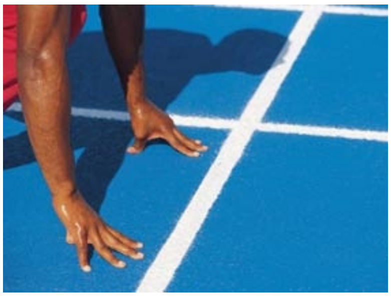
maXbox



maXbox Starter 39

Start with GEO Maps V2

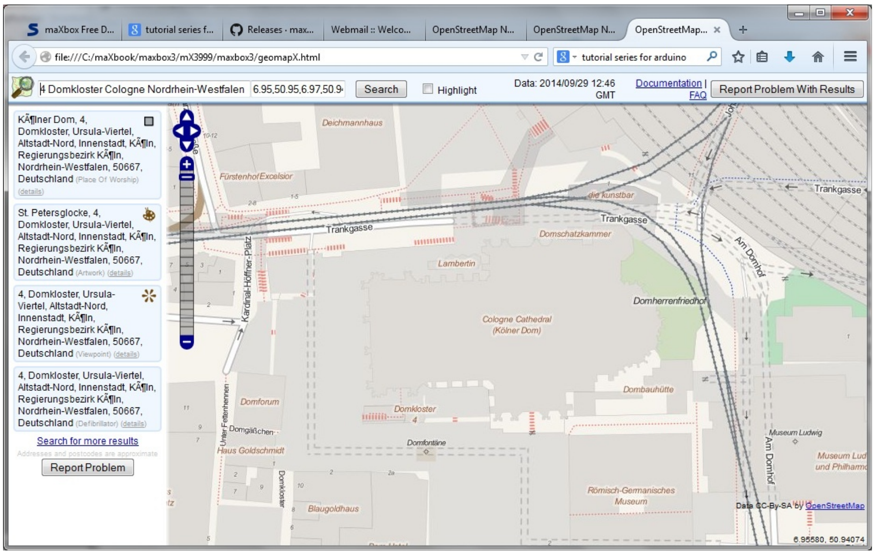
1.1 OpenStreetMap & OpenLayers

Today we run through GEO Maps coding second volume.
In the call of the Nominatim API there's a certain syntax which goes from left to right in order from a specific to a general search topic.

To find a street you have to set a detail first till a city or country last.

```
OpenMapX('4 Domkloster Cologne Nordrhein-Westfalen');
OpenMapX('1 Canal Guntzviller France');
```

With a browser you see the render result:



file:///C:/maXbook/maxbox3/mX3999/maxbox3/geomapX.html

4 Domkloster Cologne Nordrhein-Westfalen | 6.95,50.95,6.97,50.9

Data: 2014/09/29 12:46 GMT

KÃ¶lner Dom, 4, Domkloster, Ursula-Viertel, Altstadt-Nord, Innenstadt, KÃ¶ln, Regierungsbezirk KÃ¶ln, Nordrhein-Westfalen, 50667, Deutschland (Place Of Worship) (details)

St. Petersglocke, 4, Domkloster, Ursula-Viertel, Altstadt-Nord, Innenstadt, KÃ¶ln, Regierungsbezirk KÃ¶ln, Nordrhein-Westfalen, 50667, Deutschland (Artwork) (details)

4, Domkloster, Ursula-Viertel, Altstadt-Nord, Innenstadt, KÃ¶ln, Regierungsbezirk KÃ¶ln, Nordrhein-Westfalen, 50667, Deutschland (Viewpoint) (details)

4, Domkloster, Ursula-Viertel, Altstadt-Nord, Innenstadt, KÃ¶ln, Regierungsbezirk KÃ¶ln, Nordrhein-Westfalen, 50667, Deutschland (Defibrillator) (details)

Addresses and postcodes are approximate

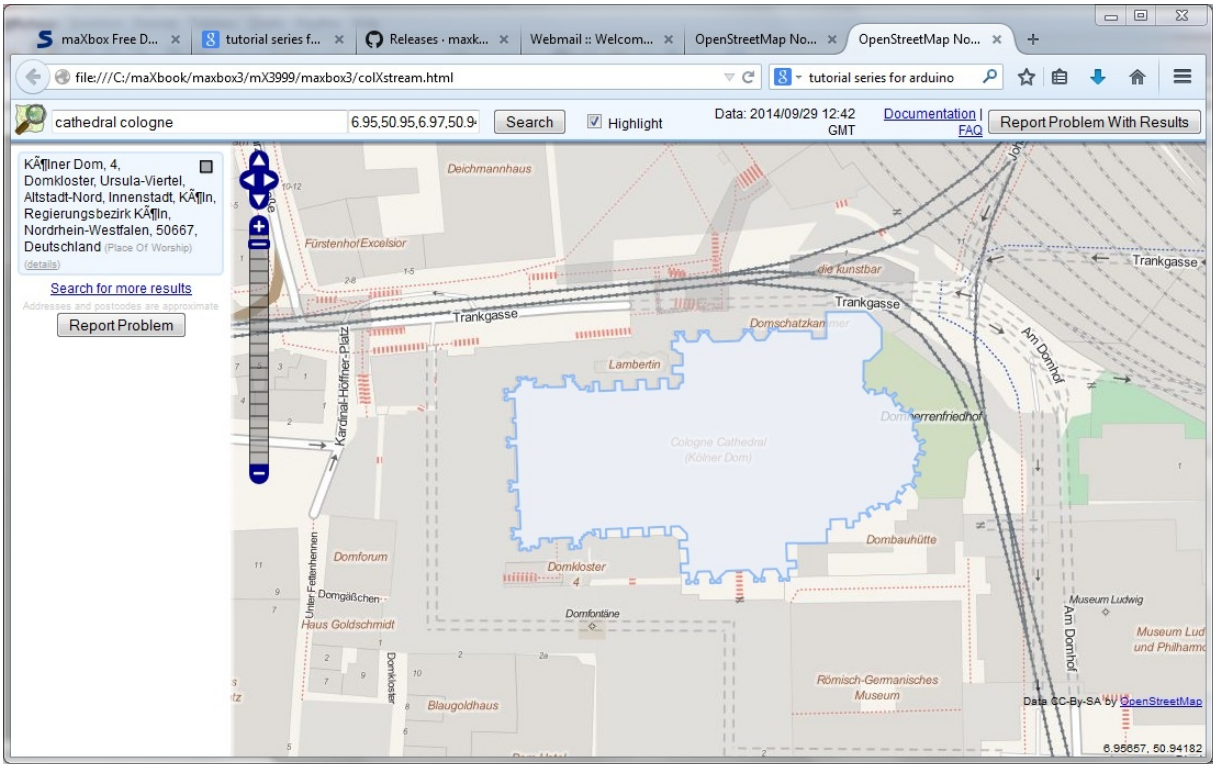
Data CC-By-SA by OpenStreetMap

6.95580, 50.94074

☝ OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source. OpenLayers is completely free, JavaScript and released under the BSD license.

Now we add a defined layer „polygon“ to the same map:

```
if GetMAPX('html&polygon=1',ExePath+'cologne2map.html',
                                   'cathedral cologne')
       then writeln('Cologne Layer Map found');
```



cathedral cologne 6.95,50.95,6.97,50.9

Data: 2014/09/29 12:42 GMT

KÃ¶lner Dom, 4, Domkloster, Ursula-Viertel, Altstadt-Nord, Innenstadt, KÃ¶ln, Regierungsbezirk KÃ¶ln, Nordrhein-Westfalen, 50667, Deutschland (Place Of Worship) (details)

You see the difference, the cathedral is shaped in blue now!
As Nominatim isn't build as a data provider, you might use to get the boundary polygons here. Output polygon outlines for items found; and there a more parameters to set with `polygon_`:

```
polygon_geojson=1
  Output geometry of results in geojson format.
polygon_kml=1
  Output geometry of results in kml format.
polygon_svg=1
  Output geometry of results in svg format.
polygon_text=1
  Output geometry of results as a WKT.
```

In maXbox you have these predefined functions to use of it:

```
Procedure GetGEOMap(C_form,apath: string; const Data: string);

Function GetMapX(C_form,apath: string; const Data: string): boolean;
Example: if GetMAPX('html',ExePath+'cologne2mapX.html',
                         'cathedral cologne') then ...


Function GetMapXGeoReverse(C_form: string; const lat,long: string):
                                                          string;
Example:
```

```
if GetMapXGeoReverse('XML',topPath,'47.0397826','7.62914761277888')
then ...

Function OpenMap(const Data: string): boolean;
Function OpenMapX(const Data: string): boolean;}

procedure GEOMapView1Click(Sender: TObject);');
Ex.:      maxform1.GEOMapView1Click(self);
```

With the click one you call an input query dialog GUI as found in the menu `View\GEO Map View`
Now we define our map search to get the map in your standard browser. We use the mapquest API for further information:

http://open.mapquestapi.com/nominatim/

The following example demonstrates a simple map search request for “Cathedral Cologne“ using the Nominatim web service. Only three parameters are being requested (menu: `View\GEO Map View`):

1. format - Output format being called. [html/json/xml]
2. json_callback - Callback function used to display results below.
3. q - Query string being searched for.

We set first a const in the script to send the request:

```
Const
   AMAPFILENAME2= 'maxmapfile2.html';
UrlMapQuestAPICode2=
'http://open.mapquestapi.com/nominatim/v1/search.php?format=
%s&json_callback=renderBasicSearchNarrative&q=%s';
```

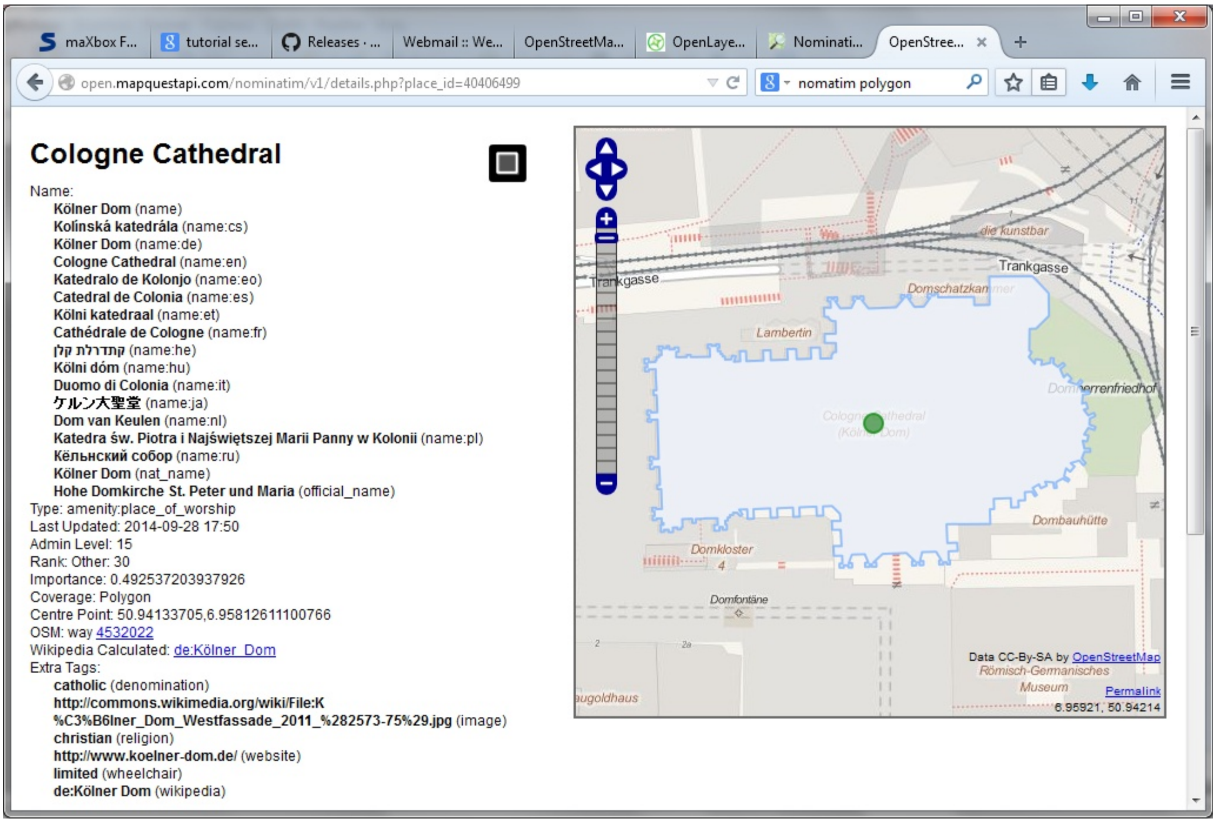
How are these 3 arguments linked together:

- `format=%s`
- `json_callback=renderBasicSearchNarrative`
- `q=%s`

You'll notice that the `json_callback` parameter is already set. The format and the query string indeed is up to you. We use a html format to prepare open a browser with the map, so here's the call again with more details:

```
 if GetMAPX('html&polygon=1&addressdetails=1&polygon_kml=1',
     ExePath+'cologne2mapX.html','cathedral cologne')
      then writeln('more Cologne Map found');
```

As a framework, OpenLayers is intended to separate map tools from map data so that all the tools can operate on all the data sources.
This separation breaks the proprietary silos that earlier GIS revolutions have taught civilization to avoid.
http://www.osgeo.org/openlayers



Cologne Cathedral

Name:
- **Kölner Dom** (name)
- **Kolínská katedrála** (name:cs)
- **Kölner Dom** (name:de)
- **Cologne Cathedral** (name:en)
- **Katedralo de Kolonjo** (name:eo)
- **Catedral de Colonia** (name:es)
- **Kölni katedraal** (name:et)
- **Cathédrale de Cologne** (name:fr)
- **קתדרלת קלן** (name:he)
- **Kölni dóm** (name:hu)
- **Duomo di Colonia** (name:it)
- **ケルン大聖堂** (name:ja)
- **Dom van Keulen** (name:nl)
- **Katedra św. Piotra i Najświętszej Marii Panny w Kolonii** (name:pl)
- **Кёльнский собор** (name:ru)
- **Kölner Dom** (nat_name)
- **Hohe Domkirche St. Peter und Maria** (official_name)

Type: amenity:place_of_worship
Last Updated: 2014-09-28 17:50
Admin Level: 15
Rank: Other: 30
Importance: 0.492537203937926
Coverage: Polygon
Centre Point: 50.94133705,6.95812611100766
OSM: way 4532022
Wikipedia Calculated: de:Kölner_Dom
Extra Tags:
- **catholic** (denomination)
- **http://commons.wikimedia.org/wiki/File:K%C3%B6lner_Dom_Westfassade_2011_%282573-75%29.jpg** (image)
- **christian** (religion)
- **http://www.koelner-dom.de/** (website)
- **limited** (wheelchair)
- **de:Kölner Dom** (wikipedia)

1.1.1 MapGap Code Behind

Obviously the most important data required for this script to work is the map from the Foundation *OpenStreetMap*. All browsers with a Java Script ECMA compatible connection are supported.

Test now the script with **F9** / F2 or press Compile. So far now we'll open the example:
 509_GEOMapMemoryStreamtest.TXT
http://www.softwareschule.ch/examples/

Now let's take a look at the second code of this app. We just change the memory stream to a more suitable string stream (or say strong stream). One of our first line is creating an object `mapStream` and the encoded URL to configure our `HTTPGet()` calling Port/ IP of the *mapquest* API with the help of `HTTPEncode`. The object makes a bind connection with the method by passing an encoded URL with a string stream:

```
procedure GetMapScriptStream(C_form,apath: string; const Data: string);
var encodURL: string;
    mapStrm: TStringStream;
begin
  encodURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
  mapStrm:= TStringStream.create('');
  try
    HttpGet(EncodURL, mapStrm);   //WinInet
    mapStrm.Position:= 0;
    SaveStringtoFile(apath, mapStrm.datastring)
    OpenDoc(apath);
  finally
    mapStrm.Free;
    encodURL:= '';
  end;
end;
```

☝ During early states of development, we used geonames to geocode collections to a point. This works well for collections which are associated with lower administrative levels such as cities, towns, and villages. However it really breaks down when you get the county, state, and country level.
What we need is a service which allows us to perform near fuzzy string searches for a place (or administrative level) and retrieve it's boundaries. In maXbox those units and many others are pre-compiled and included on demand. As long as the API is stable we don't have change calls.

```
UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search
.php?format=%s&json_callback=renderBasicSearchNarrative&q=%s';
```

After these steps, we can start with programming the rest of the GEO code called reverse geocoding with XML.

☝ Reverse geocoding is the process of back (reverse) coding of a point location (latitude, longitude) to a readable address or place name.

1.1.2 Reverse Geocoding with XML

The interesting point is now that we store no data on a file we just use a memory stream to get data direct on the console of maXbox.
Those data can be used with any console or script receiver to make further investigations. This may however vary depending on what data sentence the coordinates find or delivers, ex. the cathedral at cologne:

```
 writeln(GetMapXGeoReverse('XML','50.94134','6.95813'))
```

or more precise:

```
 GetMapXGeoReverse2('XML',topPath,'50.94133705','6.95812611100766')
```

Now the node we're interested in is `<result>` to get coordinates:

```
<?xml version="1.0" encoding="UTF-8"?>
-<reversegeocode
querystring="format=XML&json_callback=renderExampleThreeResults&lat=50.9413
3705&lon=6.95812611100766" attribution="Data © OpenStreetMap contributors,
ODbL 1.0. http://www.openstreetmap.org/copyright" timestamp="Mon, 22 Sep 14
13:36:43 +0000">
  <result lon="6.95812611100766" lat="50.94133705" ref="Kölner Dom"
```

```
   osm_id="4532022" osm_type="way" place_id="40406499">Kölner Dom, 4,
     Domkloster, Ursula-Viertel, Altstadt-Nord, Innenstadt, Köln,
     Regierungsbezirk Köln, Nordrhein-Westfalen, 50667,
     Deutschland</result>-
     <addressparts>
           <place_of_worship>Kölner Dom</place_of_worship>
           <house_number>4</house_number>
           <pedestrian>Domkloster</pedestrian>
           <neighbourhood>Ursula-Viertel</neighbourhood>
           <suburb>Altstadt-Nord</suburb>
           <city_district>Innenstadt</city_district>
           <county>Köln</county>
           <state_district>Regierungsbezirk Köln</state_district>
           <state>Nordrhein-Westfalen</state>
           <postcode>50667</postcode>
           <country>Deutschland</country>
           <country_code>de</country_code>
     </addressparts>
</reversegeocode>
```

Reverse geocoding can be carried out systematically by services which process a coordinate similarly to a geocoding process.
For ex., when a GPS coordinate is entered the street address is interpolated from a range assigned to the road segment in a reference dataset that the point is nearest to.

```
function GetMapXGeocodeReverse(C_form,apath: string; const data: string):
string;
 var encodURL, alat, alon: string;
     mapStream: TStringStream;
     xmlDoc: TXmlVerySimple; //TALXMLDocument;
     Nodes: TXmlNodeList;
     Node: TXmlNode;
 begin
   encodURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
   mapStream:= TStringStream.create('');
   xmldoc:= TXmlVerySimple.create;
   try
     HttpGet(EncodURL, mapStream);  {WinInet}
     //local test: mapstream.writeString(loadStringfromFile(apath));
     mapStream.Position:= 0;
     writeln('string stream size: '+inttostr(mapstream.size));
     {SaveStringtoFile(apath, mapStream.datastring) OpenDoc(apath);}
     xmlDoc.loadfromStream(mapstream);
     writeln('childcounts: '+inttostr(xmlDoc.root.childnodes.count))
     if xmlDoc.root.childnodes.count > 0 then begin
       Nodes:= XmlDoc.Root.FindNodes('result');
       for it:= 0 to TXMLNodeList(nodes).count-1 do begin
         //for Node in Nodes do
         Node:= TXMLNode(nodes.items[it]);
         alon:= node.attribute['lon']
         alat:= node.attribute['lat']
       end;
```

```
      result:= 'GEO Topic found: '+(node.text)+CRLF
      result:= result+('latitude: '+alat+'  longitude: '+alon)
      Nodes.Free;
    end;
  finally
    encodURL:= '';
    mapStream.Free;
    xmlDoc.Free;
  end;
end;
```

Each table has a way column containing the geometry for the object in the chosen projection. Two indices are created for each table: one for the way column and one for the osm_id column, see above.
Geometry uses coordinates in the EPSG:900913 AKA G00GlE projection and can be easily used in OpenLayers.
Now the code behind the call.
We need another *mapquest* API of the get request service:

```
 UrlMapQuestAPI3:= 'http://open.mapquestapi.com/nominatim/v1/reverse.php?
format=%s&json_callback=renderExampleThreeResults&lat=%s&lon=%s';
```

The function maps a stream to a string and is similar to above, except we don't save a file, but you can save the stream as XML or Json file:

```
 mapstream.savetofile(apath) and OpenDoc(apath);
```

```
xmlDoc.loadfromStream(mapstream);
writeln('childcounts: '+inttostr(xmlDoc.root.childnodes.count))
if xmlDoc.root.childnodes.count > 0 then begin
  Nodes:= XmlDoc.Root.FindNodes('result');
  for it:= 0 to TXMLNodeList(nodes).count-1 do begin
    //for Node in Nodes do
    Node:= TXMLNode(nodes.items[it]);
    alon:= node.attribute['lon']
    alat:= node.attribute['lat']
  end;
  result:= 'GEO Topic found: '+(node.text)+CRLF
  result:= result+('latitude: '+alat+'  longitude: '+alon)
  Nodes.Free;
end;
```

I am traversing the `TXMLNodeList` given back by `FindNodes()` in the snippet above, and looking for a span based on the attribute property.

`TMemoryStream` and `TStringStream` are both decendants of `TStream` and they work almost the same way.
Another point is to direct test a pair of coordinates. You can also convert coordinates (lat and long) to a map on the internet:
http://www.gps-coordinates.net/

You can find the address corresponding to GPS coordinates or latitude, longitude and address of any point on OpenStreetMap.
Hope you did already work with the Starter 34 on GPS topics:
http://sourceforge.net/p/maxbox/wikimax/main/

☝ **Almanac** data is data that describes the orbital courses of the satellites. Every satellite will broadcast almanac data for EVERY satellite. Your GPS receiver uses this data to determine which satellites it expects to see in the local sky; then it can determine which satellites it should track.

Feedback @
max@kleiner.com
Literature: Kleiner et al., Patterns konkret, 2003, Software & Support
http://en.wikipedia.org/wiki/Reverse_geocoding
http://openlayers.org/
http://www.kleiner.ch/kleiner/gpsmax.htm

1.2 Appendix Map Study with TMapViewer

Lille
BELGIË
BELGIQUE
Bonn
Amiens
Luxembourg
LUXEMBOURG
PARIS
Reims
Karlsruhe
Nancy
Strasbourg
Augsb
MÜN
Troyes
Mulhouse
Basel
Zürich
Dijon
Bern
Vaduz
SCHWEIZ
Loire
ours
F R A N C E

DD (decimal degrees)*
Latitude 48.725447N (48.656280963566495) - Longitude 7.143274E (6.991813648492098)
1 Rue du Canal, 57405 Guntzviller, France
Latitude : 48.725447 | **Longitude :** 7.143274 **Altitude :** 287 meters

You do also have a complete example of unit `kcMapViewer`:
https://code.google.com/r/atvliska-lazarus/source/browse/example
Source:
https://github.com/maxkleiner/maXbox3/blob/masterbox2/source/REST/kcMapViewer.pas

Script Example:
`http://www.softwareschule.ch/examples/516_mapviewer.TXT`