so for example:

```
SYMBOL
  NAME 'rail'
  TYPE ELLIPSE
  POINTS
   1 1
  END
  STYLE
   10 10 10 10
  END
 END
```

Pay attention! you have to insert this new symbol in the **/config/common/symbols/symbols-pmapper.sym** file, nested in the SYMBOLSET object.

Coming back to the layer in the mapfile, we can use the class object:

```
CLASS
        STYLE
           SYMBOL "rail"
           COLOR 51 51 51
           SIZE 2
          END
END
```

- **lakes (polygon)**

This layer represents the shapefile "Lago_poly" stored in the default folder. We want to allow the query in such a way that the user can obtain a table with two columns: 'Name' and 'Surface [mq]':

```
    METADATA
          DESCRIPTION "Lakes"
          RESULT_FIELDS "NOME_LG,AREA"
          RESULT_HEADERS "Name,Surface [mq]"
    END
```

- **stations (point)**

This file is stored in the folder 'GaussBoaga' inside the default one:

```
    DATA 'GaussBoaga/stations'
```

and we want to have a different representation according to the company that provides the service in that station and labels to show the station name.

The two companies have different COMP_CODE so we can use this parameter to divide the two categories:

```
CLASSITEM "COMP_CODE"
LABELITEM "NAME"
LABELMAXSCALEDENOM 100000
CLASS
    NAME "FNM Stations"
    EXPRESSION "FNM"
    LABEL
         TYPE bitmap
         POSITION auto
         SIZE small
         COLOR 0 0 255
   END
    STYLE
           SYMBOL '../common/symbols/train_blue.png'
    END
  END
  CLASS
    NAME "FS Stations"
    EXPRESSION "FS"
    LABEL
      TYPE bitmap
      POSITION auto
      SIZE small
      COLOR 255 0 0
    END
    STYLE
      SYMBOL '../common/symbols/train_red.png'
    END
  END
  TEMPLATE "void"
  TOLERANCE 6
```

In addition, we want that when the user clicks on the 'COMPANY' filed in the query table the wiki page about this company is shown in another tab of the browser. In the 'METADATA' object we add a line:

```
METADATA
     DESCRIPTION "Stations (GB - Roma40)"
     RESULT_HYPERLINK "COMPANY"
     RESULT_FIELDS "NAME,COMPANY,COMP_CODE,LINE"
     RESULT_HEADERS "Name,Company,Company Code,Line"
END
```

In the file **config/default/custom.js** we have to add:

```
$.extend(PM.Custom,
{
   // Sample Hyperlink function for result window
   openHyperlink: function(layer, fldName, fldValue) {
      switch(layer) {
         case 'stations':
            if (fldName == 'COMPANY') {
               window.open('http:/' + '/en.wikipedia.org/wiki/' + fldValue);
            }
            break;

         default:
            alert ('See function openHyperlink in custom.js: ' + layer + ' - ' +
fldName + ' - ' + fldValue);
      }
   }
```

- **stations_utm (point)**

Also this layer represents train stations, but in this case the shapefile is projected in UTM WGS84 - 32N and it is stored directly in the default data folder. Turning on both the layers we can notice directly the displacement introduced in the representation due to the reprojection.

```
LAYER
    NAME "stations_utm"
    STATUS off
    DATA 'stations'
    TYPE point
    CLASSITEM "COMP_CODE"
    LABELITEM "NAME"
    LABELMAXSCALEDENOM 100000
    PROJECTION
        "init=epsg:32632"
    END
    CLASS
      NAME "FNM Stations"
      EXPRESSION "FNM"
      STYLE
        symbol 'square'
        color 0 0 255
        size 10
      END
    END
    CLASS
      NAME "FS Stations"
      EXPRESSION "FS"
      STYLE
        symbol 'square'
        color 255 0 0
        size 10
      END
    END
    TEMPLATE "void"
    TOLERANCE 6
    METADATA
      DESCRIPTION "Stations (UTM - WGS84)"
      RESULT_FIELDS "NAME,COMPANY,COMP_CODE,LINE"
      RESULT_HEADERS "Name,Company,Company Code,Line"
     END
   END
```

- **alluvional-cones (line)**

As in the MapServer example, you can add a MapInfo format ("ogr" folder): alluvional-cones.tab. In this case you have to specify the connection type and the path to the file:

```
CONNECTIONTYPE OGR
CONNECTION "../../../data /ogr/alluvional-cones.tab"
```

We can choose to display the layer only above a certain scale and with a certain style:

```
MAXSCALEDENOM 100000
CLASS
   NAME "Alluvional cones"
   STYLE
          COLOR 0 215 240
   END
END
METADATA
   DESCRIPTION "Alluvional cones"
END
```

ADD A LOGO (for the webGIS project)

The procedure to add a logo (a watermark on the map) is the same than adding another layer. It has to be placed above all other layers, so at the end of the mapfile INSIDE the MAP object (before the last END).

It is a particular layer with just a feature (a point) represented by a symbol (the bitmap image, that is the logo itself). To do so we have to:

- insert the **symbol** in the file "../common/symbols/**symbols-pmapper.sym**":

```
SYMBOL
  NAME "logo"
  TYPE PIXMAP
  IMAGE "logo.png"
END
```

- insert the **layer** in the **mapfile**:

```
LAYER
  NAME "credits"
  STATUS DEFAULT
  TRANSFORM lr
  TYPE ANNOTATION
  FEATURE
    POINTS
      -70 -60
    END
    TEXT " "
  END
  CLASS
    STYLE
      SYMBOL "logo"
      #SIZE 36
    END
    LABEL
      TYPE BITMAP
      POSITION UL
      COLOR 0 0 0
      BUFFER 5
    END
  END
END
```

The parameter ‘TRASFORM’ tells MapServer to pass from your coordinates to image coordinates from the ‘lr’ (lower right) corner. In this case the logo is then placed at coordinates -70 -60.

‘config_default.xml’ editing

Inside the <ini> tag are nested all the configurations tags that we will use to customize the webGIS.

First of all, we can insert some plugins inside the <ini><pmapper> tags (after <debugLevel>3</debugLevel>):

http://svn.pmapper.net/trac/wiki/AvailablePlugins

PLUGINS:

- **<plugins>export</plugins>**

Plugin to export query results to various formats. It adds export radio buttons to query result display. Currently implemented format: XLS, PDF, CSV, SHP (experimental).

Define the formats you want to offer by adding in *config_default.xml* file:

```
<pluginsConfig>
....
          <export>
             <formats>CSV</formats>
             <formats>PDF</formats>
             <formats>SHP</formats>
          </export>
....
</pluginsConfig>
```

NOTE: Export to XLS requires the installation of the PEAR modules Spreadsheet_Excel_Writer and OLE.

- **<plugins> scalebar </plugins>**

DHTML scalebar based on a library from Tim Schaub of CommEn Space.

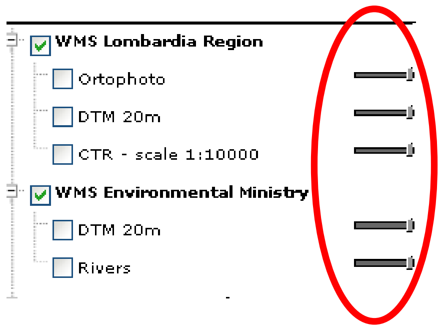
Enable the plugin by adding a line in *config_default.xml* file and add two <div> in the ***map.phtml***, like

```
<div id="scaleReference">
   <div id="scalebar"></div>
</div>
```

Style of the scale bar can be defined via CSS file.

- **<plugins>transparency2</plugins>**

Add a slider at the right side of each layer / group of layers in the Table Of Contents. The sliders could represent either transparency or opacity percentage. Transparency plugin have to be present in p.mapper installation, but not necessarily activated.

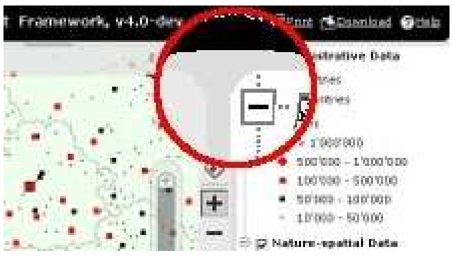


Enable the plugin by adding a line in *config_default.xml* file, then you can specify if sliders have to represent opacity or transparency by adding in *config_default.xml* file:

```
<pmapper>
   <ini>
....
      <pluginsConfig>
....
         <transparency2>
            <useOpacity>off</useOpacity>
         </transparency2>
....
      </pluginsConfig>
   </ini>
</pmapper>
```

If you don't specify it, the default setting is to represent transparency.

- **<plugins>roundedboxes</plugins>**



This small plugin add styled corners to the UI. Enable the plugin by adding a line in *config_default.xml* file.

CONFIGURATIONS:

After the plugins we can write some configuration tags such as:

```
<config>
        <pm_config_location>default</pm_config_location>
        <pm_javascript_location>javascript</pm_javascript_location>
        <pm_print_configfile>common/print.xml</pm_print_configfile>
        <pm_search_configfile>inline</pm_search_configfile>
</config>
```

Config_location: directory that contains the mapfile

Javascript_location: directory of the javascript scripts

Print_configfile: relative path to the print configuration xml file

Search_configfile: in this case we will write the search configurations inside the *config_default.xml* file, so 'inline'.

MAP:

Nested in the <map> tag we can write all the configurations concerning the map that we want to represent:

```
<map>
        <mapFile>como.map</mapFile>                          → mapfile to use
        <tplMapFile>common/template.map</tplMapFile>         → template map
```

Then we can subdivide your layers in different categories. Each category is nested inside <categories> and it is characterized by a name. E.g.:

```
<categories>
        <category name="WMS Lombardia Region">
           <group>Ortophoto</group>
           <group>DTM_20m</group>
           <group>CTR</group>
        </category>
…
</categories>
```

If you use one of the category names already written in p.mapper (stored in ‘incphp\locale’ you can find some php files, one for each language) it will be directly translated in the language that you choose. The category names already present are:

```
$_sl['cat_admin'] = 'Administrative Data';
$_sl['cat_infrastructure'] = 'Infrastructur';
$_sl['cat_nature'] = 'Nature-spatial Data';
$_sl['cat_raster'] = 'Raster Data';
$_sl['cat_satimages'] = 'Satellite Imagery';
$_sl['cat_srtm'] = 'SRTM data';
```

You can decide also to write other lines in the php file of your language in order to let p.mapper know the translation and show it when you change the language. (E.g.: I’ve add the line “$_sl['WMS Lombardia Region'] = 'WMS Regione Lombardia';” to my ‘language_it.php’ file).

Each layer has to be written into a <group> tag. Remember that you have to use the names of the layer written in the mapfile.

Once you have inserted all the categories, you have to write all you layer groups inside the tag <allGroups>:

```
<allGroups>
    <group>Como_CTR</group>
    <group>Rivers</group>
    <group>lakes</group>
…
  </allGroups>
```

Then you can specify which layers you want to show as a default when the user enter the webGIS:

```
<defGroups>
  <group>lakes</group>
  <group>municipalities</group>
</defGroups>
```

(Groups/Layers can be mutually disabled, so if one is clicked the other one will be disabled:

```
<mutualDisableList>
  <group>layer1</group>
  <group>layer2</group>
</mutualDisableList>)
```

You can define some other settings, such as:

<layerAutoRefresh>1</layerAutoRefresh> → **automatically refresh map when layers selection changed (0/1)**

<imgFormat>png</imgFormat> → **image format for map and legend icons, like png, agg_png, jpeg. Use the same format defined as OUTPUTFORMAT in the mapfile (image format for legend icons)**

<altImgFormat>jpeg</altImgFormat> → **alternative Image format for map useful eg. for imagery data**

<sliderMax>max</sliderMax>
<sliderMin>100</sliderMin>
} → **define start and end scale for zoom slider. Adapt to specific extents of the datasets or use 'max' for automatically calculate from map file**

</map>

QUERY:

In the query part we can specify all the parameters related to the query we want to allow.

<query>

<limitResult>300</limitResult> → **limit for results of selection with select tool or search**

<highlightColor>0 255 255</highlightColor> → **highlight color to identify/search zoom in RGB values**

<highlightSelected>1</highlightSelected> → **defines if SELECT function causes feature highlight**

<autoZoom>nquery</autoZoom>
<autoZoom>search</autoZoom>
} → **zoom in automatically after results are displayed possible values: off search nquery (= select)**

<zoomAll>search</zoomAll>
<zoomAll>nquery</zoomAll>
} → **add button 'zoom to All Features' to result table - possible values: off search nquery (= select)**

<infoWin>dynwin</infoWin> → **how to show the query results (identify/search) dynwin: open DHTML window anything else will be used as id of the DOM elment where to place the query result**

<alignQueryResults>1</alignQueryResults> → **automatically align column contents of result tables of queries with regard to data type (default, numeric, currency, ...)**

`<pointBuffer>10000</pointBuffer>`	**→ extent buffer for zoom extent for point layers in queries value in map units**
`<shapeQueryBuffer>0.02</shapeQueryBuffer>`	**→ extent buffer for zoom extent for non-point layers in queries value in a fraction of the original extent**
`</query>`	

USER INTERFACE:

`<ui>`	
`<tocStyle>tree</tocStyle>`	**→ categories Style in TOC (only has effect when useCategories = 1): tree or flat**
`<legendStyle>attached</legendStyle>`	**→ - attached: together with TOC** **- swap: swapping with TOC**
`<useCategories>1</useCategories>`	**→ use categories to thematically group layers categories defined in incphp/custom.php works for TOC setting 'flat' and 'tree'**
`<catWithCheckbox>1</catWithCheckbox>`	**→ use checkboxes for en/disable complete categories with child groups/layers**
`<scaleLayers>1</scaleLayers>`	**→ layer list (TOC) automatically updated according to scale**

`<icoW>18</icoW>`
`<icoH>14</icoH>`
	→ icon Width/Height in pixels

`</ui>`	

LOCALE:

`<locale>`	
`<defaultLanguage>en</defaultLanguage>`	**→ default language**
`<defaultCharset>UTF-8</defaultCharset>`	**→ default character set UTF-8**

```
        <map2unicode>1</map2unicode>
</locale>
```

→ if map file contains non-ASCII characters,eg for layer DESCRIPTION or CLASS names, and is not in UTF (UNICODE) encoding set value to 1

PRINT:

```
<print>
    <pdfres>2</pdfres>

    <printImgFormat>png</printImgFormat>
    <printAltImgFormat>jpeg</printAltImgFormat>

</print>
```

→ PDF print resolution. Factor to increase resolution for better image quality

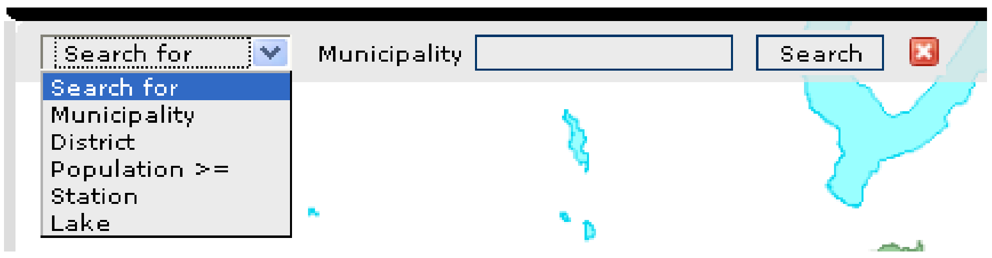
$\}$ **→ print formats**

DOWLOAD:

```
<download>
        <dpiLevels>150</dpiLevels>
        <dpiLevels>200</dpiLevels>
        <dpiLevels>300</dpiLevels>
</download>
```

→ DPI levels for map download - used in downloaddlg.phtml

SEARCH ITEM:



In p.mapper you have the possibility to implement an easy searching tool. To do so, you can write an xml file with all the settings or write them directly in *config_default.xml*.

We want to define a search for:

- **Municipality**: name of the municipality (layer: municipalities, field: "NAME")

- **District**: name of the district (layer: municipalities, field: "DISTRICT")
- **Population**: number of inhabitants in a municipality (layer: municipalities, field: "POPULATION")
- **Station**: name of a train station (layer: stations, field: "NAME")
- **Lake**: name of the lake (layer: lakes, field: "NOME_LG")

E.g.1: Municipality

```
<searchitem name="municipalities" description="Municipality">
          <layer type="shape" name="municipalities">
             <field type="s" name="NAME" description="Municipality"  wildcard="0" />
          </layer>
</searchitem>
```

In the tag **<searchitem>** you have to insert the parameters '**name**' = unique identifier, typically the same as layer name if it is univoque, and '**description**' = name visible in GUI (see figure above).

Nested inside you have the **<layer>** tag, in which you have to define the '**type**' = data source type (possible values: 'shape', 'postgis', 'xy', 'oracle'), and '**name**' = layer name in the mapfile.

Then you have the final **<field>** tag, in which you have the '**type**' = 's' for string field, 'n' for numeric field, '**name**' = field name in dataset, '**description**' = name visible in GUI, '**wildcard**' =

"0": search always uses a 'non-exact' pattern matching;

"1": requires that the user explicitly adds "*" for wildcards to his search string

"2": exact search, usually just appropriate for 'suggest' or 'options'.

E.g.2: Population

```
<searchitem name="population" description="Population >=">
   <layer type="shape" name="municipalities">
        <field type="n" name="POPULATION" description="Population >="
        wildcard="0" compare=">="/>
   </layer>
</searchitem>
```

In this case we have add the parameter ‘compare’ because we want to search the municipalities with a population equal or greater than a certain value inserted by the user. Another way could be to define the operators that we want to use:

```
<definition type="operator">
            <option name=">" value=">" />
            <option name="=" value="=" />
            <option name="<" value="<" />
</definition>
```

‘uielements.php’ editing

The purpose of this php file is to create User Interface HTML elements, such as:

- map zone
- toolbar
- TOC and legend container
- tool links
- tabs for TOC/legend containers
- reference map
- search form
- scale
- zoom slider
- header/footer
- coordinates display

We are not going to see into detail all the script, just a little add in the footer function. For instance, we want to add two flags with a link in order to choose the language of our webGIS:

```
<div style=\"float:right;\">
<a href=\"http://webgis.como.polimi.it/GIScourse10/pmapper_utenteXX/map.phtml?
language=it&config=default\">
```

```
<img src=\"images/it.gif\"/></a>
</div>
<div style=\"float:right;\">
<a href=\"http://webgis.como.polimi.it/GIScourse10/pmapper_ utenteXX /map.phtml?
language=en&config=default\">
<img src=\"images/en.gif\"/></a>
</div>
```

NOTE: substitute 'XX' with you user number.

In the same way you can add the logo of the Politecnico with the url to the home page.

Main references:

http://svn.pmapper.net/trac/wiki/XmlFileSettings
http://svn.pmapper.net/trac/wiki/DocManualsearch