p.mapper 4

Alcuni appunti a proposito della versione 3 di p.mapper sono ancora validi, vedere la pagina **p.mapper 3**.

Link documentazione

- **Documentazione p.Mapper**
- **User Manual**
- **Download pacchetti**
- **Debian Packages**

Scaletta

- Impostare l'ambiente PHP
  - Installare **php5-mapscript**
  - Installare **php-pear**
  - Verificare con `phpinfo()` presenza supporto a **iconv**
  - Gestione estensioni **.phtml**, verificare `/etc/apache2/mods-enabled/php5.conf`
  - Verificare con `phpinfo()` il caricamento di **mapscript.so**. Debian ha il file `/etc/php5/apache2/conf.d/mapscript.ini`. Solo in caso di bisogno mettere `extension=php_mapscript.so` in **`/etc/php5/apache2/php.ini`**, lasciando pure disabilitata la funzione **`dl()`**: `enable_dl = Off`.
  - Verificare la gestione degli errori.
  - Verificare con `phpinfo()` la configurazione delle sessioni (Debian fa le cose per bene):
    - `session.save_path`: deve puntare ad una directory scrivibile dal server web
    - `session.use_trans_sid = 0`: per essere compatibili *W3C XHTML Strict*
    - `session.auto_start = Off`
- Scompattare gli archivi.
- Impostare i permessi.
- Configurazione di p.mapper:
  - **`config/config_default.xml`**
    - `<ini><config><pm_config_location>` Directory che contiene il file .map
    - `<ini><map><mapFile>` Il file .map in uso
    - `<ini><map><categories>`: Categorie tematiche dei layer o dei gruppi di layer
    - `<ini><map><allGroups>`: Layer o gruppi di layer
    - `<ini><map><defGroups>`: Layer o gruppi di layer visibili per default
    - `<ini><map><imgFormat>`: Usare il nome definito in `OUTPUTFORMAT` nel file .map
    - `<ini><map><sliderMax>`: Scala massima (zoom out). Usare *max* per calcolare autom. dal file .map
    - `<ini><map><sliderMin>`: Scala minima (zoom in)
  - **`config/default/js_config.php`** (modificare se necessario)
  - **`config/default/custom.js`** (modificare se necessario)
- Configurazione del file .map (nella demo è **`pmapper_demo.map`**):
  - **`config/default/test.map`**
    - Impostare la **`MAP RESOLUTION`** su 96

- Togliere la **SCALEBAR**
- Definire un **SYMBOL NAME 'circle'**
- Impostare **EXTENT** alla massima estensione desiderata
- Copiare la reference map in **images/**

Errori PHP

Per il debug di applicazioni p.Mapper è importante poter intercettare gli errori PHP. Verificare le impostazioni in **/etc/php5/apache2/php.ini**:

```
display_errors = {On|Off}
log_errors = {On|Off}
;error_log = /var/log/php/error.log
;error_log = syslog
```

In generale gli errori vanno cercati in:

- **/var/log/apache2/error.log**
- **/var/log/php/error.log**
- **/var/log/syslog**

Un eventuale file di log separato (es. `/var/log/php/error.log`) viene generato con i privilegi dell'utente **www-data** e deve essere ruotato ad esempio con **logrotate(8)**.

Provare ed eseguire una divisione per zero in uno script PHP e controllare dove viene mostrato il messaggio di errore.

Debug p.mapper

Vedere i consigli del **wiki**.

Abilitare gli errori di PHP in un file separato con una configurazione del tipo:

```
log_errors = On
error_log = /var/log/php/error.log
```

Poi abilitare **<debugLevel>3</debugLevel>** nel **config_default.xml**. In questo modo p.mapper genera un suo file di log nella stessa directory indicata da `error_log` del PHP, con il nome **pm_debug.log**.

Permessi

p.mapper ha bisogno che la directory **images/legend/** sia scrivibile dal server web (`www-data` in Debian). Questo ovviamente si aggiunge alla necessità di MapServer di una directory scrivibile per i file temporanei. Esempio:

```
chgrp -R www-data images/legend
```

```
chmod 2775 images/legend
chmod 664 images/legend/*
```

Impostazioni extra per Apache

Proteggere i file .map

Verificare che non sia possibile leggere dal browser il file .map chiamando un URL del tipo **`http://host/pmapper/config/default/pmapper_demo.map`**. Il rischio è che siano visibili le credenziali per accedere al database. p.mapper installa il file **`.htaccess`** che aggiunge una direttiva per impedire la visualizzazione di tutti i file .map, verificare che Apache onori tale direttiva.

Disabilitare i cookie di sessione

In generale **la sessione p.mapper viene salvata tramite un cookie** associato all'indirizzo IP del server. Se lo stesso server ospita diverse installazioni p.mapper può essere un problema, conviene disabilitare l'uso dei cookie di sessione.

Aggiungere alla configurazione di Apache (**`/etc/apache2/sites-availabe/default`** in Debian) le seguenti righe:

```
<Directory "/var/www/pmapper/">
    php_flag session.use_cookies off
</Directory>
```

Verificare con `phpinfo()` eseguito nella directory di p.mapper che **session.use_cookies** sia **Off**.

Note su config_default.xml

È possibile definire **configurazioni alternative**. Se ad esempio si crea il file **`config_alt.xml`** sarà possibile aggiungere all'URL il parametro **`config=alt`**.

Per richiamare p.mapper preimpostando alcuni valori (estensione, layer attivi, ecc.) è possibile aggiungere all'ULR alcuni parametri, consultare la pagina wiki **Miscellaneous Functions**.

Per avere l'interfaccia in **altra lingua** aggiungere all'URL il parametro **`language=it`**, oppure modificare **`defaultLanguage`** in `config_default.xml`.

Il **nome della categoria** viene scelto da una lista predefinita (es. `cat_infrastructure`), in tal modo il nome viene nazionalizzato automaticamente. Per vedere le categorie esistenti consultare il file `incphp/locale/language_it.php`.

I valori **minimo e massimo di scala** impostabili con lo slider sono influenzati dai valori del file .map: **`MAP.EXTENT`**, **`MAP.WEB.MINSCALE`** e **`MAP.WEB.MAXSCALE`**.

L'etichetta visualizzata per il gruppo o per il singolo layer viene definita nel file .map, tramite il tag **`LAYER.METADATA.DESCRIPTION`**. I gruppi vengono definiti nel file .map con il tag **`LAYER.GROUP`**.

Query su layer

Esistono tre strumenti di query, sono associati ai pulsanti con la lettera **i**:

1. **Identifica**: identifica tutti gli elementi dei layer interrogabili, vicini al punto in cui si fa click.
2. **Seleziona**: si sceglie dal menu a tendina il layer da interrogare e poi si seleziona un rettangolo sulla mappa.
3. **Tooltip**: si sceglie dal menu a tendina il layer da interrogare e poi si passa il mouse sopra la mappa.

Per poter **eseguire query su un layer** bisogna:

- Definire nel LAYER un **TEMPLATE**, anche vuoto.
- Definire in LAYER.METADATA l'elenco dei campi che si vuole restituiti dalla query, con il tag **RESULT_FIELDS** (lista separata da virgole).
- Definire in LAYER.METADATA le intestazioni delle colonne con il tag **RESULT_HEADERS** (lista separata da virgole).
- Eventualmente definire la **TOLERANCE** nel LAYER, aumentando il valore predefinito di 3 pixel per layer di tipo POINT e LINE.

Ad esempio:

```
LAYER
  NAME "entita_lineari"
  STATUS ON
  TYPE LINE
  DATA "shape/geofesta2008/firenze10k/s_275040/275040el"
  CLASS
    NAME "Entità lineari"
    COLOR 120 120 120
  END
  TEMPLATE "void"
  TOLERANCE 6
  METADATA
    DESCRIPTION    "Elementi CTR"
    RESULT_FIELDS  "FOGLIO,CODICE,RECORD,TOPON,DIMENS,IDENTIF"
    RESULT_HEADERS "Foglio,Codice,Record,Toponimo,Dimensione,Identificativo"
  END
END
```

Query su layer con JOIN su database

Nel risultato di una query è possibile integrare gli attributi del layer stesso con **altri campi contenuti in una tabella di un database**, eseguendo in effetti una **JOIN** tra un field del layer e un field della tabella.

Sono necessari alcuni pacchetti aggiuntivi, alcuni **non** presenti nel repository Debian Lenny:

php5-pgsql	Da pacchetto Debian

php-mdb2	Installare con **pear install 'pear/MDB2'**
php-mdb2-driver-pgsql	Installare con **pear install 'pear/MDB2#pgsql'**

NOTA1: I pacchetti installati con `pear install` non risultano al gestore dei pacchetti Debian, vengono scaricati e scompattati in **/usr/share/php/**. Tutti i pacchetti sopra elencati sono invece disponibili in **Debian Squeeze**.

NOTA2: il presente esempio fa uso di PostgreSQL, per interfacciarsi con altri database occorre installare il relativo driver MDB2.

Le istruzioni per eseguire l'interrogazione sul database sono contenute nel file .map, nella sezione LAYER, METADATA, **RESULT_JOIN**. Ogni campo restituito dalla query sul DB viene aggiunto alle colonne della query eseguita sul layer, pertano si devono indicare dei **RESULT_HEADERS** aggiuntivi. Esempio:

```
LAYER
  NAME "entita_lineari"
  STATUS ON
  TYPE LINE
  DATA "shape/geofesta2008/firenze10k/s_275040/275040el"
  TEMPLATE "void"
  METADATA
    DESCRIPTION    "Elementi CTR"
    RESULT_FIELDS  "FOGLIO,CODICE,RECORD,TOPON,DIMENS,IDENTIF"
    RESULT_HEADERS
"Foglio,Codice,Record,Toponimo,Dimensione,Identificativo,Field1,Field2"
    RESULT_JOIN
"pgsql://mapserver:MySecret@127.0.0.1/mapserver||tab_dati@id_record@0@field1
,field2||RECORD||0"
  END
  ...
END
```

Ovviamente per far funzionare l'esempio sopra, nel database deve esistere la tabella **tab_dati** con le colonne **id_record**, **field1** e **field2**.

Per aiutare in fase di debug, se si utilizza PostgreSQL, può essere utile attivare temporaneamente l'opzione **log_statement = 'all'** in `/etc/postgresql/8.3/main/postgresql.conf` e poi controllare il contenuto di `/var/log/postgresql/postgresql-8.3-main.log`.

Hyperlink nei risultati di una query

E' possibile trasformare uno o più campi del risultato della query in hyperlink cliccabili. Bisogna aggiungere **una riga nel file .map e modificare una funzione JavaScript**, come indicato sul **wiki**. Esempio:

```
  METADATA.
    "DESCRIPTION"      "Cities"
    "RESULT_FIELDS"    "NAME, ISO2_CODE, POPULATION, GTOPO30"
    "RESULT_HEADERS"   "Name,Country,Inhabitants,Altitude"
```

```
    "RESULT_HYPERLINK" "NAME"
    "LAYER_ENCODING"   "UTF-8"
  END  # Metadata..
```

La funzione da modificare **openHyperlink** si trova nel file **config/default/custom.js**.

Ricerca per parola chiave

p.mapper consente la **attribute search**, cioè la ricerca di un'entità geometrica per parola chiave, anche su campi multipli. Le ricerche possibili si definiscono nel file **config/config_default.xml**, ecco due esempi di ricerca, uno su shapefile con campo singolo ed uno su layer PostGIS su due campi:

```
<searchlist version="1.0">
    <dataroot>$</dataroot>
    <searchitem name="strade" description="Strade">
        <layer type="shape" name="tratti_stradali">
            <field type="s" name="TOPONIMO_S" description="Strada"
wildcard="0" />
        </layer>
    </searchitem>
    <searchitem name="civici" description="Civici">
        <layer type="postgis" name="civici">
            <field type="s" name="toponimo_s" description="Toponimo"
wildcard="0" />
            <field type="s" name="descrizion" description="Civico"
wildcard="1" />
        </layer>
    </searchitem>
</searchlist>
```

layer.type	**shape**, **postgis**, **xy** or **oracle**.
field.type	**s** for string field **n** for numeric field
field.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'

Verificare anche il tag **<sql>** che dovrebbe poter essere incluso in **<layer>** e consentire di specificare la sintassi per la query SQL.

Nel **mapfile** assicurarsi di aver impostato un **TEMPLATE** (anche vuoto), i **METADATA.RESULT_FIELDS** e i **METADATA.RESULT_HEADERS**. Per i layer di tipo PostGIS conviene impostare la direttiva **DATA** come segue, che consente di specificare sia la chiave primaria che i campi richiesti nella query:

```
LAYER
  NAME "civici"
  STATUS DEFAULT
  TYPE POINT
```

```
  CONNECTIONTYPE postgis
  CONNECTION "user=catasto password=**** dbname=catasto host=127.0.0.1"
  DATA "the_geom from (select gid, toponimo_s, descrizion, the_geom from
civici) AS foo USING UNIQUE gid"
  TEMPLATE "void"
  ...
  METADATA
    DESCRIPTION "Numeri civici"
    RESULT_FIELDS  "toponimo_s,descrizion"
    RESULT_HEADERS  "Strada,Numero civico"
  END
```

Il risultato della ricerca viene visualizzato in un pop-up; cliccando sull'icona della lente di ingrandimento la mappa zoomma automaticamente sulla geometria selezionata. Se si tratta di un punto conviene definire in **`config/config_default.xml`** il parametro **pointBuffer**, cioè l'area di zoom in unità di mappa.

La ricerca su campi di tipo **postgis** potrebbe generare una query che fa riferimento al campo inesistente **`oid`** (il campo `oid` è stato rimosso da PostgreSQL a partire dalla versione 8.1). In tal caso può servire mettere nel file .map una direttiva DATA che esplicita la chiave primaria della tabella:

```
  CONNECTIONTYPE POSTGIS
  CONNECTION "user=username password=secret dbname=database host=localhost"
  #DATA "the_geom from particelle"
  DATA "the_geom from (select * fom particelle) as foo using unique gid"
```

ATTENZIONE: la ricerca su campi multipli su **layer shapefile** ha dei problemi, probabilmente dovuti a qualche bug della funzione **`queryByAttributes()`** di **MapScript**. Pare che non sia possibile combinare in una espressione logica (AND, OR, ...) la ricerca per regular expression.

Rotellina del mouse

Per impostare il comportamento della mouse wheel stile Google (wheel forward = zoom in) bisogna modificare il valore di **`wheelZoomGoogleStyle`** nel codice JavaScript. Si imposta in **`config/default/js_config.php`**:

```
PM.ZoomBox.wheelZoomGoogleStyle = true;
```

Measure/Digitize e Add Point of Interest

Pmapper ha due strumenti che consentono di disegnare sulla mappa: **Measure/Digitize** e **Add Point of Interest**. Per far funzionare quest'ultimo verificare che sia disponibile il font **FreeSans** nella directory indicata da **FONTSET** del mapfile in uso.

Gli oggetti disegnati con questi strumenti sono persistenti solo all'interno della sessione PHP: chiudendo il browser vengono persi.

Personalizzazione

Come effettuare digitizing in modo persistente (insert nel database) ed eventualmente anche editing?

Per **personalizzare lo strumento Add Point** basta ridefinire (ad esempio in `config/default/custom.js`) il metodo **`PM.Dlg.openPoi`** la cui versione originale si trova in **`javascript/src/pm.pdraw.js`**, scrivendo qualcosa del genere:

```
$.extend(PM, {
    Dlg: {
        /**
         * Open popup dialaog for adding POI
         */
        openPoi: function(imgxy) {
            var coordsList = imgxy.split('+');
            var mpoint = PM.ZoomBox.getGeoCoords(coordsList[0],
coordsList[1], false);

            // Round values (function 'roundN()' in 'measure.js')
            var rfactor = 5;
            var px = isNaN(mpoint.x) ? '' : PM.roundN(mpoint.x, rfactor);
            var py = isNaN(mpoint.y) ? '' : PM.roundN(mpoint.y, rfactor);

            alert("Screen: (" + coordsList + ")\nMap: (" + px + ", " + py
+")");
        }
    }
});
```

Allo stesso modo, per **personalizzare lo strumento Measure/Digitize**, è possibile ridefinire il metodo **`PM.Draw.onDigitizedPolygon`**. Con questo metodo si intercetta il doppio click subito dopo che la polilinea è stata chiusa, pertanto si lavora sul poligono.

Se si desidera intercettare il doppio click e agire sulla polilinea è necessario ridefinire il metodo **`PM.Draw.measureDrawSymbols`**, ma questo richiede la duplicazione di molto più codice.