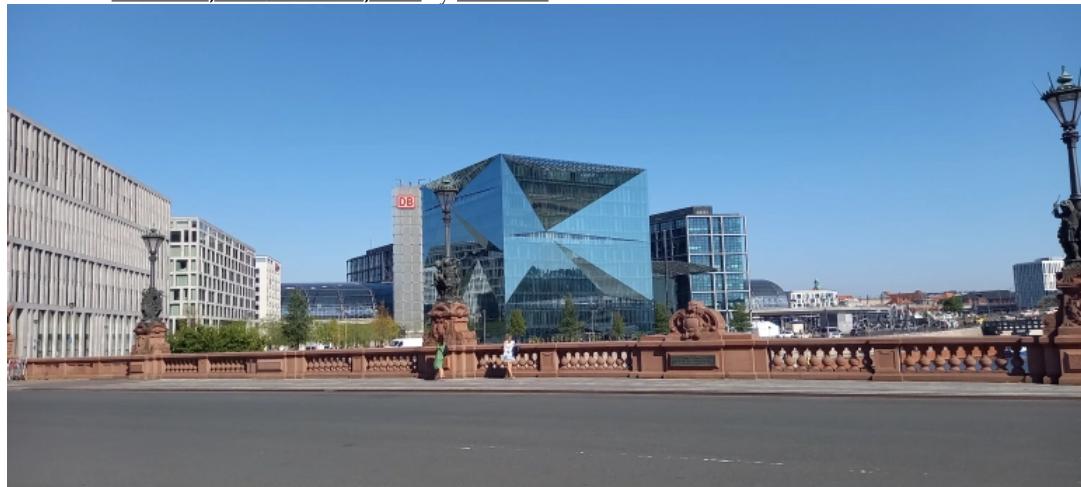


maXbox

Station2Station

Posted on October 2, 2024October 6, 2024 by maxbox4



Use this tool as a script (1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas) to get the directions between any point using google maps. Enter a city or an address in both the **From** and the **To** address inputs. You can also set direct an lat- and longitude. Click Got to Google... (Find Directions), and the tool will display the route you need to take to get from your starting location to your end location in the browser. The turn by turn directions will be displayed below the map, and will contain the distance and approximate time it will take to get from one location to the other.

Form64 Googel Directions WebForm61

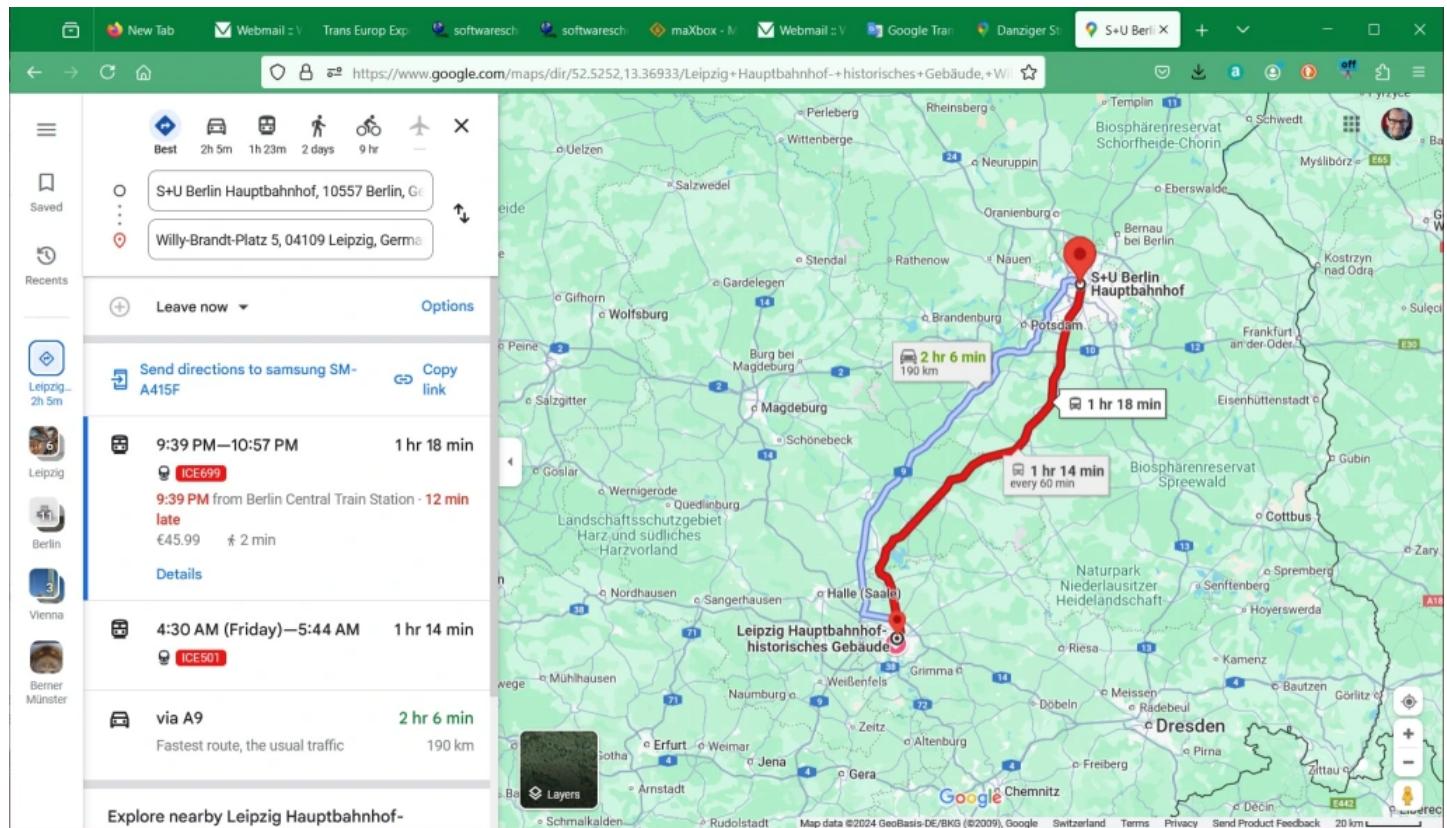
Google Directions NavUtils61

From	Hauptbahnhof, Berlin, Germany
Source Longitude	13.3693321
Source Latitude	52.5251951
13° 22' 9.596"E 52° 31' 30.702"N	
To	Hauptbahnhof, Leipzig, Germany
Dest Longitude	12.3808565
Dest Latitude	51.344033400000006
12° 22' 51.083"E 51° 20' 38.520"N	
<input checked="" type="checkbox"/> Check To See Link In Browser	Go to Google..
https://www.google.com/maps/dir/052.52520,013.36933/051.34403,012.38086/@052.52520,013.36933,9z	
Distance as Crow Flies =147.8km	
GeoCoords: Dest: lat: 51.3393 - lon: 12.3726	

The GoogleDirForm from to box with geocoding and from open street maps.

The licence is a "licence": "Data © OpenStreetMap contributors, ODbL 1.0. <http://osm.org/copyright>", and you can also get your own API-

key. The a browser starts with the google maps directions window, as we checked the "Check To See Link In Browser":



Berlin – Leipzig as Station to Station

Get or find coordinates: Search for a place using its latitude and longitude coordinates, or get the coordinates of a place you've already found.

Code behind

TDirection is working as a URL Builder: IsNavUtils – Provides a **RNavigateLongLat** record as class to manipulate Longitude and Latitude references and the functions and constants required to achieve this.

```

1  procedure TDirectionsBtnGoGoogleClick(Sender: TObject);
2  Var
3    LocFrom,LocTo:RNavigateLongLat;
4    Long,Lat:Double;
5  begin
6    LocFrom:=RNavigateLongLat.create;
7    LocTo:=RNavigateLongLat.create;
8    Long:=RealFrmDegreeText(EdtLong.Text);
9    Lat:= RealFrmDegreeText(EdtLat.Text);
10   LocFrom.CreateDec(Long,Lat);
11   Long:=RealFrmDegreeText(EdtLong2.Text);
12   Lat:= RealFrmDegreeText(EdtLat2.Text);
13   LocTo.CreateDec(Long,Lat);
14   // { 0 Start 1 End 3 Center }
15   EdtGoogleLink.Text:=LocFrom.GoogleLinkDirectionsTo(LocTo, 0);
16   LBlCrowFlies.Caption:='Distance as Crow Flies ='+FormatFloat('0.0km',LocTo.MetresFrom(LocFrom)/1000);
17   if CBxGoNow.Checked then
18     LocFrom.GoGoogleDirectionsTo(LocTo, 0);
19 end;
```

For the geocoding we use a restclient **THttpRequestC** a JSON Lib and the OSM API from nominatim. Nominatim uses **OpenStreetMap** data to find locations on Earth by name and address (geocoding). It can also do the reverse, find an address for any location on the planet. For occasional use. Use the API. Usage policy. API documentation. For power users. Install your own. The latest release is 4.5.0.

```
1  function TAddressGeoCodeOSM8(AURL, location, aApiKey: string): tlatlong;
2  var Httpreq: THttpRequestC; httpres: string;
3  Body: TMultipartFormBody;
4  jsn: TMJsonItem;
5 begin
6  httpreq:= THttpRequestC.create(self);
7  httpreq.headers.add('Accept: application/json; charset=utf-8');
8  httpreq.userAgent:= USERAGENT4;
9  httpreq.SecurityOptions:= [soSsl3, soPct, soIgnoreCertCNInvalid];
10 try
11  if httpreq.get(Format(AURL,[location])) then begin
12    //httpres:= (httpreq.Response.ContentAsString)
13    httpres:= (httpreq.Response.ContentAsUTF8String)
14    //writeln('conttype '+httpreq.Response.ContentType);
15    writeln('debug back '+formatJson(httpres));
16    jsn:= TMJsonItem.Create;
17    jsn.AsJSON:= httpres;
18    result.lat:= jsn.at(0,'lat').asnumber;
19    result.long:= jsn.at(0,'lon').asnumber;
20    result.descript:= Format('Coords: lat %2.5f lng %2.5f %s place_id: %d',
21                           [result.lat,result.long,jsn.at(0,'display_name').asString,
22                            jsn.at(0,'place_id').asinteger]);
23  end else Writeln('APIError '+inttostr(Httpreq.Response.StatusCode2));
24  //StrReplace(httpres, '[{', '{');
25 finally
26  writeln('Status3: '+gethttpcod(httpreq.Response.statuscode2))
27  httpreq.Free;
28  sleep(200)
29  jsn.Free;
30 end;
31 end;
```

Result: OSM _from: Coords: lat 46.94724 lng 7.45158 Münster, 1, Münsterplatz, Altstadt, Grünes Quartier, Stadtteil I, Bern, Verwaltungskreis Bern-Mittelland, Verwaltungsregion Bern-Mittelland, Bern/Berne, 3000, Schweiz/Suisse/Svizzera/Svizra place_id: 78089330
OSM _to: Coords: lat 51.33933 lng 12.37260 Thomaskirche, 18, Thomaskirchhof, Zentrum, Mitte, Leipzig, Sachsen, 04109, Deutschland place_id: 118838745
get geocoords: lat: 51.3393 - lon: 12.3726
mX5



executed: 26/09/2024 09:52:14 Runtime: 0:0:4.38 Memload: 62% use

The GoogleDirForm is simple and a really simple form to put in Long/Lat references and also Locations and then get Google Directions in Browser:

The image contains two side-by-side screenshots of a browser window displaying Google Directions. Both screenshots show a map with a blue route line and several route segments with estimated times and distances.

Left Screenshot (Germany Map):
- Route: Mainz Cathedral (Market, 00511 Mainz) to Saint Thomas Church, Thomaskirchhof 18, Leipzig.
- Segments:

- 11:23 AM (Sunday) — 4:24 PM: 5 hr 1 min (via Autobahn)
- 11:26 AM (Sunday) — 4:35 PM: 5 hr 9 min (via Autobahn)
- 11:31 AM (Sunday) — 4:35 PM: 5 hr 4 min (via Autobahn)
- 12:23 PM (Sunday) — 4:38 PM: 5 hr 7 min (via Autobahn)
- 1:23 PM (Sunday) — 4:35 PM: 5 hr 12 min (via Autobahn)

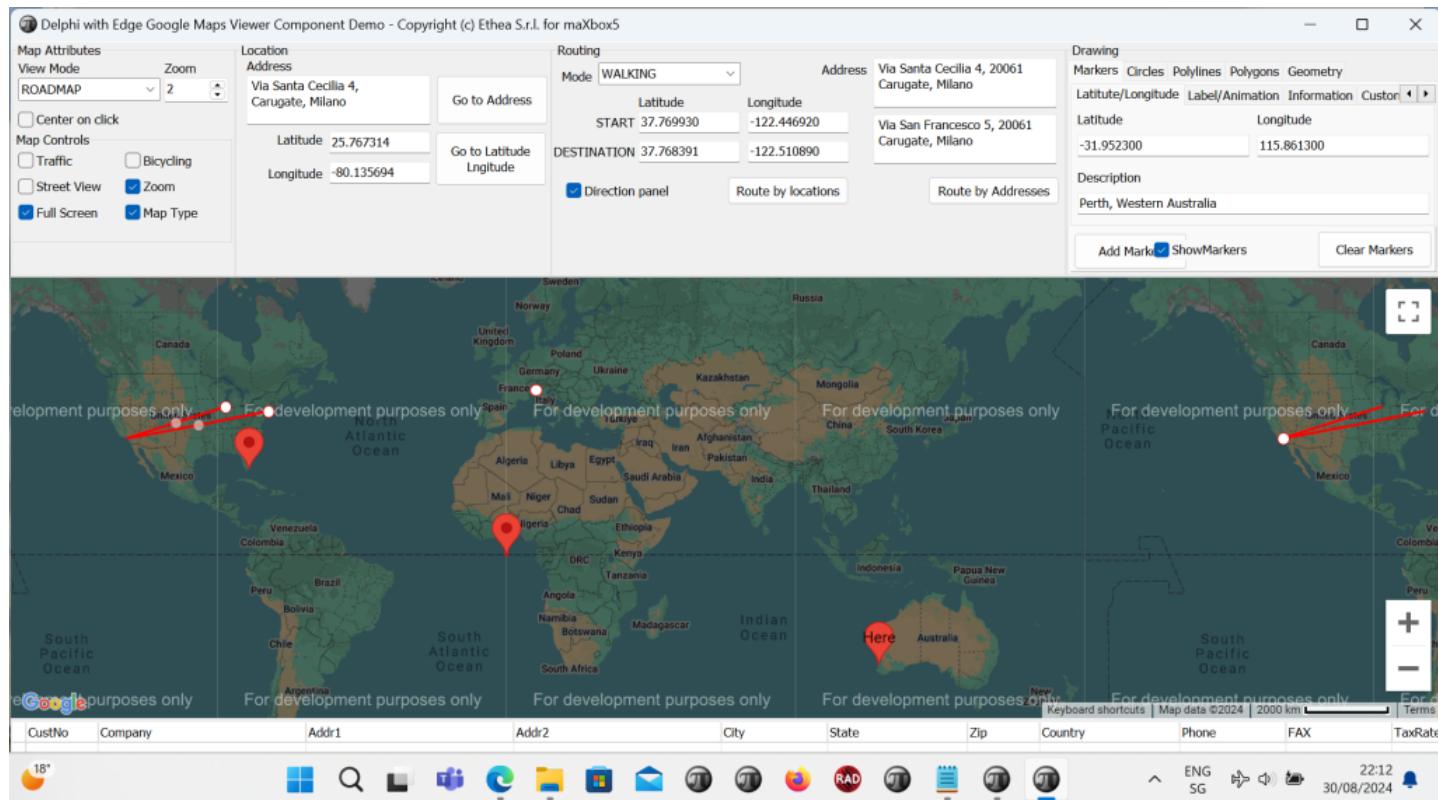
Right Screenshot (Europe Map):
- Route: Rome, Italy to Vienna, Austria.
- Segments:

- 6:53 AM (Wednesday) — 7:53 PM: 13 hr (via Autobahn)
- Napoli Centrale — Bolzano: 6 hr 30 min (via Autobahn)
- Bolzano — Genoa: 1 hr 35 min (via Autobahn)
- Genoa — Vienna: 1 hr 35 min (via Autobahn)

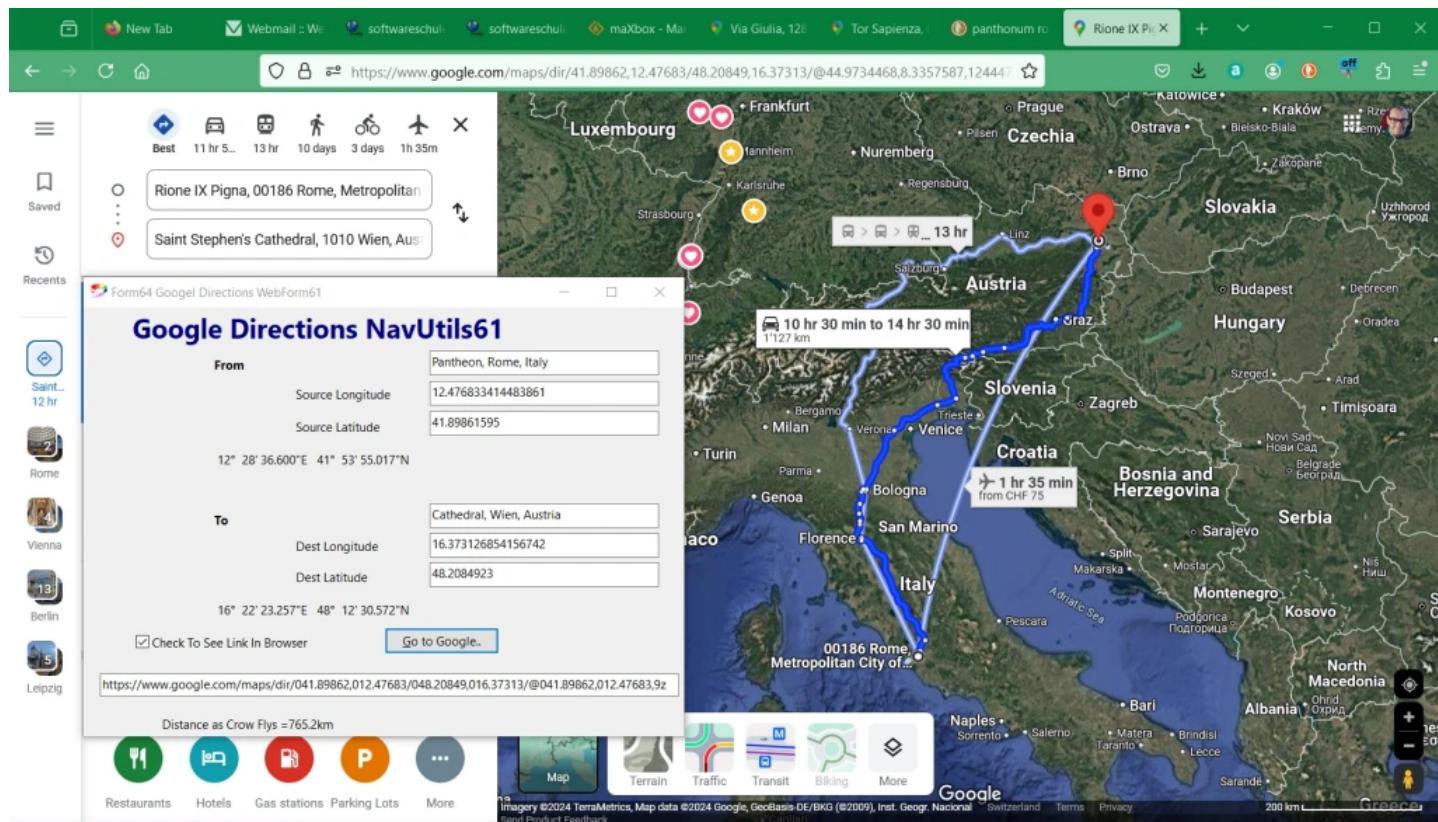
From the WinForm to a WebForm

A major learning curve was how the user interface needs to be adapted for the very small screens and make it scriptable at runtime with maXbox5. I chose a **Multi Layer Tab** control template "Tabbed with Navigation" offered with Tokyo as a basis for my trial app. I had to add the functionality to support the hardware back button and was not very successful with gesture use opting in the end for Forward and Back Speed buttons so more to learn there. The Templates were for the time being not offered with my Rio and Alexandria Community Install.

Another point is to integrate the EdgeView browser with the RAD Studio 10.4 Sydney brings support for working with web content through the Chromium-based Edge WebView2 browser control in VCL applications via the new **TEdgeBrowser** (<http://docwiki.embarcadero.com/Libraries/Sydney/en/Vcl.Edge.TEdgeBrowser>) component.



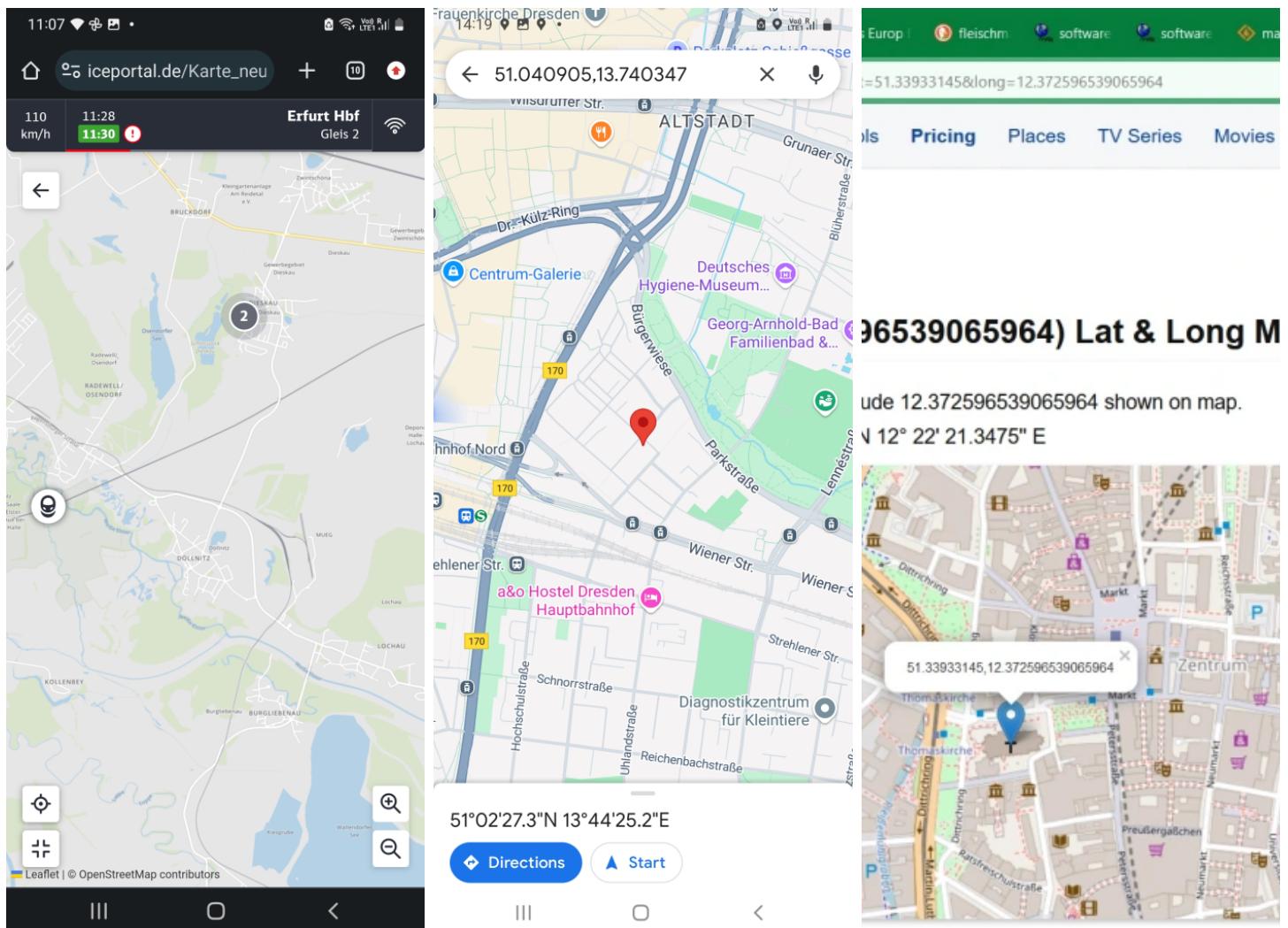
TEdgeBrowser supersedes **TWebBrowser** (<http://docwiki.embarcadero.com/Libraries/Sydney/en/FMX.WebBrowser.TWebBrowser>), which uses the Internet Explorer WebBrowser browser control. However **TWebBrowser** remains in the VCL component set, with some notable changes.



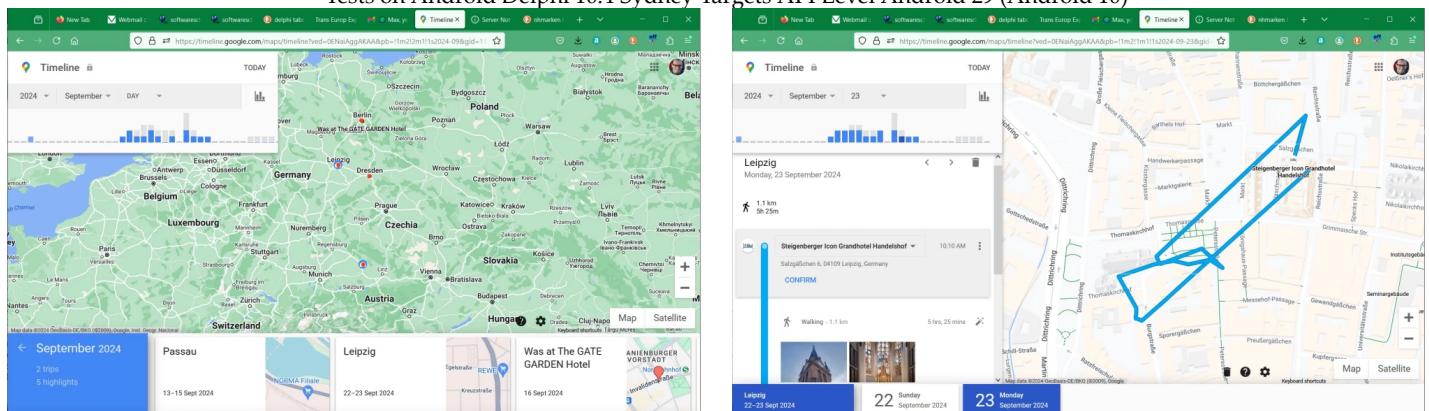
Delphi form as a browser plugin with a forthcoming web assembly

The script you can find at: <https://sourceforge.net/projects/maxbox/files/Examples/EKON/>

[EKON28/1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas/download](https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON28/1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas/download) (https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON28/1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas/download)



Tests on Android Delphi 10.4 Sydney Targets API Level Android 29 (Android 10)



Timeline as an output from directions tracker

15 % Rabatt
für die EKON 28

Meine Session:
Effektive Routenplanung
und Geocoding mit dem
Google Maps API

Max Kleiner | kleiner kommunikation



EKON 28

EUER RABATTCODE:

4. – 6. November 2024
Düsseldorf

MaKl_EKON_28

EKON 28 will be xenomorph

Treffst mich auf der EKON 28! In meiner Session zeige ich, wie du mit Google Directions API und Google Maps Wegbeschreibungen und Routen für verschiedene Transportmittel berechnest und visualisierst. Ich erstelle ein Navigation Advisor Projekt und demonstriere Geocoding sowie die Verwaltung von Diensten und Karten mit APIs und SDKs. Nutze den Rabattcode MaKl_EKON_28 und erhalte 15 % Ermäßigung auf die Tickets.

maXbox5 64-bit ScriptStudio pydemo67.txt

File Program Options View Debug Output Help

Load Find Replace / Refact Go Compile! Use Cases

12 pydemo67.txt

```
391     println('3. Sum of even numbers in a List: '+
392     | | | | | | | evalstr('sum([num for num in a if num%2 == 0])'));
393     execstr('a = [1,2,3,4,5]; del a[1::2]');
394     println('4. Deleting Multiple Elements from a List '+evalstr('a'));
395     //println('5. Reading Files '+
396     | | evalstr('[line.strip() for line in open('''firstdemo.txt'')])');
397
398     println('6. Writing data to file: ')
399     execstr('with open("pydata.txt", ''a'',newline=''\n'') as f: f.write("Python & Pascal are awesome")');
400     //openfile(exepath+'pydata.txt');
401     println('7. Creating Lists '+evalstr('[i for i in range(0,10)]'));
402     println('7. Creating Lists '+evalstr('[("Hi "+i) for i in ["Karl","Abhay","Zen"]]');
403     execstr('text = ''Level''');
404     println('8. Palindrome: '+evalstr('text[::-1]'));
405     execstr('import random');
406     println('10. Simulating Toss of a coin: '+evalstr('random.choice(["Head","Tail"])'));
```

maXbox5 C:\maxbox\maxbox51\examples\pydemo67.txt Ct:27/08/2024 22:32:33 Mem:56% Rtime:0:0:2.810 Thr:22 S

3. Sum of even numbers in a List: 12
4. Deleting Multiple Elements from a List [1, 3, 5]
6. Writing data to file:
7. Creating Lists [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
7. Creating Lists ['Hi Karl', 'Hi Abhay', 'Hi Zen']
8. Palindrome: level
10. Simulating Toss of a coin: Tail
□□□ mX5 executed: 27/08/2024 22:32:34 Runtime: 0:0:2.810 Memload: 56% use
RemObjects Pascal|Script. Copyright (c) 2004-2024 by RemObjects Software & maXbox5

Pydemo67



Interrail in the meantime



Re 4/4 I-IV Track N (Lemaco, Fleischmann, Hobbytrain)

Summary

- The Station2Station tool is a script (1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas) that uses Google Maps to get directions between any two points.
- Users can enter a city or address in the From and To address inputs, or set direct latitude and longitude coordinates. The tool displays the route in the browser, along with turn-by-turn directions, distance, and approximate time it will take to get from one location to the other.
- The GoogleDirForm uses geocoding and Open Street Maps, with a license from OpenStreetMap contributors under ODbL 1.0. Users can get their own API-key for the tool.
- The code behind the tool uses TDirection as a URL Builder, with RNavigateLongLat record as a class to manipulate longitude and latitude references. The tool uses a rest client, THtppRequestC, a JSON Lib, and the OSM API from Nominatim for geocoding.
- Nominatim uses OpenStreetMap data to find locations on Earth by name and address, and can also do reverse geocoding.
- The tool has been tested on Android Delphi 10.4 Sydney Targets API Level Android 29 (Android 10).
- The script is available for download at https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON28/1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas/download.
- The tool's author will be presenting a session at EKON 28, demonstrating how to use Google Directions API and Google Maps to calculate and visualize routes for different modes of transportation. A discount code, MaKI_EKON_28, is available for 15% off tickets to EKON 28.

2Rex

In the following section we can see the integration of weather- or geocode data in our app station2station. We use an API and a RegEx:

```

1 Const
2 UrlWeatherReport25=
3   'https://api.openweathermap.org/data/2.5/weather?q=%s&units=metric&APPID=55013bf3d09cfb0619989a00ed5bedyourAPIKey';
4
5 GEOCoordREX3 =
6   '"lon":([\d\.-]+)."lat":([\d\.-]+)."temp":([\d\.-]+).*';

```

Next we call the API with a simple HTTPGet() to return the JSON String Stream:

```

1  function GetGeoWeather(const location: string;
2                           const UrlRestAPI: string): string;
3  var lStrm: TStringStream;
4  begin
5    lStrm:= TStringStream.Create('');
6    try
7      try
8        HTTPGet(Format(UrlRestAPI,[location]),lStrm);
9      except
10        //if something wrong try using a backup server.
11        //writeln('htmlback: '+GetURLAsString('http://api.openweathermap.org_(http://api.openweathermap.org)'))
12        writeln('Weather_Map Exception: '+Gethtm(UrlWeatherReport25))
13      end;
14      lStrm.Seek(0,0);
15      result:= +(lStrm.ReadString(lStrm.Size));
16    finally
17      lStrm.Free;
18    end;
19  end;

```

With the https result we parse the Utf8 JSON with a RegEx:

```

1  function GEOCoord2PointX2(apistr: string): TFloatPoint;
2  begin
3    setdecimalseparator('.');
4    with TRegEx.Create(GEOCoordREX3,[rroIgnoreCase]) do try
5      if match(apistr).success then begin
6        result.x:= strtofloat(match(apistr).groups[1].value)
7        result.y:= strtofloat(match(apistr).groups[2].value)
8        temp:= strtofloat(match(apistr).groups[3].value)
9      end;
10     finally Free;
11   end;
12 
```

In the end we do have separation of concern in call the API, parse the result and show the data:

```

1  procedure ShowGeoWeather;
2  begin
3    //sr:= GetGeoWeather(removeSpaces('Melbourne,AU'),UrlWeatherReport25);
4    sr:= GetGeoWeather(removeSpaces('Bern, CH'),UrlWeatherReport25);
5    writeln(formatjson(sr))
6    // writeln('GEO_Weather_Report5: '+sr);
7    lonf:= GEOCoord2PointX2(sr).x
8    latf:= GEOCoord2PointX2(sr).y
9    PrintF('Lon: %.4f - Lat: %.4f',[lonf, latf])
10   writ('geowather coords: '+format('Lon: %.4f -Lat: %.4f -temp:%.2f°',
11                                         [lonf, latf, temp]));
12 
```

Beginner RegEx question. I have lines of JSON in a textfile, each with slightly different Fields like coords or weather array, but there are 3 fields I want to extract for each object line if it has it, ignoring everything else. How would I use a regex (in editpad, script, shell or anywhere else) to do this?

Don't try to parse JSON yourself. This is a solved problem. People have already written, tested and debugged code that handles this already. You can use a JSON Parser or a RegEx Group!

This next with a JSON Parser:

```

1  function GEOCoord2PointX3(apistr: string): TFloatPoint;
2  begin
3    setdecimalseparator('.');
4    with TMcJsonItem.Create do begin
5      AsJSON:= apistr;
6      result.x:= at(0,'lat').asnumber;
7      result.y:= at(0,'lon').asnumber;
8      temp:= at(3,'temp').asnumber;
9      free;
10    end;
11 
```

Get a key-value JSON with a RegEx, the following regex expression extract exactly the "fid" field value "321":

```

1  regs:= '{"fid":"321","otherAttribute":"value"} ';
2  writeln(formatjson(regs))
3  writ(regexFindall(regs, '(?<=\"fid\":\"')(?:\\")[^"]*'));

```

```

1  {
2      "coord": {
3          "lon": 7.4474,
4          "lat": 46.9481
5      },
6      "weather": [
7          {
8              "id": 500,
9              "main": "Rain",
10             "description": "light rain",
11             "icon": "10n"
12         }
13     ],
14     "base": "stations",
15     "main": {
16         "temp": 11.03,
17         "feels_like": 10.33,
18         "temp_min": 10.27,
19         "temp_max": 12.74,
20         "pressure": 1011,
21         "humidity": 82,
22         "sea_level": 1011,
23         "grnd_level": 939
24     },
25     "visibility": 10000,
26     "wind": {
27         "speed": 0.81,
28         "deg": 142,
29         "gust": 0.47
30     },
31     "rain": {
32         "1h": 0.52
33     },
34     "clouds": {
35         "all": 60
36     },
37     "dt": 1728239017,
38     "sys": {
39         "type": 2,
40         "id": 2012960,
41         "country": "CH",
42         "sunrise": 1728192953,
43         "sunset": 1728234020
44     },
45     "timezone": 7200,
46     "id": 2661552,
47     "name": "Bern",
48     "cod": 200
49 }

```

The screenshot shows the Json Parser Online interface. On the left, the original JSON code is displayed in a code editor-like window. On the right, two panes show the parsed JSON structure. The top pane, titled 'String parse', shows the JSON with some parts collapsed using the '...' button. The bottom pane, titled 'JS eval', shows the full expanded JSON object. The JSON structure includes coordinates, weather conditions, temperature, humidity, pressure, visibility, wind speeds, rainfall, cloud coverage, system details, and timestamps.

```

{
    "coord": {
        "lon": 7.4474,
        "lat": 46.9481
    },
    "weather": [
        {
            "id": 801,
            "main": "Clouds",
            "description": "few clouds",
            "icon": "02n"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 10.74,
        "feels_like": 10.01,
        "temp_min": 9.42,
        "temp_max": 12.59,
        "pressure": 1011,
        "humidity": 82,
        "sea_level": 1011,
        "grnd_level": 939
    },
    "visibility": 10000,
    "wind": {
        "speed": 1,
        "deg": 138,
        "gust": 0.75
    },
    "clouds": {
        "all": 16
    },
    "dt": 1728240961,
    "sys": {
        "type": 2,
        "id": 2012960,
        "country": "CH",
        "sunrise": 1728192953,
        "sunset": 1728234020
    },
    "timezone": 7200,
    "id": 2661552,
    "name": "Bern",
    "cod": 200
}

```

JSON Parser Online

```
"dt": 1728209107,  
"sys": {  
"type": 1,  
"id": 6937,  
"country": "CH",  
"sunrise": 1728192953,  
"sunset": 1728234020  
},  
"timezone": 7200,  
"id": 2661552,  
"name": "Bern",  
"cod": 200  
}  
Lon: 7.4474 - Lat: 46.9481  
geoweather coords: Lon: 7.4474 - Lat: 46.9481 -temp:9.01°  
mX5 EXECUTED: 06/10/2024 12:08:21 Runtime: 0:0:3.244 Memload: 59% use
```

JSON Parser as RegEx

1 | This regular expression finds a key-value pair in JSON formatted strings

For those who want to see that a **TMachCollection** are matches of groups I found a stackoverflow testdocument in which we convert the RegEx in a TMatchCollection:

```

1 const TESTJSON =
2 [
3   {
4     "_id": "56af331efbeca6240c61b2ca",           '+LF+
5     "index": 120000,                            '+LF+
6     "guid": "bedb2018-c017-429E-b520-696ea3666692", '+LF+
7     "isActive": false,                          '+LF+
8     "balance": "$2,202,350",                   '+LF+
9     "object": {                                '+LF+
10       "name": "amx",                           '+LF+
11       "lastname": "lang"                      '+LF+
12     }                                         '+LF+
13   }                                         '+LF+
14 ]                                         ;
15
16 writeln(regexFindall(TESTJSON,
17   '(?:\"|\\')(?<key>[^"]*)(?:\"|\\')(?=:(?::\s*)(?::\"|\\")?(?<value>true|false|[0-9a-zA-Z\+\-\,\.\$\"]*))');
18
19 >>> 8 Matches [ ]:
20 [ "_id": "56af331efbeca6240c61b2ca" ][ "index": 120000, ]
21 [ "guid": "bedb2018-c017-429E-b520-696ea3666692" ]
22 [ "isActive": false ][ "balance": "$2,202,350" ][ "object": ]
23 [ "name": "am" ][ "lastname": "lang" ]
24
25 with TRegEx.Create(sr,[rroIgnoreCase]) do try
26   mats:= Matches(TESTJSON);
27   //for group in matches do
28   for m:= 0 to mats.count-1 do begin
29     for g:= 1 to mats[it].groups.count-1 do
30       writeln(format('matchgrp[%d,%d]: %s',[m,g,mats[m].groups[g].value]));
31     end;
32     writeln('match group[direct] balance: '+mats[4].groups[2].value);
33   finally Free;
34 end;
35
36 >>> 8 matches each 3 groups
37 matchgrp[0,1]: _id
38 matchgrp[0,2]: 56af331efbeca6240c61b2ca
39 matchgrp[1,1]: index
40 matchgrp[1,2]: 120000,
41 matchgrp[2,1]: guid
42 matchgrp[2,2]: bedb2018-c017-429E-b520-696ea3666692
43 matchgrp[3,1]: isActive
44 matchgrp[3,2]: false
45 matchgrp[4,1]: balance
46 matchgrp[4,2]: $2,202,350
47 matchgrp[5,1]: object
48 matchgrp[5,2]:
49 matchgrp[6,1]: name
50 matchgrp[6,2]: amx
51 matchgrp[7,1]: lastname
52 matchgrp[7,2]: lang
53 match group[direct] balance: $2,202,350
54 mX5 executed: 06/10/2024 16:49:24 Runtime: 0:0:3.482 Memload: 57% use
55 https://regex101.com/r/zR2vU9/4

```



Märklin 3067.5, from '1975 to '1976, body Red/Black DSB inscription fine

Posted in [EKON](#), [Engineering](#), [Geocoding](#), [maXbox](#), [Python](#), [Trains](#) Tagged [API](#), [Coding](#), [data-science](#), [gis](#), [Internetrail](#), [Python](#), [Travel](#), [tutorial](#) [4 Comments](#)

4 thoughts on “Station2Station”

1.

[maxbox4](#) says:

[October 3, 2024 at 8:14 am](#) [Edit](#)

AI paraphrasing poses a significant risk to academic integrity in that it erodes the principles of academic integrity by promoting deception and dishonesty. It undermines the trust and credibility of educational institutions and devalues the importance of original thought and scholarly discourse.

[AI paraphrasing detection: Strengthening the integrity of academic writing \(turnitin.com\)](#)

[REPLY ▾](#)

[maxbox4](#) says:

[October 3, 2024 at 2:53 pm](#) [Edit](#)

A **functor** is something you can map a function over. It's like applying a transformation to each item in a container without touching the container itself. It's more than just functional programming — it's **powerful composition**.

2. [REPLY ▾](#)

[maxbox4](#) says:

[October 4, 2024 at 7:48 am](#) [Edit](#)

For me the point is not that fake news exist, its worse that we cannot differentiate from real news!

Paraphrase example

For me, it's not about the fact that there is fake news, but worse still, that we cannot distinguish it from real news.

[REPLY ▾](#)

[maxbox4](#) says:

[October 6, 2024 at 3:27 pm](#) [Edit](#)

Want to know more about the thriving AI scene in the Greater Zurich Area?

<https://www.greaterzuricharea.com/en/artificial-intelligence-greater-zurich-area>

[Blog at WordPress.com.](#)