Article
Yuri Marx · May 20 8m read

Open Exchange

Geocoding with IRIS and Google Maps API

One of the crucial business dimensions is the dimension "Where". It is necessary to know where a customer was born or where he lives. Where will the order be delivered? Where have there been more sales, and where can we sell more? Where are our stores located? From Where is the customer accessing our e-commerce? These and other principal questions use the "Where" dimension. Today the best service to provide geographic accuracy and quality for these data is Google Maps. Taking advantage of the Python support of the new IRIS (2021.2+), we can use the Geocoder library (https://geocoder.readthedocs.io/) to get all the details of an address from the coordinates (lat/long) or the name of the street or building. With the geocoder, we can also know the address and data origin of a specific IP or the IP of the client who is using your application.

This article details how to do all of those things.

Using Geocoder library

In Github, download the application sample iris-geocoder (https://github.com/yurimarx/iris-geocoder). It shows how to use a geocoder to get all details about addresses, lat/long coordinates and IPs. Follow these steps:

```
IMPORTANT NOTE:
To use http://localhost:52773/iris-geocoder/forward or http://localhost:52773/iris-ge
ocoder/reverse, you must have a Google account and have an active and configured paym
ent plan. So, you can generate your Google API Key and put it inside a Dockerfile lin
e 31:
ENV GOOGLE_API_KEY=YOUR-API-KEY-HERE
See detailed instructions on: https://developers.google.com/maps/get-started
```

1. **Clone/git pull the repo into any local directory**

```
$ git clone https://github.com/yurimarx/iris-geocoder.git
```

2. **Open a Docker terminal in this directory and run:**

```
$ docker-compose build
```

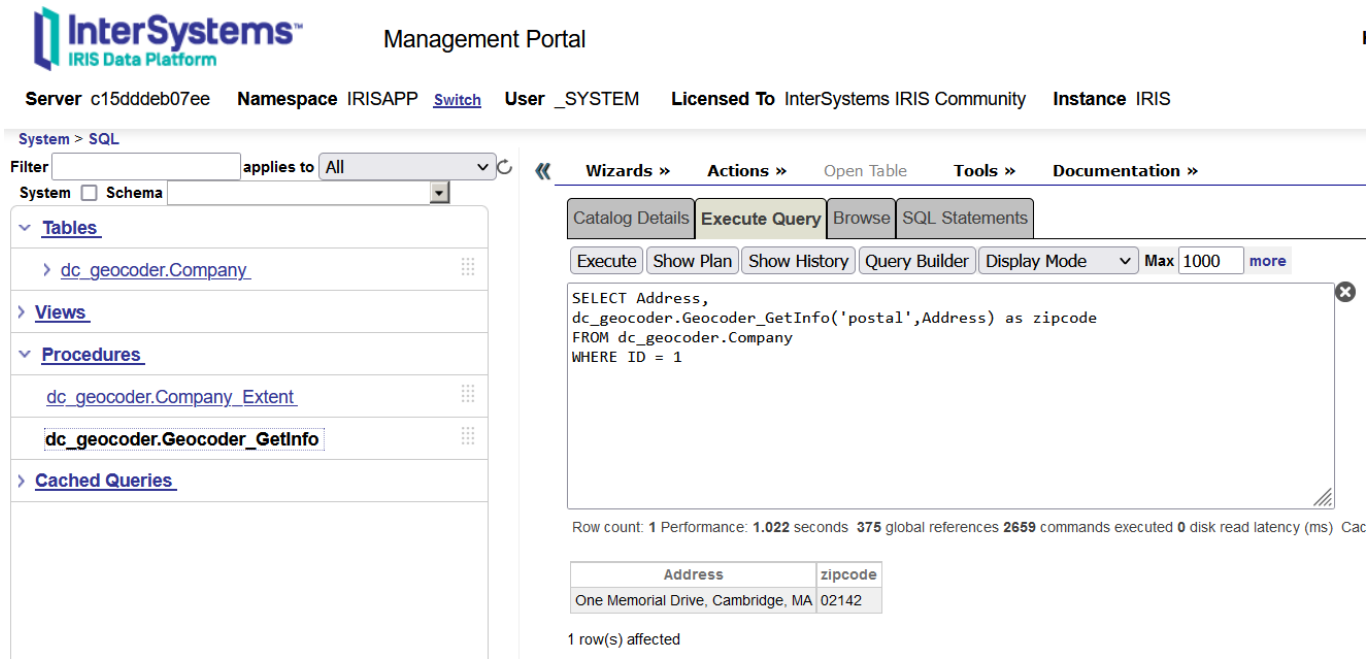
3. **Run the IRIS container:**

```
$ docker-compose up -d
```

4. **For geocoding any IRIS SQL Table**: Go to IRIS Management Portal > System > SQL and use the sample table dc_geocoder.Company to test geocoding functions. Write the following SQL sentence:

```
SELECT Address,
dc_geocoder.Geocoder_GetInfo('postal',Address) as zipcode
FROM dc_geocoder.Company
WHERE ID = 1
```

Result



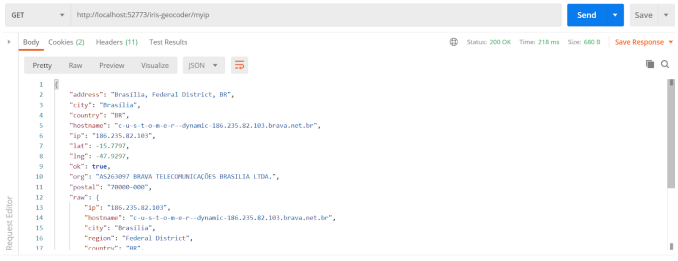
As you can see, this app has an SQL procedure called dc_geocoder.Geocoder_GetInfo. This procedure receives 2 parameters. The first parameter is the geocoding data you want to get (postal for zipcode, lat for latitude, and other values in the next table). The second parameter is a textual Address (street + city + state code, like One Memorial Drive, Cambridge, MA). So the procedure will return zip codes, latitudes, longitudes or these options:

Geocoding information
address
city
country
housenumber

county
quality
lat
lng
state
street

Some information may return blank depending on the accuracy of the address passed to geocoding. It is desirable to fill in at least street + city + state.

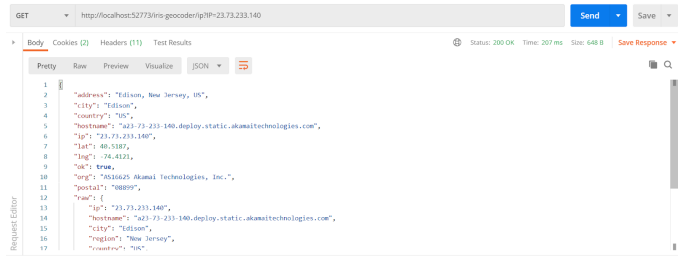
5. **For Geocoding your current IP:** Go to your Postman (or another similar REST client) and config the request like this, and click send to see the results:



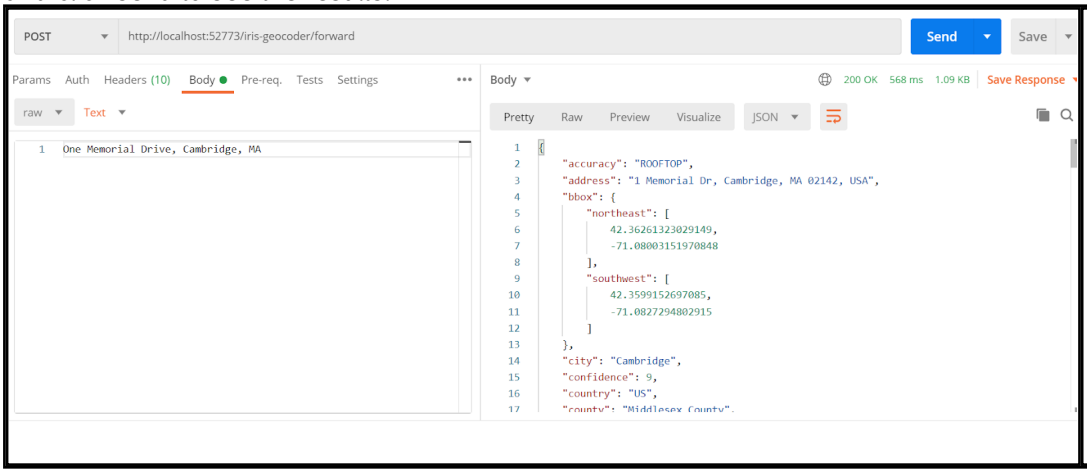
- Method: GET
- URL: http://localhost:52773/iris-geocoder/myip

6. **For Geocoding a particular IP:** Go to your Postman (or another similar REST client) and config the request like this, and click send to see the results:

- Method: GET
- URL: http://localhost:52773/iris-geocoder/ip?IP=23.73.233.140 (this is the IP for InterSystems Site. You can choose another one if you want)



7. **For Geocoding an address:** Go to your Postman (or another similar REST client) and config the request like this, and click send to see the results:



- Method: POST
- URL: http://localhost:5277
- Body: raw
- Body text: One Memorial

8. **For Geocoding a Lat/Long point:** Go to your Postman (or another similar REST client) and config the request like this, and click send to see the results:

Method: GET
URL:
http://localhost:52773/iris-geocoder/reverse?Lat=42.36261323029149&Long=...

Behind the scenes - the source code for SQL code

To use geocoding on the SQL sentences, a Geocoder class was created. See it below:

```
 Class dc.geocoder.Geocoder Extends %RegisteredObject
{
  ClassMethod GetInfo(field As %String, address As
%String) As %String [ SqlProc ]
{

    Set Result = {}

Set LatLong = ##class(dc.geocoder.
```

```
GeocoderService).DoForwardGeocoding(address)
    Set Result = {}.%FromJSON(LatLong)
    Quit Result.%Get(field)
}
  }
```

The ClassMethod GetInfo was created with the SqlProc modifier, allowing this method to be used inside SQL sentences as a stored procedure. So the class uses GeocoderService (the detailed description is in the next section). The result contains several geocoding data fields, so the field parameter returns the field which is programmed to be returned by the stored procedure.
In the application, five real addresses were created for testing purposes. These data are stored in the Company table. See it below:

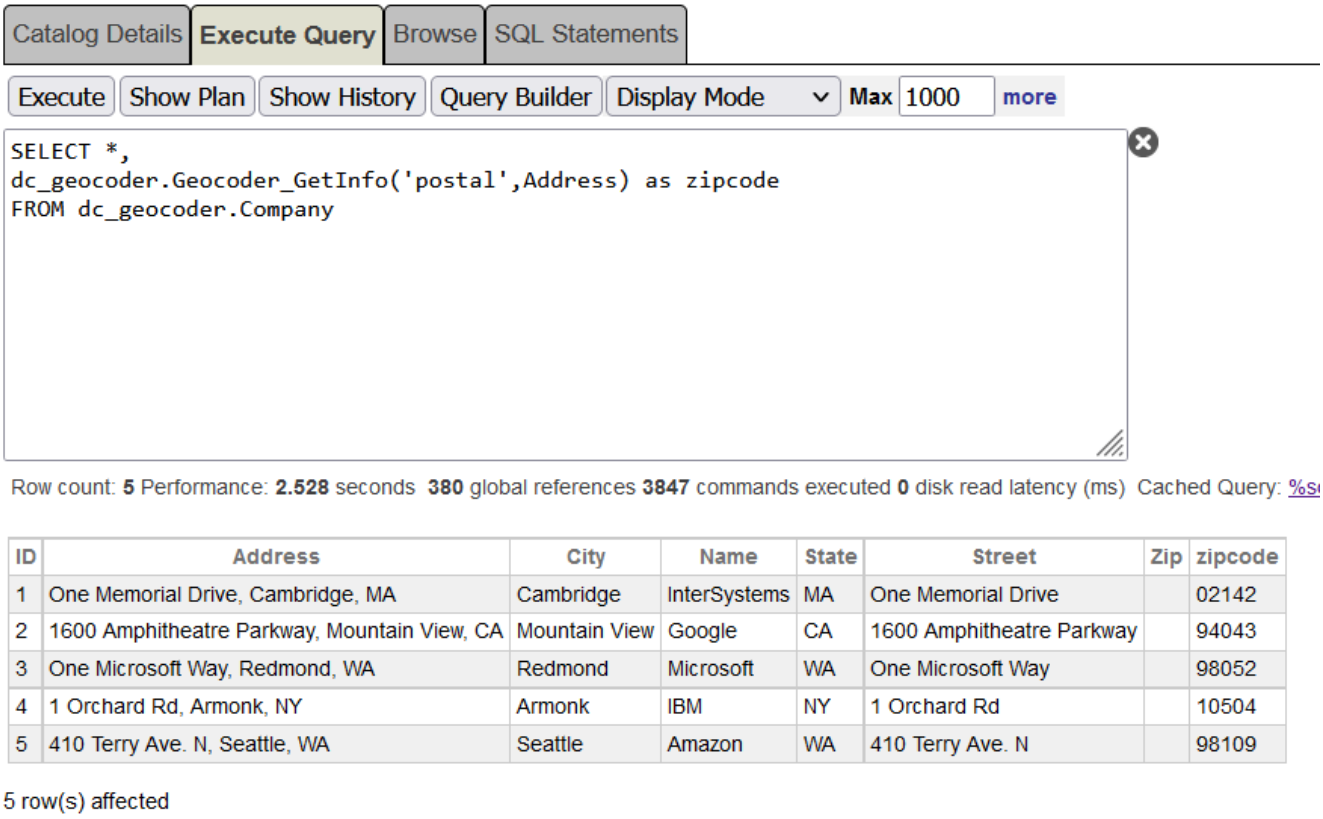
```
 Class dc.geocoder.Company Extends %Persistent
{
  Property Name As %String;
  Property Street As %String(MAXLEN = 200);
  Property City As %String(MAXLEN = 80);
  Property State As %String(MAXLEN = 2);
  Property Zip As %String(MAXLEN = 5);
  Property Address As %String [ Calculated,
SqlComputeCode = {set {*}={Street}_", "_{City}_",
"_{State}}, SqlComputed ];
  ClassMethod CreateFiveCompanies() As %Status
{
    Do ..CreateCompany("InterSystems", "One Memorial
Drive", "Cambridge", "MA", "")
    Do ..CreateCompany("Google", "1600 Amphitheatre
Parkway", "Mountain View", "CA", "")
    Do ..CreateCompany("Microsoft", "One Microsoft
Way", "Redmond", "WA", "")
    Do ..CreateCompany("IBM", "1 Orchard
Rd", "Armonk", "NY", "")
    Do ..CreateCompany("Amazon", "410 Terry Ave.
N", "Seattle", "WA", "")
}
  ClassMethod CreateCompany(Name As %String, Street As
%String, City As %String, State As %String, Zip As
%String) As %Status
{

    Set company = ##class(dc.geocoder.Company).%New()
```

```
    Set company.Name = Name
    Set company.Street = Street
    Set company.City = City
    Set company.State = State
    Set company.Zip = Zip
      Set sc = company.%Save()
      Return sc
}
  Storage Default
{
<Data name="PersonDefaultData">
<Value name="1">
<Value>%%CLASSNAME</Value>
</Value>
<Value name="2">
<Value>Name</Value>
</Value>
<Value name="3">
<Value>Street</Value>
</Value>
<Value name="4">
<Value>City</Value>
</Value>
<Value name="5">
<Value>State</Value>
</Value>
<Value name="6">
<Value>Zip</Value>
</Value>
<Value name="7">
<Value>Address</Value>
</Value>
</Data>
<DataLocation>^dc.geocoder.PersonD</DataLocation>
<DefaultData>PersonDefaultData</DefaultData>
<IdLocation>^dc.geocoder.PersonD</IdLocation>
<IndexLocation>^dc.geocoder.PersonI</IndexLocation>
<StreamLocation>^dc.geocoder.PersonS</StreamLocation>
<Type>%Storage.Persistent</Type>
```

```
}
   }
```

This class loads to you the addresses of five tech companies (Google, InterSystems, IBM, Microsoft and Amazon), with a company Name, Street, City, State and Zip Code (empty). This class has a calculated field to get a qualified address (street + city + state) to be used in the geocoding testing. Check the table content:



ID	Address	City	Name	State	Street	Zip	zipcode
1	One Memorial Drive, Cambridge, MA	Cambridge	InterSystems	MA	One Memorial Drive		02142
2	1600 Amphitheatre Parkway, Mountain View, CA	Mountain View	Google	CA	1600 Amphitheatre Parkway		94043
3	One Microsoft Way, Redmond, WA	Redmond	Microsoft	WA	One Microsoft Way		98052
4	1 Orchard Rd, Armonk, NY	Armonk	IBM	NY	1 Orchard Rd		10504
5	410 Terry Ave. N, Seattle, WA	Seattle	Amazon	WA	410 Terry Ave. N		98109

5 row(s) affected

In this sample, zip code was not stored, but our geocoding procedure returned the zip code to you.

Behind the scenes – the source code for REST API

The magic occurs inside src/dc/geocoder/GeocoderService.cls. Take a look at it:

1. Get geocoding data from a simple address:

```
 // Forward Geocoding
  ClassMethod DoForwardGeocoding(address) [ Language =
python ]
{
        import geocoder
        import os
        import json
```

```
g = geocoder.google(address, key=os.environ[
"GOOGLE_API_KEY"])
        return json.dumps(g.json)
}
```

It is very simple. You need to import a geocoder and go to a geocoder.google() call to get a JSON with all the geocoding information you need. As you can see, you need to have a GOOGLE_API_KEY. Find more details on https://developers.google.com/maps/get-started.

2. Get geocoding data from lat/long coordinates:

```
 // Reverse Geocoding
  ClassMethod DoReverseGeocoding(lat, long) [
Language = python ]
{
        import geocoder
        import os
        import json

g = geocoder.google([lat, long], key=os.environ[
"GOOGLE_API_KEY"], method='reverse')
        return json.dumps(g.json)
}
```

In this scenario we use the same geocoder.google() but setting method='reverse', because we are using lat/long to get geocoding and not a textual address.

3. Get geocoding data from an IP address:

```
 // IP Addresses
  ClassMethod DoIPGeocoder(ip) [ Language = python ]
{
        import geocoder
        import json
        g = geocoder.ip(ip)
        #g = geocoder.ip('me')
        return json.dumps(g.json)
}
```

To get geocoding data about an IP, you do not need Google Maps API. You should call geocoder.ip() for the passing IP to be processed, and you will get all the data you need.

4. Get geocoding data from the user IP:

```
 // My IP Addresses
  ClassMethod DoMyIPGeocoder() [ Language = python ]
{
```

```
        import geocoder
        import json
        g = geocoder.ip('me')
        return json.dumps(g.json)
}
```

If you want to get geocoding data about your client's IP, it is very simple to do. Just call geocoder.ip('me').

More About it

Geocoder allows you to use other geocoding providers (Bing, ArcGIS, Baidu, Mapbox and OpenStreetMap). Discover more details on https://geocoder.readthedocs.io/, section Providers.

#Embedded Python #InterSystems IRIS
Check the related application on InterSystems Open Exchange

Source URL: https://community.intersystems.com/post/geocoding-iris-and-google-maps-api