Quantum GIS (QGIS) Web Client

Installation Guide

Monday March 12, 2012

Last Updated: Monday March 12, 2012 Last Change : Monday March 12, 2012

Contents

1 For the terminally lazy

Listing

```
sudo apt-get install apache2 libapache2-mod-fcgid
cp apache-conf/qgis-web-client.conf.templ apache-conf/qgis-web-client.conf
```

Update the paths in the copied file then:

Listing

```
cd /etc/apache2/sites-available/
ln -s <path to apache-conf/qgis-web-client.conf> .
sudo a2ensite qgis-web-client.conf
sudo /etc/init.d/apache2 reload
```

1. Check the symlink in cgi-bin is correct.
2. Check the QGIS libs are in your /etc/ld.so.conf path
3. Copy site/index.xml and check paths match your system OR Modify index.html and point your browser to that

2 Purpose

A WMS based webgis client that makes use of QGIS specific WMS extensions (e.g. highlighting, printing, metadata, etc.). QGIS webclient reads the configuration from the WMS GetCapabilities command and builds the layer tree accordingly. Supports legend graphic, feature info requests and printing.

The client builds on existing Web-GIS libraries OpenLayers and GeoExt, as well as ExtJS 3 for the GUI widgets.

All major browsers should be supported.

3 Installation

Requirements (Server):

- Apache2 - Webserver (Ubuntu: apache2)
- mod-fcgid (Ubuntu: libapache2-mod-fcgid)
- QGIS and QGIS Server (best installed from source)

On ubuntu you can meet these requirements by simply doing:

Listing

```
sudo apt-get install libapache2-mod-fcgid
```

The QGIS server compilation and installation will be covered in the QGIS manual.

For searching:

- python-wsgi for searching (Ubuntu: libapache2-mod-wsgi)
- psycopg2 PostgreSQL db driver (Ubuntu: python-psycopg2)
- webob - Python module providing WSGI request and response objects (Ubuntu: python-webob)

The client part needs svn checkout with the following command: svn co https://svn.osgeo.org/qgis/trunk/qg qgiswebclient

4 Configuration of Client

Global Settings for all projects (make a copy from one of the templates provided):

Listing

```
site/js/GlobalOptions.js
```

Translations (additional languages):

Listing

```
site/js/Translations.js
```

Project settings and index:

Listing

```
site/index.xml or site/index.html
```

Stylesheet of project index:

Listing

```
site/gis-project_listing.xsl
```

Thumbnails for individual projects (if you take the index.xml route):

Listing

```
thumbnails/projectname.png
```

4.1 Configuration of search panels

There are two types of search panels supported, using a direct WMS GetFeatureInfo request or using URL rewriting with a much shorter search URL.

The search panels are configured in `site/js/GlobalOptions.js`.

4.1.1 Using WMS GetFeatureInfo

Listing

```
var simpleWmsSearch = {
  title: "Search continent",
  query: 'simpleWmsSearch',
```


```
    useWmsRequest: true,
    queryLayer: "Country",
    formItems: [
      {
        xtype: 'textfield',
        name: 'name',
        fieldLabel: "Name",
        allowBlank: false,
        blankText: "Please enter a name (e.g. 'africa')"
      }
    ],
    gridColumns: [
      {header: 'Name', dataIndex: 'name', menuDisabled: 'true'}
    ],
    selectionLayer: 'Country',
    selectionZoom: 0
  };
```

- `title`: title of the search tab
- `query`: identifier for this search
- `useWmsRequest`: enabled for WMS GetFeatureInfo request
- `queryLayer`: name of query layer
- `formItems`: list of Ext.form.FormPanel item configs
  - `xtype`: form field type
  - `name`: name of query layer attribute
  - `fieldLabel`: visible text for this field
  - `blankText`: popup text for blank fields
- `gridColumns`: list of Ext.grid.GridPanel column configs to show search results
- `selectionLayer`: name of layer for marking selected results (the same as `queryLayer`)
- `selectionZoom`: zoom level for jump-to when selecting results

Request URL:

When performing a search query using the above configuration, the following get request will be made.

"http://localhost/wms/helloworld?SERVICE=WMS&VERSION=1.1.1& REQUEST=GetFeatureInfo&LA
FEATURE_COUNT=10&INFO_FORMAT=text/xml&SRS=EPSG:4326& FILTER=Country:"name"+=-

4.1.2 Using URL Rewriting

For security and neatness, you may prefer to use rewritten URLs (so that your internal server file paths are not revealed. In that case your options file would contain something like this:

Listing

```
var urlRewriteSearch = {
  title: "Search letter",
  query: 'samplesearch',
  formItems: [
    {
      xtype: 'hidden',
      name: 'query',
      value: 'samplesearch'
    },
    {
      xtype: 'textfield',
      name: 'colour',
      fieldLabel: "Colour",
      allowBlank: false,
      blankText: "Please enter a colour (e.g. 'orange')"
    }
  ],
  gridColumns: [
    {header: 'PKUID', dataIndex: 'pkuid', menuDisabled: 'true'},
    {header: 'Colour', dataIndex: 'colour', menuDisabled: 'true'}
  ],
  selectionLayer: 'Hello',
  selectionZoom: 1
};
```

- `title`: title of the search tab
- `query`: identifier for this search
- `formItems`: list of Ext.form.FormPanel item configs, the `query` form field is required to match the rewrite rule (value is the same as `query`)
  - `xtype`: form field type
  - `name`: name of query layer attribute
  - `fieldLabel`: visible text for this field
  - `blankText`: popup text for blank fields
- `gridColumns`: list of Ext.grid.GridPanel column configs to show search results
- `selectionLayer`: name of layer for marking selected results
- `selectionZoom`: zoom level for jump-to when selecting results

For every search of this type you have to add a URL rewrite rule in the Apache config. *Note:* Linebreaks added for formatting - they should be removed in your config file.

Listing

```
RewriteCond %{QUERY_STRING} ^(?:.*)query=samplesearch&*(?:.*)$
RewriteCond %{QUERY_STRING} ^(?:(?:.*)&)?colour=([^&]*)(?:.*)$
RewriteRule ^/wms/(.+)$ /cgi-bin/qgis_mapserv.fcgi?map=/
<path-to-qgis-server-projects>/$1.qgs&SERVICE=WMS&VERSION=1.1.1&
REQUEST=GetFeatureInfo&LAYERS=Hello&QUERY_LAYERS=Hello&FEATURE_COUNT=20&
INFO_FORMAT=text/xml&SRS=EPSG:4326&FILTER=Hello:"colour"\ =\ '%1' [PT]
```

The first RewriteCond matches the `query` id of the search panel config. The second RewriteCond extracts the values of the search request parameters.

The RewriteRule composes the actual WMS GetFeatureInfo request to QGIS mapserver.

Request URL:

`http://localhost/wms/helloworld?query=samplesearch&colour=orange`

4.1.3 Add search panels to projects

In order for your search panel to appear in the web UI, you must enumerate them in your GlobalOptions.js for example (with url rewriting):

Listing

```
var mapSearchPanelConfigs = {
  "helloworld": [simpleWmsSearch, urlRewriteSearch]
};
```

Example (no rewriting):

Listing

```
var mapSearchPanelConfigs = {
  "../projects/helloworld.qgs": [simpleWmsSearch, urlRewriteSearch]
};
```

Search panels are added to a project by adding a new key for the map name with a list of search panel configs to `mapSearchPanelConfigs`. If there is no search panel configuration for a project, the search will be hidden in the GUI.

The map name is whatever is passed in the get request for your .qgs file. For example if your url includes this:

Listing

```
http://localhost/cgi-bin/qgis_mapserv.fcgi?map=../projects/helloworld.qgs
```

then your mapSearchPanelConfigs should reflect ../projects/helloworld.qgs as the key for the search list.

5 URL Rewriting

Using a standard installation of QGIS server, GlobalOptions.js will have a WMS server configuration like

Listing

```
var serverAndCGI = "/cgi-bin/qgis_mapserv.fcgi";
```

A sample URL for QGIS Web Client installed in /var/www/qgis-web-client:

Listing

```
http://localhost/qgis-web-client/qgiswebclient.html?map=/opt/geodata/maps/NaturalEarth.qgs&visibleLayers=HYP_50M_SR_W
```

With the following rules for Apache mod_rewrite you can shorten the URLs to

Listing

```
var serverAndCGI = "/wms";
```

and

Listing

```
http://localhost/maps/NaturalEarth?visibleLayers=HYP_50M_SR_W
```

Rules in VirtualHost configuration:

Listing

```
# Forbid direct access
RewriteRule ^/cgi-bin/.*$ - [F]

# Search with SearchPanel (e.g. Address)
RewriteCond %{QUERY_STRING} ^(?:.*)query=address&*(?:.*)$
RewriteCond %{QUERY_STRING} ^(?:(?:.*)&)?street=([^&]*)(?:(?:.*)&)+number=([^&]*)(?:.*)$
RewriteRule ^/wms/(.+)$ /cgi-bin/qgis_mapserv.fcgi?map=/opt/geodata/maps/$1.qgs&SERVICE=WMS&VERSION=1.1.1&REQUEST=Get

# Rewrite /wms/mapname to qgis_mapserv.fcgi?map=mappath/mapname.qgs
RewriteRule ^/wms/(.+)$ /cgi-bin/qgis_mapserv.fcgi?map=/opt/geodata/maps/$1.qgs [QSA,PT]
# Rewrite /maps/mapname to qgis-web-client main page. mapname will be extracted for wms calls in Javascript code.
RewriteRule ^/maps/([^\.]+)$ /qgis-web-client/site/qgiswebclient.html [PT]
# Rewrite /maps/* to qgis-web-client/site (e.g. /maps/gis_icons/mActionZoomNext.png -> /qgis-web-client/site/gis_icon
RewriteRule ^/maps/(.*) /qgis-web-client/site/$1 [PT]
```

For supporting qgs files in subdirectories (e.g. /maps/subdir/mapnampe) replace last rule with:

Listing

```
RewriteRule ^/maps/[^/]+/(.*) /qgis-web-client/site/$1 [PT]
```

For adding zones in different subdirecories (e.g. maps and maps-protected) add the following rules:

Listing

```
RewriteRule ^/wms-protected/(.+)$ /cgi-bin/qgis_mapserv.fcgi?map=/opt/geodata/maps-protected/$1.qgs [QSA,PT]
RewriteRule ^/maps-protected/([^\.]+)$ /qgis-web-client/site/qgiswebclient.html [PT]
RewriteRule ^/maps-protected/(.*) /qgis-web-client/site/$1 [PT]
```


6 Configuration of search python script

Searching is handled by two separate python-wsgi scripts: "search.wsgi" lists back a hit list while the user is typing in the searchbox. It groups the results and returns a bounding box of the result. "getSearchGeom.wsgi" returns the actual wkt geometry for a selected search result. It is recommended to install the wsgi scripts in a separate directory, e.g. /home/www/wsgi, a place that is not reachable by regular web traffic.

6.1 Configuration of mod_wsgi

You need to enable mod_wsgi as root. (Ubuntu: a2enmod mod_wsgi).

You need to configure apache with the following lines (e.g. in file /etc/apache2/sites-available/default):

Listing

```
#mod_wsgi
WSGIDaemonProcess gis processes=5 threads=15 display-name=%{GROUP}
WSGIScriptAlias /wsgi/ /home/www/wsgi/
WSGIScriptAliasMatch ^/wsgi/([^/]+) /home/www/wsgi/$1.wsgi
```

6.2 Adaption of the wsgi scripts to your settings and needs

6.2.1 DB connection

In the files "search.wsgi" and "getSearchGeom.wsgi" please edit the line containing the db connection strings. Search for the line

Listing

```
conn = psycopg2.connect("host='yourhost' dbname='yourdb' port='5432' user='yourusername' password='yourpassword'")
```

and adapt the parameters according to your server/db.

6.2.2 Search type to be used

The search can use PostgreSQL's tsvector data type. "A tsvector value is a sorted list of distinct lexemes, which are words that have been normalized to merge different variants of the same word." (from the doc at http://www.postgresql.org/docs/9.0/interactive/datatype-textsearch.html#DATATYPE-TSVECTOR) Thus tsvector skips all the fill words and reduces nouns to their single form, a behaviour useful for searching texts. However as

we are normally dealing with **place names** here we want them to stay as they are. If you use a language where the single form is a lot different from the plural form but your name contains a plural you will not get a suitable result. If you want to use the tsvector search option you should activate the line

Listing

```
sql += "searchstring_tsvector @@ to_tsquery(\'not_your_language\', '"+querystrings[j]+":*')"
```

not_your_language is to be replaced with an entry e.g. *finnish* if you have German place names. Thus plural forms and fillwords are kept as they are. Be aware of side effects! Be sure to fill the field *searchstring_tsvector* with `'to_tsvector('not_your_language', 'yourstring')'`.

The use of

Listing

```
sql += "searchstring::tsvector @@ lower('"+querystrings[j]+":*')::tsquery"
```

is **discouraged** as it does not find a place name like *Stoke-sub-Hamden* when you enter *Stoke*.

If you do not want to use tsvector at all you can enable the full string comparison on the field *searchstring* (activated by default).

Listing

```
sql += "searchstring ILIKE \'%"+querystrings[j]+"%\'"
```

This method however is slower than tsvector but not relevantly at least if you only have a couple 1000 datasets.

7 PostgreSQL table setup for searching

Listing

```
CREATE TABLE cadastre.searchtable
(
  searchstring text, --the search string (all lower case), e.g. "zÃ¼richstrasse 46, 8610 uster"
  displaytext text NOT NULL, --the display text for the search combobox, e.g. "ZÃ¼richstrasse 46, 8610 Uster (address
  search_category text, --should have a leading two digit number:, e.g.
                        --"03_parcels", where 03 is the order of the search categories, the number
                        --should be unique across all search tables
  the_geom geometry,    --the actual geometry
  geometry_type text,   --the geometry type as returned by ST_GeometryType(the_geom)
  searchstring_tsvector tsvector, -- be sure to fill this with to_tsvector()
  CONSTRAINT searchtable_pkey PRIMARY KEY (displaytext)
)
WITH (
  OIDS=FALSE
);
GRANT SELECT ON TABLE cadastre.searchtable TO alle;

-- Index: cadastre.in_cadastre_searchstring_tsvector_gin

CREATE INDEX in_cadastre_searchstring_tsvector_gin
  ON cadastre.searchtable
  USING gin
  (searchstring_tsvector);
```

The above search table can also be a view or materialized view. One can combine several search tables by specifying the ‘`searchtables=searchtable1,searchtable2,searchtable3`‘ parameter when requesting the *search.wsgi* script.

Using views is generally slower than properly indexed tables, check for yourself what works best.

8 License

BSD

9 Acknowledgements

We'd like to thank the OpenLayers and GeoExt team for providing their base libraries.