Geocoding Using Public Data

By Thomas Maul, 4D Germany

Technical Note 09-30

Table of Contents

Abstract

Geocoding means finding geographical information from an address or vice versa. Tech Note 09-29, Embedding Maps in 4D Using Google Maps (KB asset #75854), shows how to use the Google Maps API for this purpose. This Tech Note uses free services with Creative Commons licenses for the purpose of demonstrating how to enhance 4D applications through the use of geocoding.

Introduction

There are several services available for geocoding. Tech Note 09-29 describes using the Google Maps API. Microsoft Terraserver and Yahoo provide similar services – and there are several sources using "Open" data, usually following the creative common license, either as Creative Commons Share Alike or Creative Commons Attribution.

This Tech Note focuses on using Geonames.org and OpenStreetMap as sources.

Creative Commons License

Similar to the license systems for Open Source Software, such as GPL, there are licenses for "open data", such as Creative Commons: http://creativecommons.org/

Geonames.org uses the Creative Commons Attribution license, which allows using their data as long you mention them as the source, even for commercial usage. You can even download their data as a worldwide text file to use it in offline applications or access them as web services using the internet.

OpenStreetMap uses the Creative Commons Share Alike license, which mainly means that you need to allow your user to use the result under the same conditions. As a result, a picture with an embedded town name based on GPS coordinates received from OpenStreetmap needs to be published under CC-SA as contributed work. We suggest checking the usage with your application with a lawyer. See the following links for more information:

http://wiki.openstreetmap.org/wiki/OpenStreetMap_License

http://creativecommons.org/licenses/by-sa/2.0/

Geonames.org

From their web site (http://www.geonames.org/about.html):

"The GeoNames geographical database is available for download free of charge under a creative commons attribution license. It contains over eight million geographical names and consists of 6.5 million unique features whereof 2.2 million

populated places and 1.8 million alternate names. All features are categorized into one out of nine feature classes and further subcategorized into one out of 645 feature codes.

The data is accessible free of charge through a number of web services and a daily database export. GeoNames is already serving up to over 11 million web service requests per day.

GeoNames is integrating geographical data such as names of places in various languages, elevation, population and others from various sources. All lat/long coordinates are in WGS84 (World Geodetic System 1984). Users may manually edit, correct and add new names using a user friendly wiki interface."

Getting data from GeoNames

GeoNames allows both a full download of the data or requests via services:

http://www.geonames.org/export/

Loading the full data set and shipping it with your application is a possible approach for single user applications running on a notebook without Internet access or in a closed environment. This Tech Note focuses on online requests.

To learn of the available features you do not need to write code, simply use your favorite browser and enter:

http://ws.geonames.org/search?name=Paris

You will get an answer similar to this:

```
<geonames style="MEDIUM">
   <totalResultsCount>708</totalResultsCount>
   <geoname>
         <name>Paris</name>
         <lat>48.85341</lat>
         <lng>2.3488</lng>
         <geonameId>2988507</geonameId>
         <countryCode>FR</countryCode>
         <countryName>France</countryName>
         <fcl>P</fcl>
         <fcode>PPLC</fcode>
</geoname>
<geoname>
         <name>Paris</name>
         <lat>33.6609389</lat>
         <lng>-95.555513</lng>
         <geonameId>4717560</geonameId>
         <countryCode>US</countryCode>
         <countryName>United States</countryName>
         <fcl>P</fcl>
         <fcode>PPL</fcode>
</geoname>
```

The service finds 708 places named Paris, which could be towns, villages, mountains or other kinds of places. Using additional parameters we can filter the result – or expand it.

The parameter "style" allows you to drastically expand it. Allowed values are "short", "medium", "long" or "full. Try this next:

http://ws.geonames.org/search?name=Paris&style=full

The first entry for Paris, France, already contains too much information here to fit in this document, so take a look yourself!

If you do not specify additional parameters, the service automatically uses default values. The next test is for non English speaking readers. The returned "name" of a place is English by default. For many cities, the service knows translated names. You can request translated cities using the "lang" parameter, followed by the international country code; "de" for Germany, "fr" for French or "ja" for Japanese. Try:

http://ws.geonames.org/search?name=Paris&lang=ja

Note that the name and country fields are now returned in Japanese, all others are always in English.

The country parameter allows us to limit the search to a country:

http://ws.geonames.org/search?name=Paris&country=us&type=xml&style=full

As you see in the result (scroll to the end), it contains not just places but also streams or mountains. We can limit it to villages by using the featureClass parameter:

http://ws.geonames.org/search?name=Paris&country=us&type=xml&style=full&featureClass=P

or finding only streams named Paris in USA:

http://ws.geonames.org/search?name=Paris&country=us&type=xml&style=full&featureCode=SPNG

FeatureCode allows limiting the result to villages, banks, ATMs, gas stations, etc. Check the following link for more information:

http://www.geonames.org/export/codes.html

Using GeoNames from 4D

Integrating this service in 4D is a two step process. You need to download the information first then analyze it. For downloading, this Tech Note uses improved

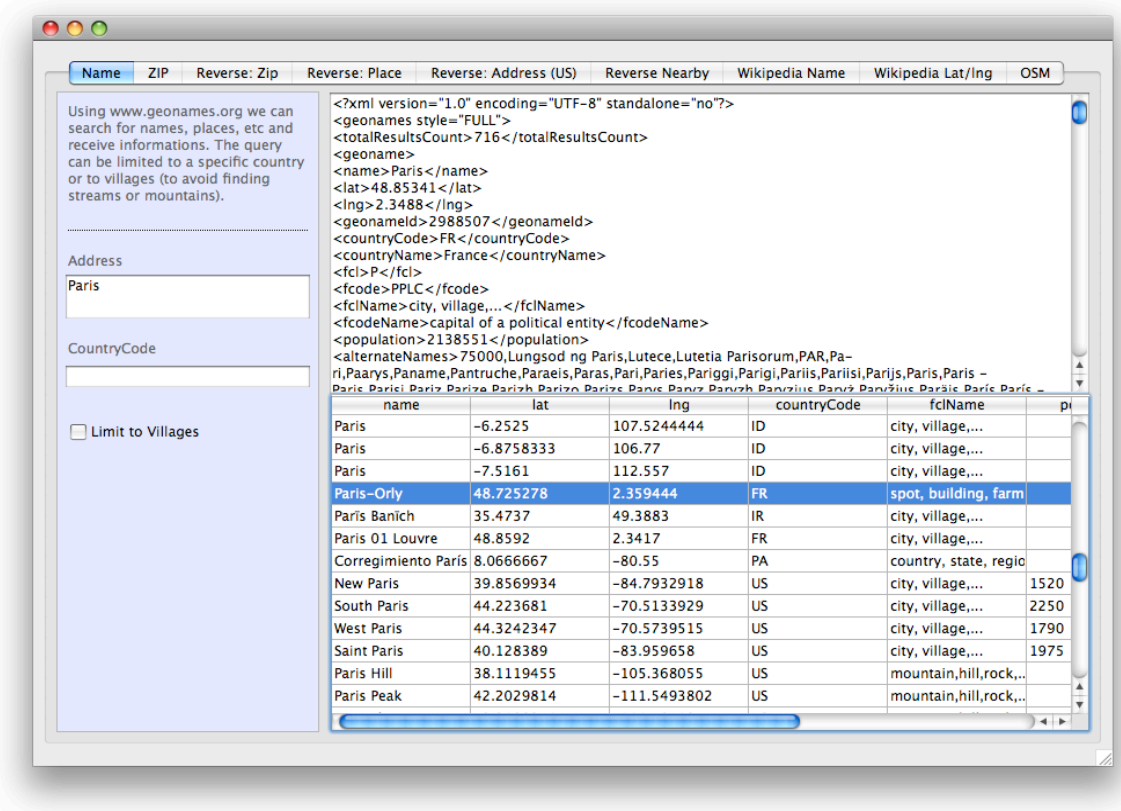
code from Tech Note 05-31, HTTP Download with a single method (KB asset #39384). This method handles all of the download process, you just need to create the URL. For analyzing first parse the XML data and extract the needed information. A set of examples are prepared to show some of the possible features. Feel free to enhance them.

Example database

The example database contains one form with several pages.

Page 1 – Name/Country

Allows you to find information about a name (such as a city) worldwide – or limited to a country. A check box allows limiting to villages/cities/towns or also to include places (as the airport "Paris Orly" in the picture below):

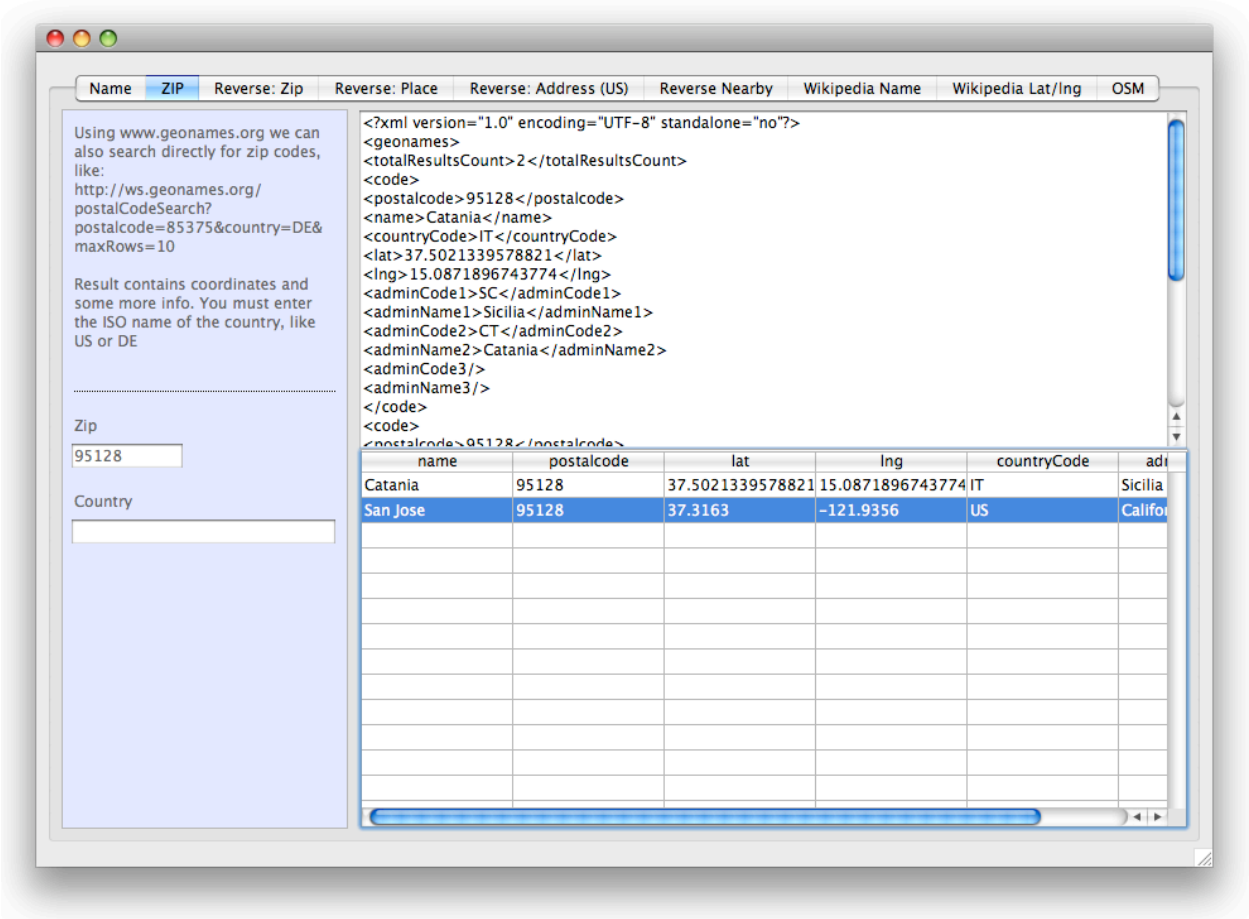


The code to achieve this is discussed a little later. First take a look at what is possible. The result is displayed as XML. Below the XML, pertinent information is separated in arrays and displayed in a list box.

You may also want to test in the first page names such as "Eiffel tower" or "Grand Canyon" (try with and without "limit to Village").

Page 2 – Zip/Country

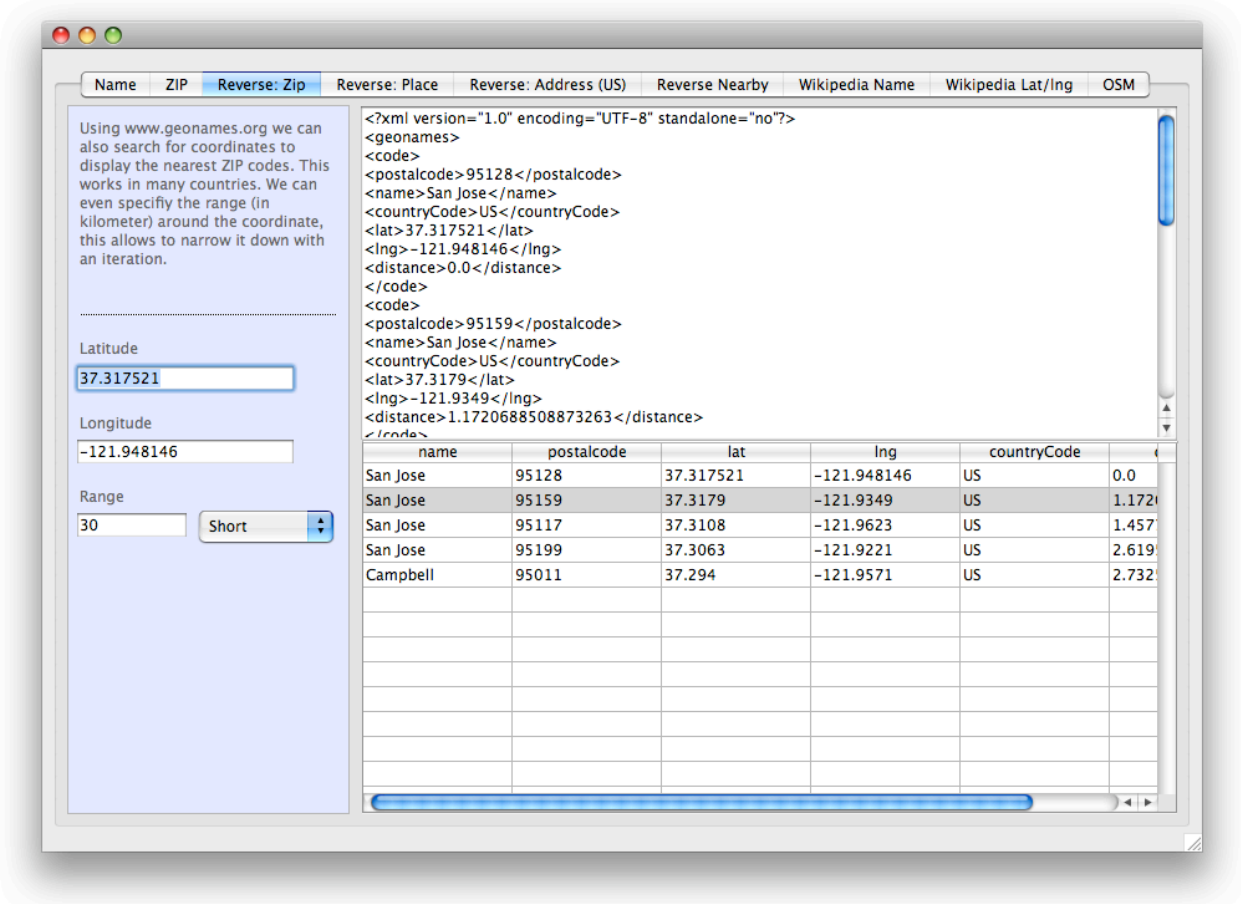
Using the zip/postal code of a town, usually in combination with a country code, directly provides the geographical coordinates of that place. This is a quite simple and fast way to build coordinates for a customer base to calculate distance to a road show, draw sales regions in a map, etc. It is possible to find all towns worldwide of a zip code in case you need that.



Page 3 – Reverse Zip

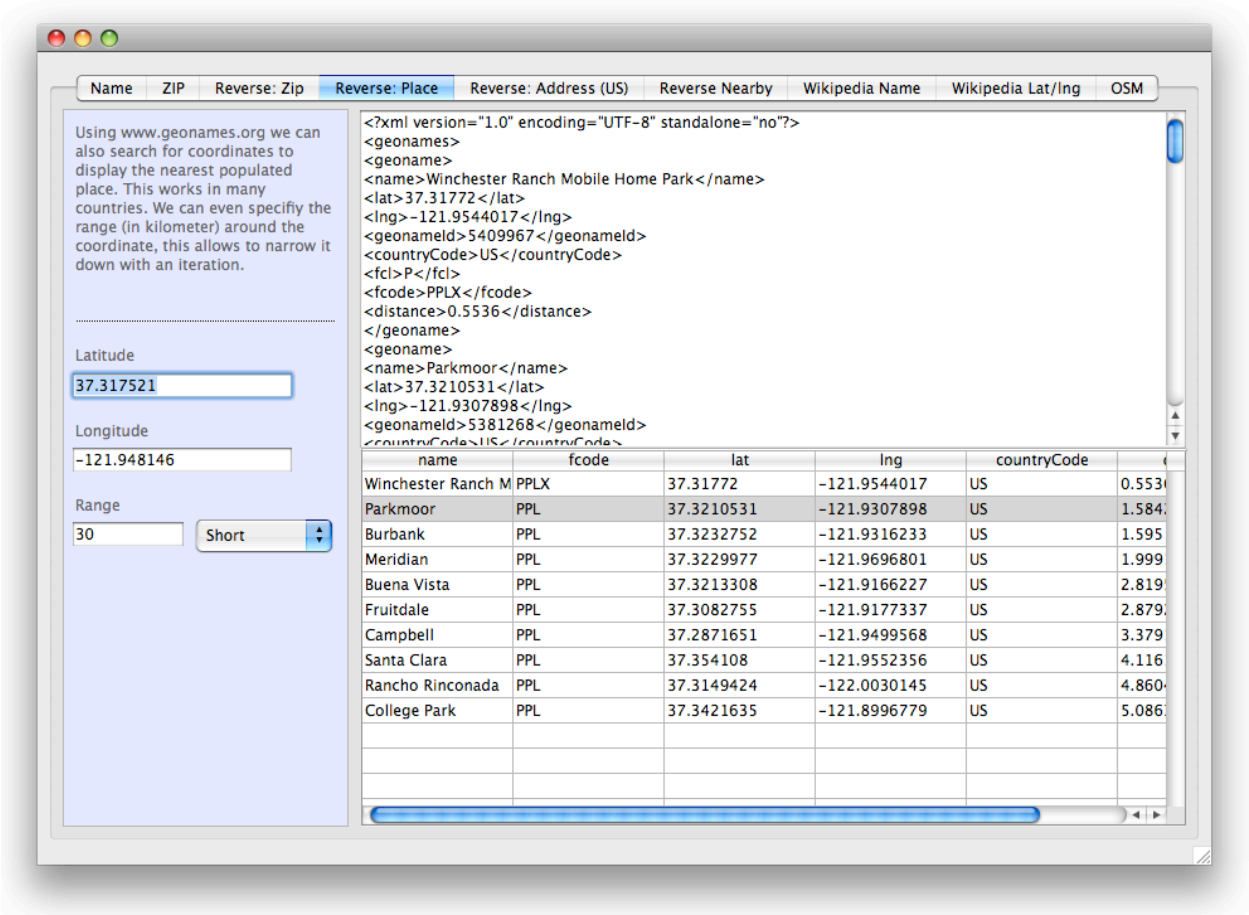
Knowing the coordinates of a place – for example from a GPS device or from a picture – you can find the nearest town and receive the zip code and town name.

The service also provides the coordinates of that place and the distance to our point. For a point in the middle of a large town using several zip codes such as Munich, the distance is quit important:



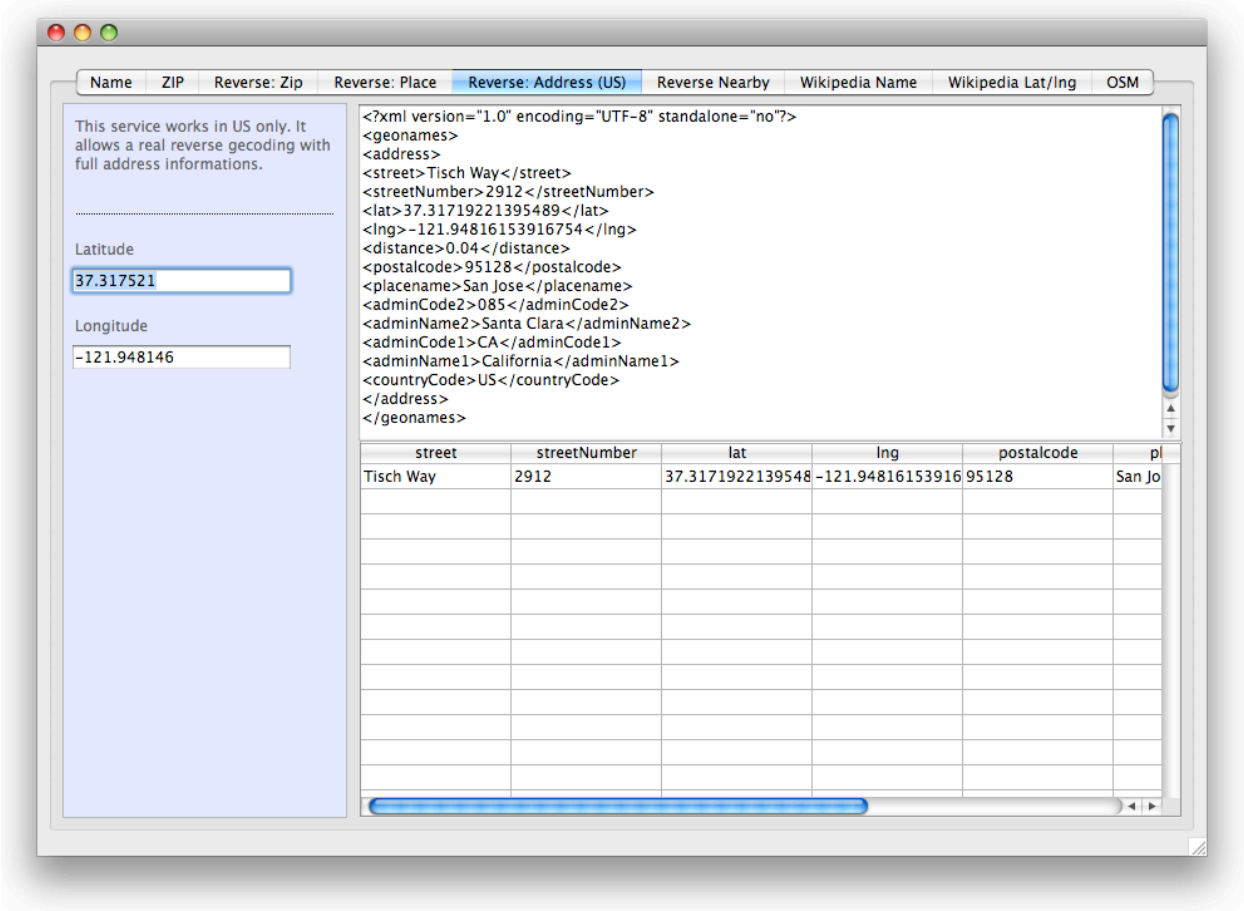
Page 4 – Reverse Place

Similar to page 3, Reverse Place returns places near to the specified coordinates. A place does not need to have unique zip codes. It may be parts of a town, for example. Below is a screenshot of places that were located near our search from Page 3:



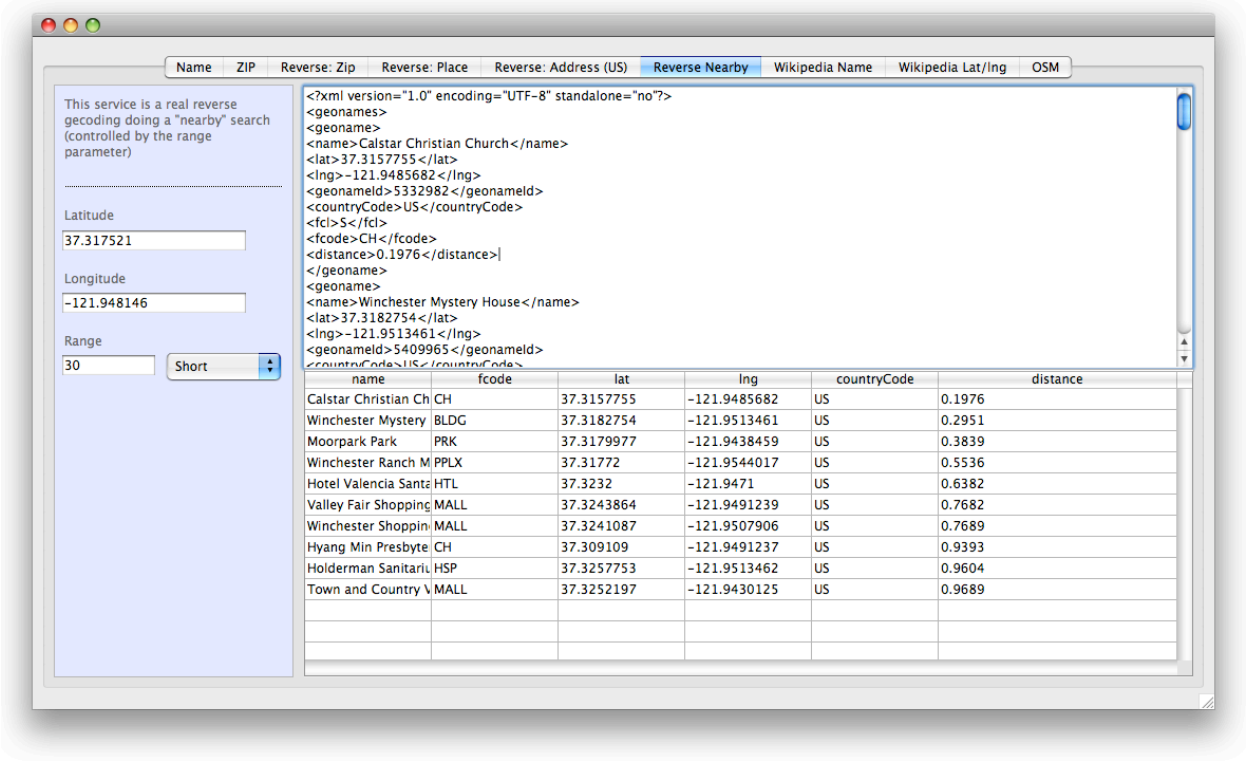
Page 5 – Reverse Address (US only)

This service is only available for coordinates inside the United States. It provides the full address information to the closed street and even reports the distance from the provided point:



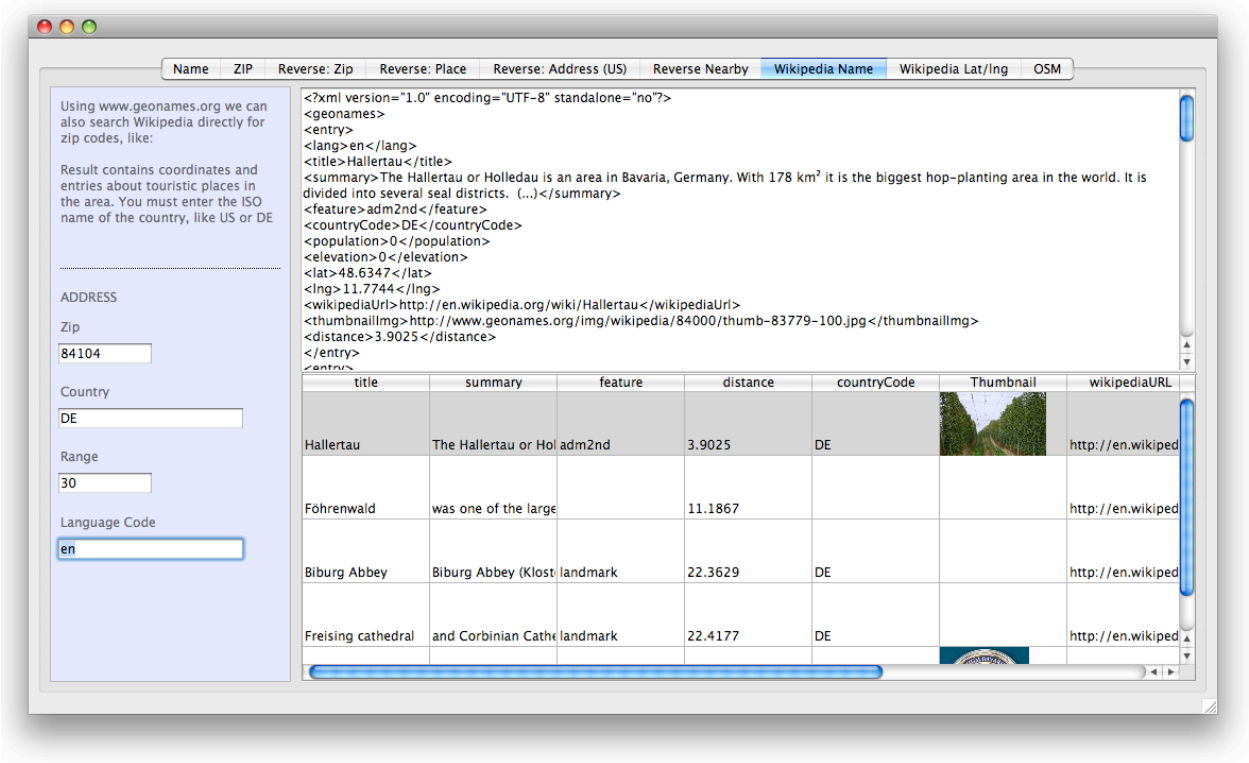
Page 6 – Reverse Nearby

This service searches for the "nearest" place to the given coordinates in the specified range. While this sometimes does not produce useful information (like hotels around a place in larger cities, for example) it is a very useful service for pictures taken at tourist locations.



Page 7 – Wikipedia search for Zip/Country

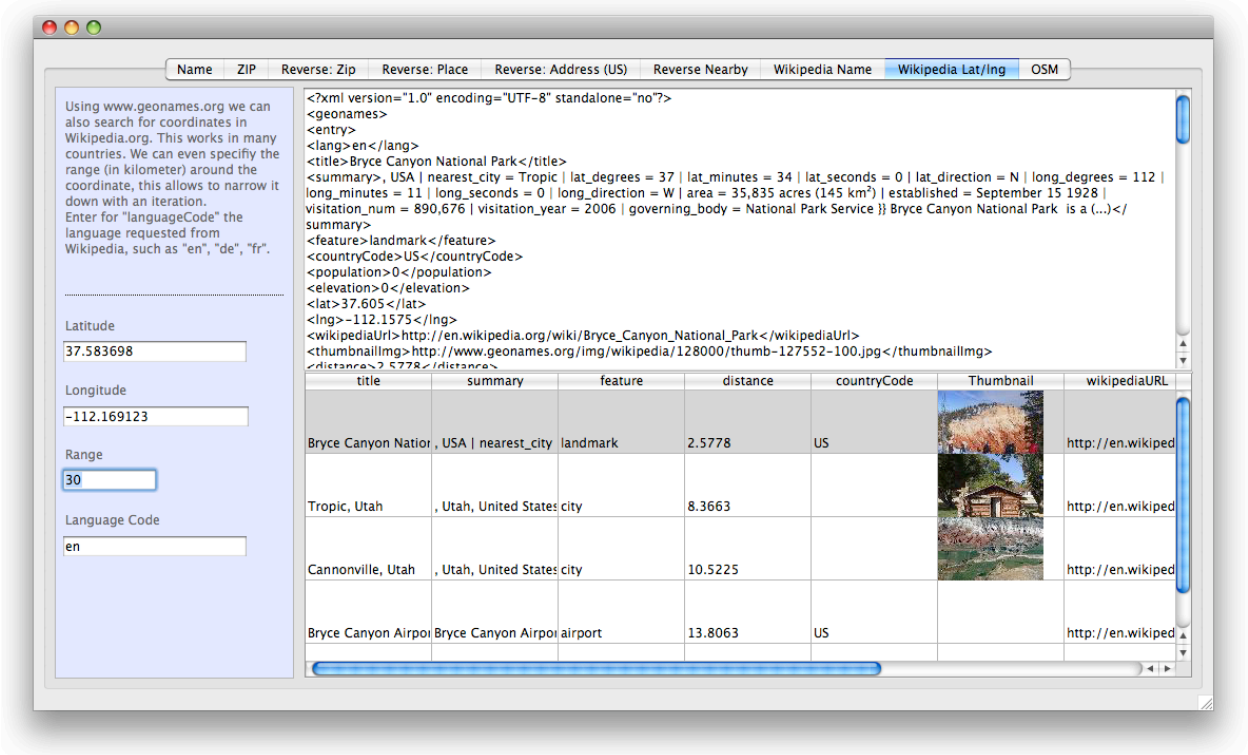
More and more Wikipedia articles all over the world are now tagged with geographical coordinates. Really cool, yes. And even better, we can search them, simply by providing ZIP/Country:



The result contains places near to the place, specified using Zip/Country, including a name, first paragraph from Wikipedia, distance to our given position, coordinates of that place, an URL to the Wikipedia article and – if available – a thumbnail about that article. Quite interesting, the query works for the specified Wikipedia language, (as "en" for English, "fr" for French, "de" for German and so on).

Page 8 – Wikipedia search for Coordinates

Similar to page 7, just based on coordinates:



Page 9 – searching in OpenStreetMap

OpenStreetMap is a community based system, similar to Wikipedia. They describe themselves on their web site as:

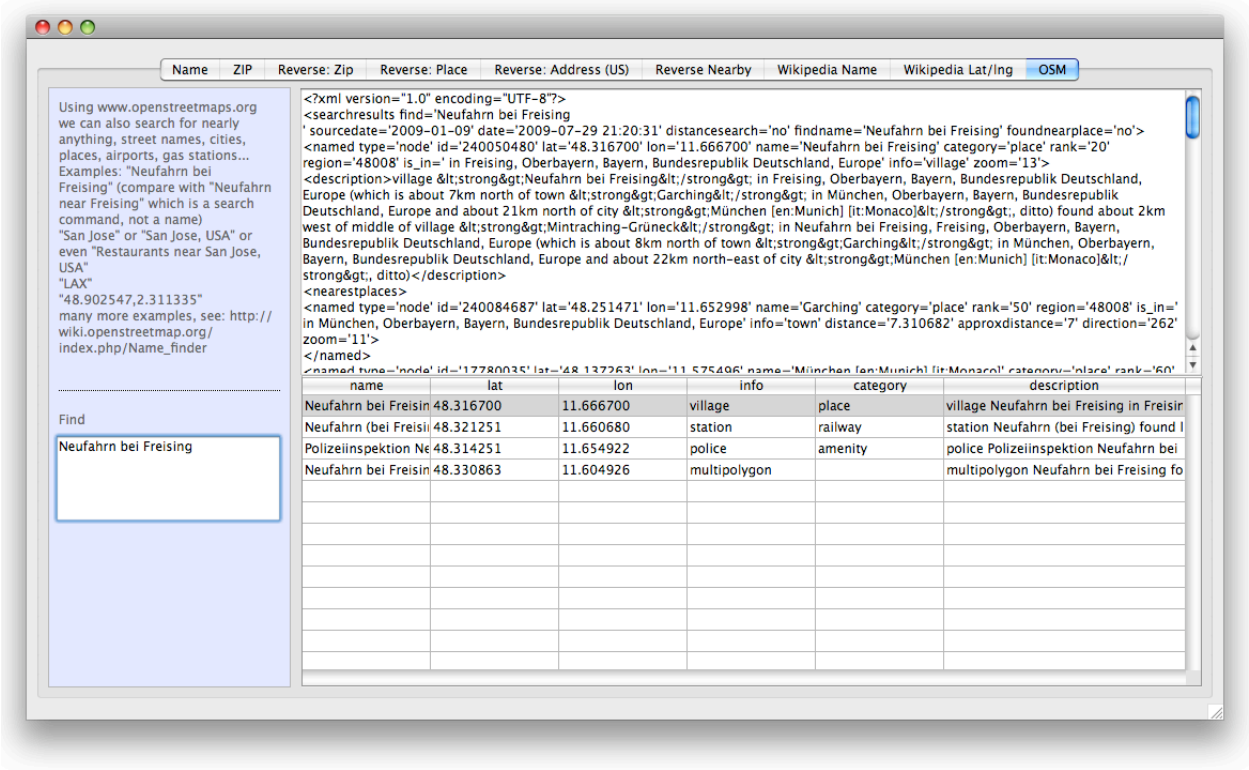
"OpenStreetMap is a free editable map of the whole world. It is made by people like you. OpenStreetMap allows you to view, edit and use geographical data in a collaborative way from anywhere on Earth. "

Looking at www.OpenStreetmap.org gives the impression that it is a map, but in fact it is a huge collection of XML nodes, each describing a point, usually connected to another point. Points (Nodes) usually have attributes, such as name (name of a village or street), often with additional attributes as zip code or other information.

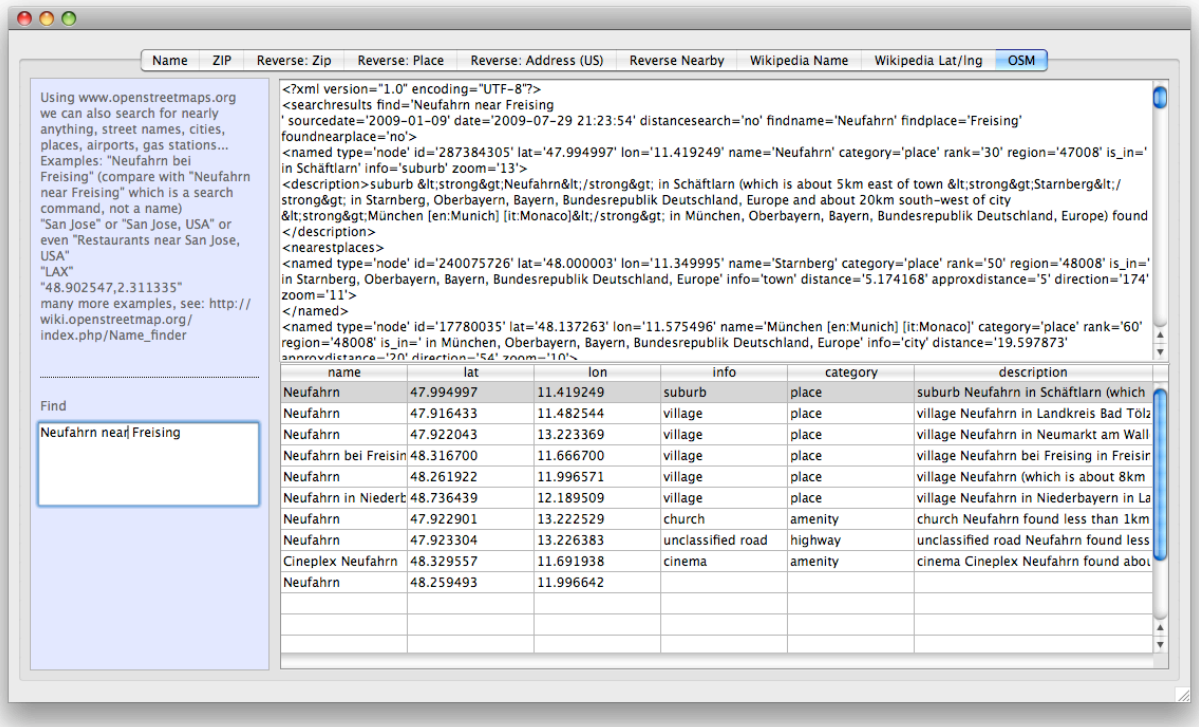
While the first idea using OpenStreetMap is as a map, similar to Google Maps, the data collection can also be used to do geocoding: Searching for a name of a city, to retrieve the coordinates, or for a book store, gas station, street name. Of course it can be also done in reverse, finding nodes near to coordinates. The information base is huge, so the response time is not always as fast as requested, but the "worldfile" can be downloaded and stored locally.

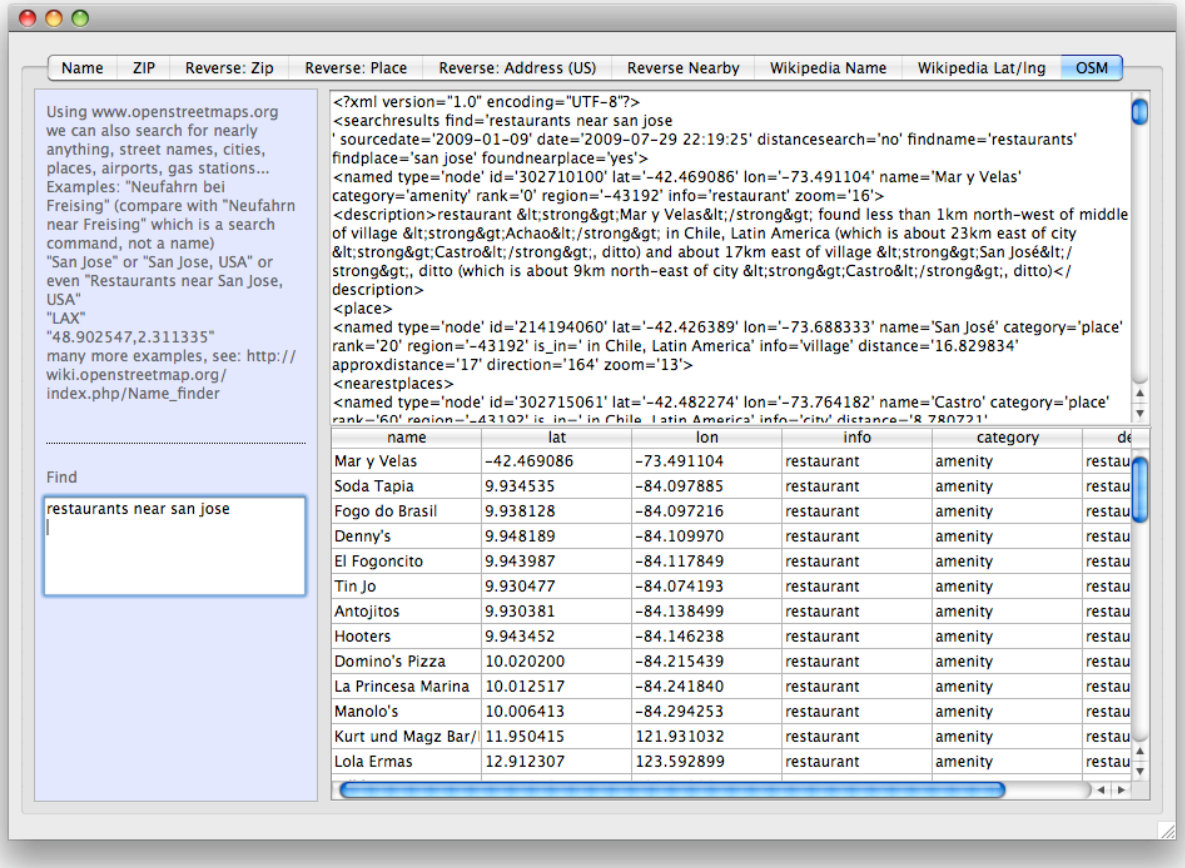
Using data from OpenStreetMap requires accepting the license, see page 3 for more information on the Creative Common Share Alike license.

The following image displays what the OpenStreetMap page displays when looking for Neufahrn bei Freising:



To illustrate the flexibility of the service, the following screenshots show what is returned when looking for points named Neufahrn near places named Freising and looking for restaurants near San Jose, respectively:


Name
ZIP
Reverse: Zip
Reverse: Place
Reverse: Address (US)
Reverse Nearby
Wikipedia Name
Wikipedia Lat/lng
OSM
Using www.openstreetmaps.org we can also search for nearly anything, street names, cities, places, airports, gas stations...
Examples: "Neufahrn bei Freising" (compare with "Neufahrn near Freising" which is a search command, not a name)
"San Jose" or "San Jose, USA" or even "Restaurants near San Jose, USA"
"LAX"
"48.902547,2.311335"
many more examples, see: http://wiki.openstreetmap.org/index.php/Name_finder
Find
Neufahrn near Freising
<?xml version="1.0" encoding="UTF-8"?>
<searchresults find='Neufahrn near Freising
' sourcedate='2009-01-09' date='2009-07-29 21:23:54' distancesearch='no' findname='Neufahrn' findplace='Freising' foundnearplace='no'>
<named type='node' id='287384305' lat='47.994997' lon='11.419249' name='Neufahrn' category='place' rank='30' region='47008' is_in='in Schäftlarn' info='suburb' zoom='13'>
<description>suburb Neufahrn in Schäftlarn (which is about 5km east of town Starnberg in Starnberg, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe and about 20km south-west of city München [en:Munich] [it:Monaco] in München, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe) found</description>
<nearestplaces>
<named type='node' id='240075726' lat='48.000003' lon='11.349995' name='Starnberg' category='place' rank='50' region='48008' is_in='in Starnberg, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe' info='town' distance='5.174168' approxdistance='5' direction='174' zoom='11'>
</named>
<named type='node' id='17780035' lat='48.137263' lon='11.575496' name='München [en:Munich] [it:Monaco]' category='place' rank='60' region='48008' is_in=' in München, Oberbayern, Bayern, Bundesrepublik Deutschland, Europe' info='city' distance='19.597873'
name | lat | lon | info | category | description
Neufahrn | 47.994997 | 11.419249 | suburb | place | suburb Neufahrn in Schäftlarn (which
Neufahrn | 47.916433 | 11.482544 | village | place | village Neufahrn in Landkreis Bad Tölz
Neufahrn | 47.922043 | 13.223369 | village | place | village Neufahrn in Neumarkt am Wall
Neufahrn bei Freisin | 48.316700 | 11.666700 | village | place | village Neufahrn bei Freising in Freisin
Neufahrn | 48.261922 | 11.996571 | village | place | village Neufahrn (which is about 8km
Neufahrn in Niederb | 48.736439 | 12.189509 | village | place | village Neufahrn in Niederbayern in La
Neufahrn | 47.922901 | 13.222529 | church | amenity | church Neufahrn found less than 1km
Neufahrn | 47.923304 | 13.226383 | unclassified road | highway | unclassified road Neufahrn found less
Cineplex Neufahrn | 48.329557 | 11.691938 | cinema | amenity | cinema Cineplex Neufahrn found abou
Neufahrn | 48.259493 | 11.996642


Name
ZIP
Reverse: Zip
Reverse: Place
Reverse: Address (US)
Reverse Nearby
Wikipedia Name
Wikipedia Lat/lng
OSM
Using www.openstreetmaps.org we can also search for nearly anything, street names, cities, places, airports, gas stations...
Examples: "Neufahrn bei Freising" (compare with "Neufahrn near Freising" which is a search command, not a name)
"San Jose" or "San Jose, USA" or even "Restaurants near San Jose, USA"
"LAX"
"48.902547,2.311335"
many more examples, see: http://wiki.openstreetmap.org/index.php/Name_finder
Find
restaurants near san jose
<?xml version="1.0" encoding="UTF-8"?>
<searchresults find='restaurants near san jose
' sourcedate='2009-01-09' date='2009-07-29 22:19:25' distancesearch='no' findname='restaurants' findplace='san jose' foundnearplace='yes'>
<named type='node' id='302710100' lat='-42.469086' lon='-73.491104' name='Mar y Velas' category='amenity' rank='0' region='-43192' info='restaurant' zoom='16'>
<description>restaurant Mar y Velas found less than 1km north-west of middle of village Achao in Chile, Latin America (which is about 23km east of city Castro, ditto) and about 17km east of village San José, ditto (which is about 9km north-east of city Castro, ditto)</description>
<place>
<named type='node' id='214194060' lat='-42.426389' lon='-73.688333' name='San José' category='place' rank='20' region='-43192' is_in=' in Chile, Latin America' info='village' distance='16.829834' approxdistance='17' direction='164' zoom='13'>
<nearestplaces>
<named type='node' id='302715061' lat='-42.482274' lon='-73.764182' name='Castro' category='place'
name | lat | lon | info | category
Mar y Velas | -42.469086 | -73.491104 | restaurant | amenity
Soda Tapia | 9.934535 | -84.097885 | restaurant | amenity
Fogo do Brasil | 9.938128 | -84.097216 | restaurant | amenity
Denny's | 9.948189 | -84.109970 | restaurant | amenity
El Fogoncito | 9.943987 | -84.117849 | restaurant | amenity
Tin Jo | 9.930477 | -84.074193 | restaurant | amenity
Antojitos | 9.930381 | -84.138499 | restaurant | amenity
Hooters | 9.943452 | -84.146238 | restaurant | amenity
Domino's Pizza | 10.020200 | -84.215439 | restaurant | amenity
La Princesa Marina | 10.012517 | -84.241840 | restaurant | amenity
Manolo's | 10.006413 | -84.294253 | restaurant | amenity
Kurt und Magz Bar/ | 11.950415 | 121.931032 | restaurant | amenity
Lola Ermas | 12.912307 | 123.592899 | restaurant | amenity

More possible search options are listed in the OSM wiki:

http://wiki.openstreetmap.org/index.php/Name_finder

How it works

All pages are from the project form "OpenSourceExamples".

All objects call the method "DemoOpenSource_FormPage", which is like a dispatcher doing all the work.

The method starts with a case of for each page, to prepare the URL to requests the data:

```
Case of
    : (arrRegister=1)   ` name search
$url:="http://ws.geonames.org/search?name"
$url:=$url+"="+Tools_UTF8Encoder (tAddress)+"&style=full"
```

If the search is for a name, the entered text needs to be encoded. Next step is to download the data:

```
$result:=HTTP_Download ("blob";->$blob;$url)
resulttext:=Convert to text($blob;"UTF-8")
```

In our example, the results are shown as simple text, **Convert to Text** extracts the text from the blob. To display it in the list box, use another case of for each page and then extract the data using some helper methods:

```
Case of
 : (arrRegister=1)   ` name search
   Tools_AppendToTextArray (->tLBArrayList;"name";"lat";"lng";
      "countryCode"; "fclName";"population";"elevation";"adminName1")
   GN_ExtractSearchResult (->$blob;->tLBArrayList;"geoname";->tLBArray1;
      ->tLBArray2;->tLBArray3;->tLBArray4;->tLBArray5;->tLBArray6;->tLBArray7;
      ->tLBArray8)
   Tools_SetListBoxArrays ("Listbox";->tLBArrayList;"tLBArray";"LBHeader")
```

The code for all pages is quite similar. They are filled by using the method Tools_AppendToTextArray and the array tLBArrayList with the names of the XML elements or attributes we are interested in.

The method GN_ExtractSearchResult uses DOM XML commands to extract the requested XML elements or attributes, as specified from array tLBArrayList and fills the result in the passed arrays.

Finally the method Tools_SetListBoxArrays dynamically modifies the listbox by adding the arrays.

If you want to receive less or more information – simply change the list of elements used in Tools_AppendToTextArray.

For the Wikipedia search the code also checks if a thumbnail URL was returned, if yes it downloads the image and pass it to a picture array.

Conclusion

Geocoding can be integrated into 4D applications to provide a wealth of geographic information about a particular point of interest. The code and examples in this document should give you an introduction how to retrieve and analyze information from geocoding services. If you need to request additional information or use another service, it should not be a big step to modify or enhance the code.