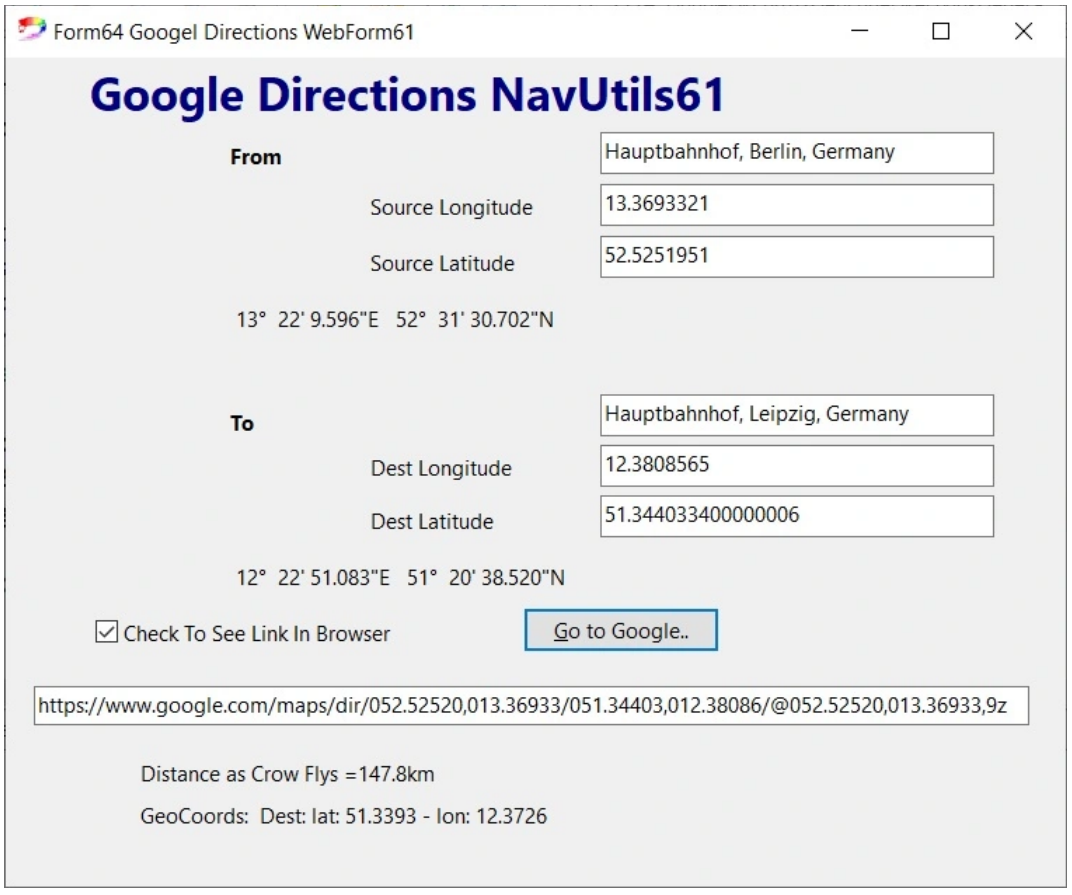
Welcome to maXbox Code (Blog)

maXbox

Station2Station

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

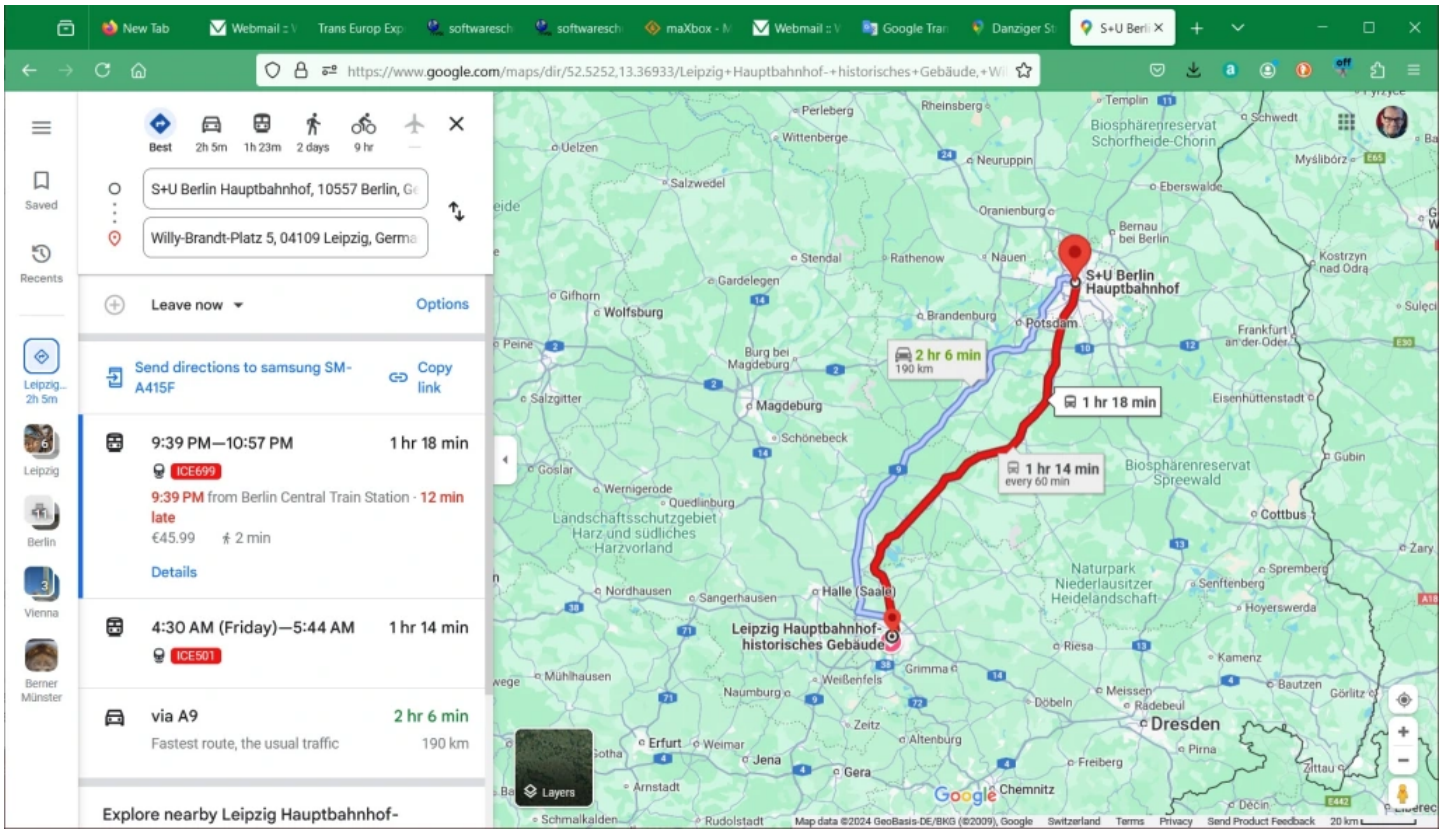
Use this tool as a script (*1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas*) to get the directions between any point using google maps. Enter a city or an address in both the **From** and the **To** address inputs. You can also set direct an lat- and longitude. Click Got to Google… (Find Directions), 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.



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.

```
procedure TDirectionsBtnGoGoogleClick(Sender: TObject);
Var
  LocFrom,LocTo:RNavigateLongLat;
  Long,Lat:Double;
begin
  LocFrom:=RNavigateLongLat.create;
  Locto:=RNavigateLongLat.create;
  Long:=RealFrmDegreeText(EdtLong.Text);
  Lat:= RealFrmDegreeText(EdtLat.Text);
  LocFrom.CreateDec(Long,Lat);
  Long:=RealFrmDegreeText(EdtLong2.Text);
  Lat:= RealFrmDegreeText(EdtLat2.Text);
  LocTo.CreateDec(Long,Lat);
  // { 0 Start 1 End 3 Center }
  EdtGoogleLink.Text:=LocFrom.GoogleLinkDirectionsTo(LocTo, 0);
  LBlCrowFlies.Caption:='Distance as Crow Flys ='+FormatFloat('0.0km',LocTo.MetresFrom(LocFrom)/1000);
  if CBxGoNow.Checked then
     LocFrom.GoGoogleDirectionsTo(LocTo, 0);
end;
```

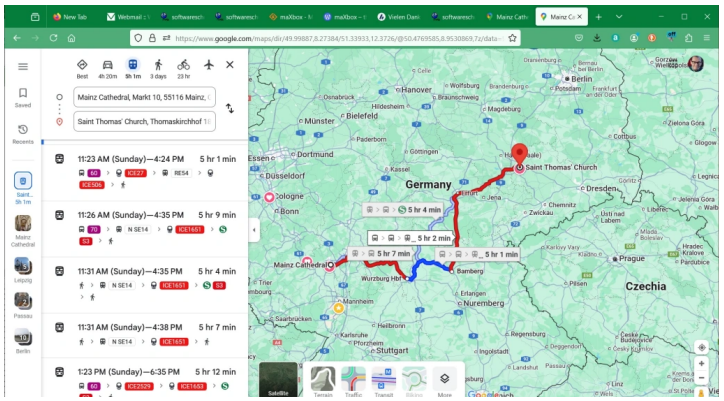
For the geocoding we use a restclient **THttpRequestC** a JSON Lib and the OSM API from nominatim. Nominatim uses **OpenStreetMap** data to find locations on Earth by name and address (geocoding). It can also do the reverse, find an address for any location on the planet. For occasional use. Use the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

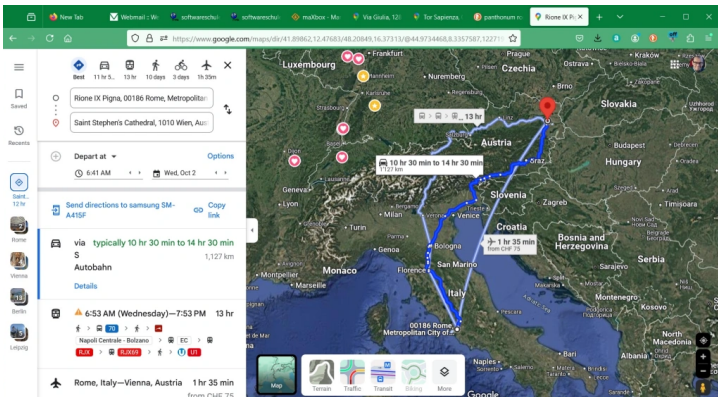
```
function TAddressGeoCodeOSM8(AURL, location, aApikey: string): tlatlong;
var Httpreq: THttpRequestC; httpres: string;
    Body: TMultipartFormBody;
    jsn: TMcJsonItem;
begin
  httpreq:= THttpRequestC.create(self);
  httpreq.headers.add('Accept: application/json; charset=utf-8');
  httpreq.useragent:= USERAGENT4;
  httpreq.SecurityOptions:= [soSsl3, soPct, soIgnoreCertCNInvalid];
  try
    if httpreq.get(Format(AURL,[location])) then begin
       //httpres:= (httpreq.Response.ContentAsString)
       httpres:= (httpreq.Response.ContentAsUTF8String)
       //writeln('conttype '+httpreq.Response.ContentType);
       writ('debug back '+formatJson(httpres));
       jsn:= TMcJsonItem.Create;
       jsn.AsJSON:= httpres;
       result.lat:= jsn.at(0,'lat').asnumber;
       result.long:= jsn.at(0,'lon').asnumber;
       result.descript:= Format('Coords: lat %2.5f  lng %2.5f %s place_id: %d',
                                [result.lat,result.long,jsn.at(0,'display_name').asstring,
                                                        jsn.at(0,'place_id').asinteger]);
    end else Writeln('APIError '+inttostr(Httpreq.Response.StatusCode2));
    //StrReplace(httpres, '[{', '{');
  finally
    writeln('Status3: '+gethttpcod(httpreq.Response.statuscode2))
    httpreq.Free;
    sleep(200)
    jsn.Free;
  end;
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:

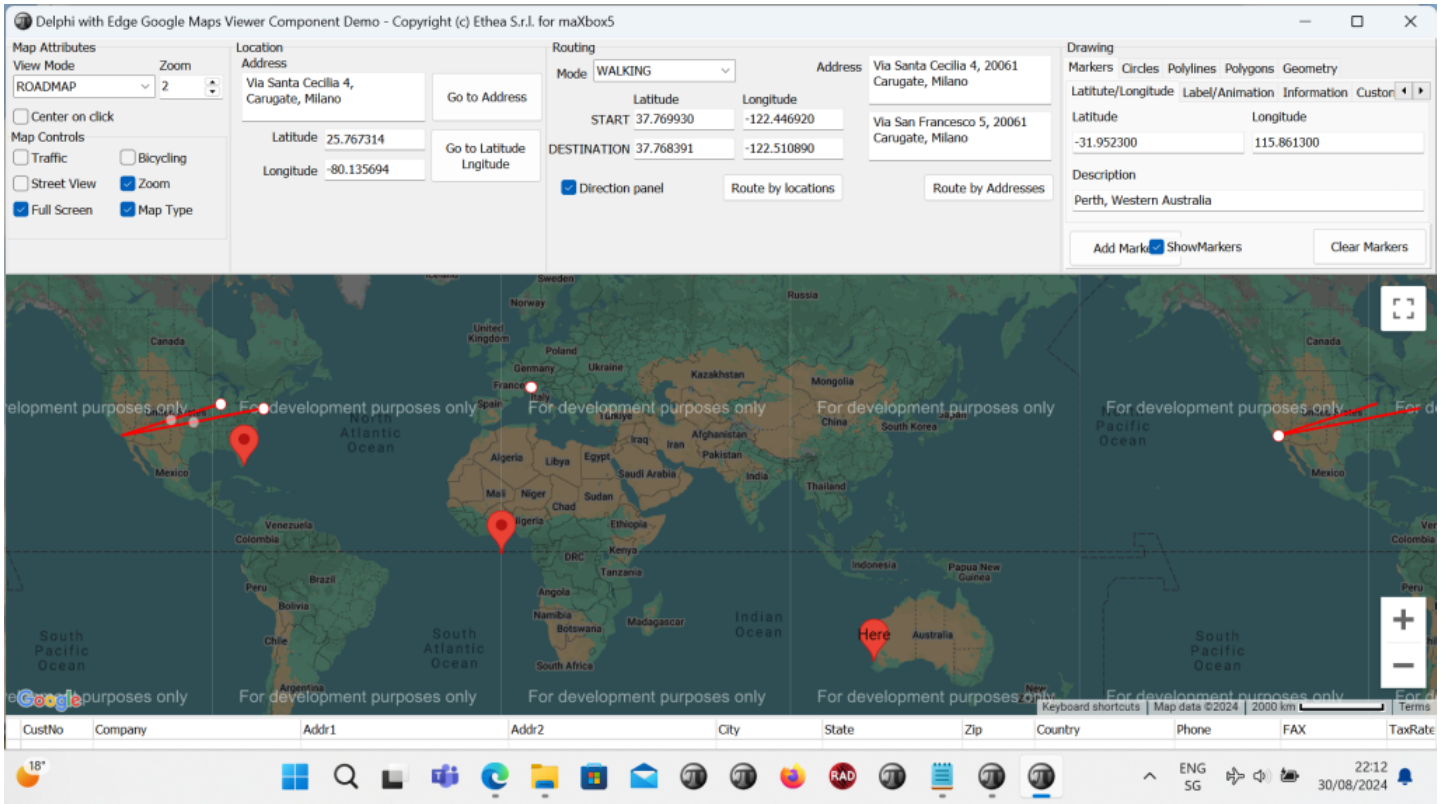




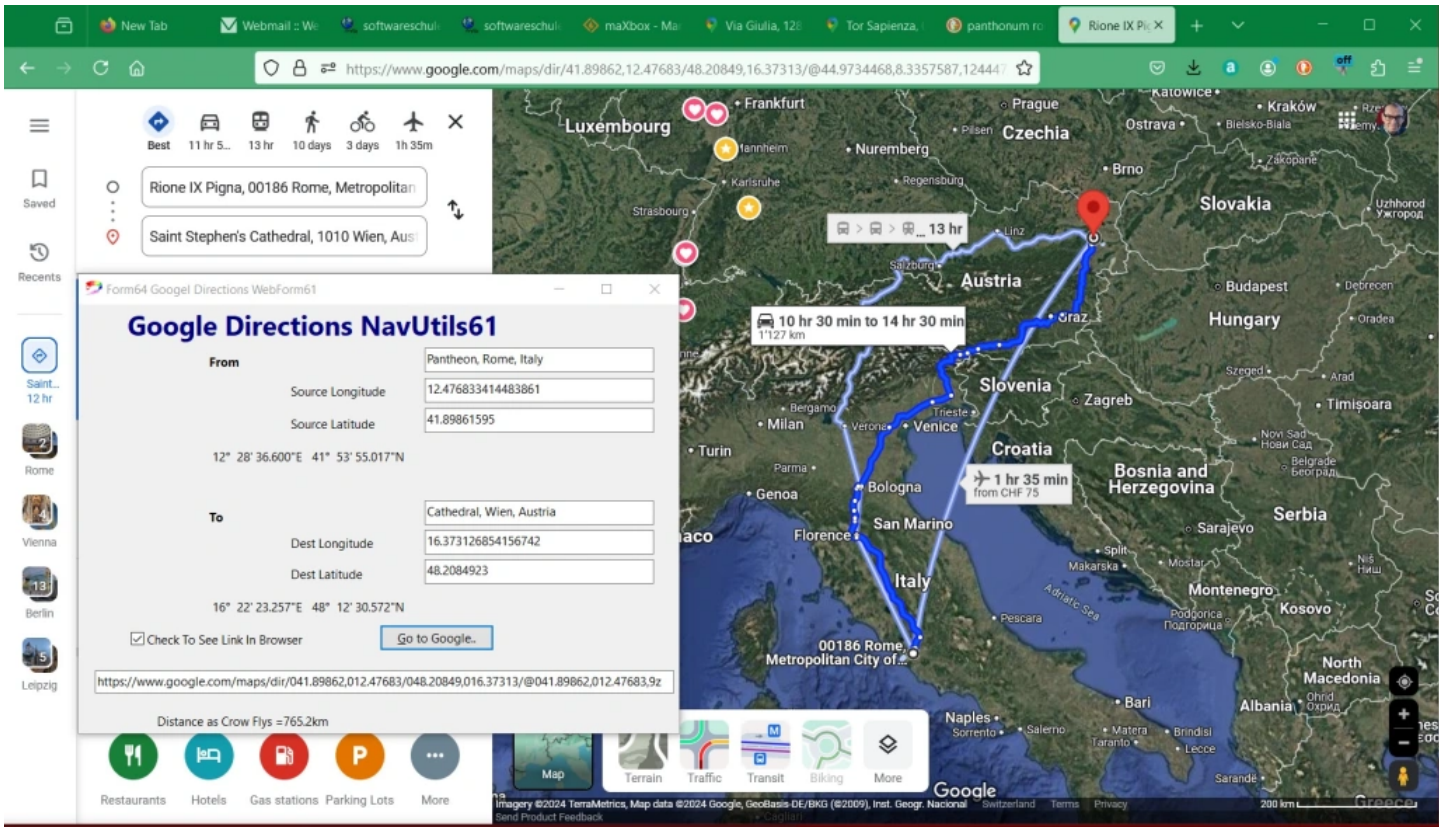
From the WinForm to a WebForm

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 trail 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.



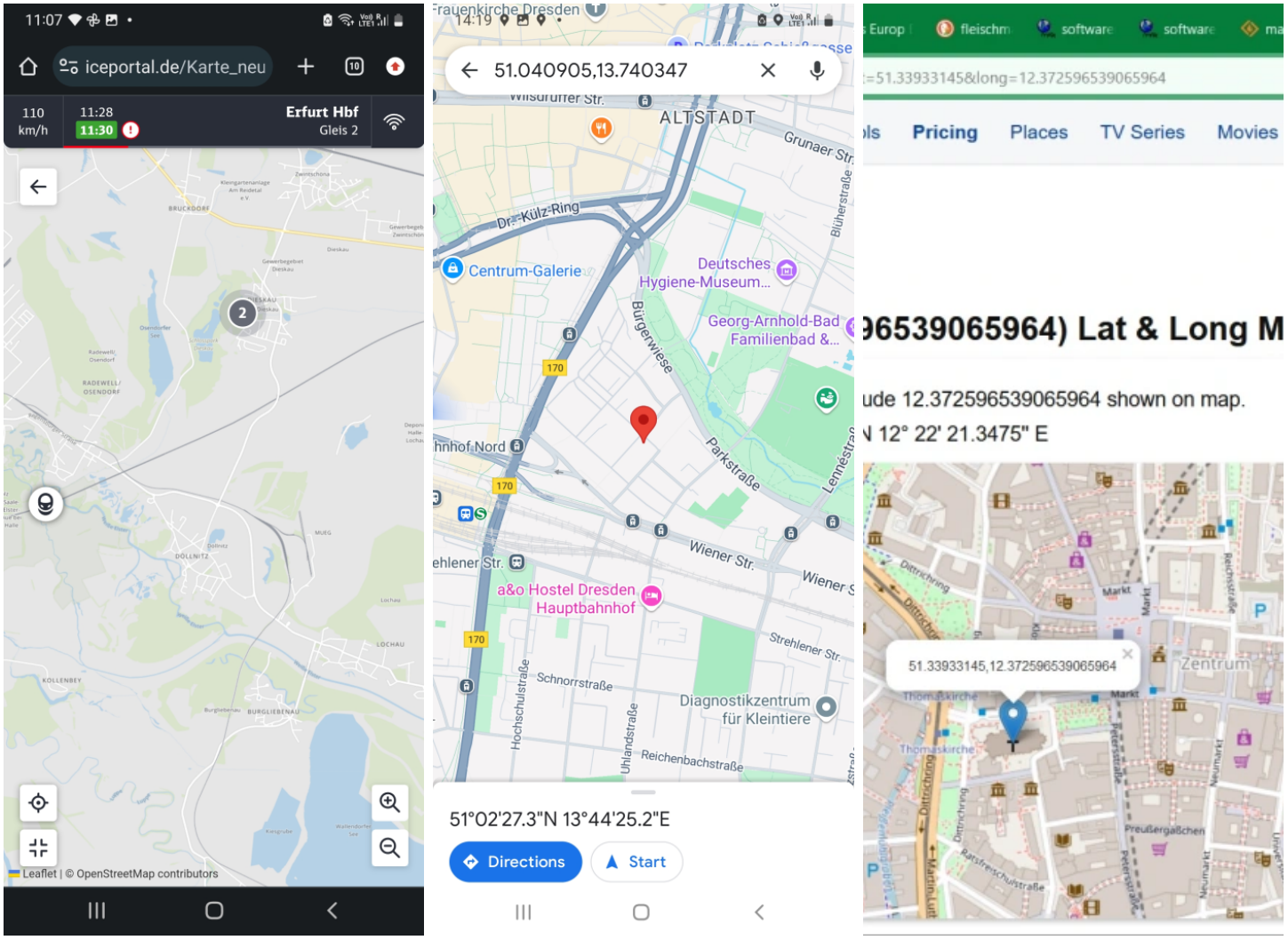
TEdgeBrowser supersedes **TWebBrowser (http://docwiki.embarcadero.com/Libraries/Sydney/en/FMX.WebBrowser.TWebBrowser)**, which uses the Internet Explorer WebBrowser browser control. However TWebBrowser remains in the VCL component set, with some notable changes.



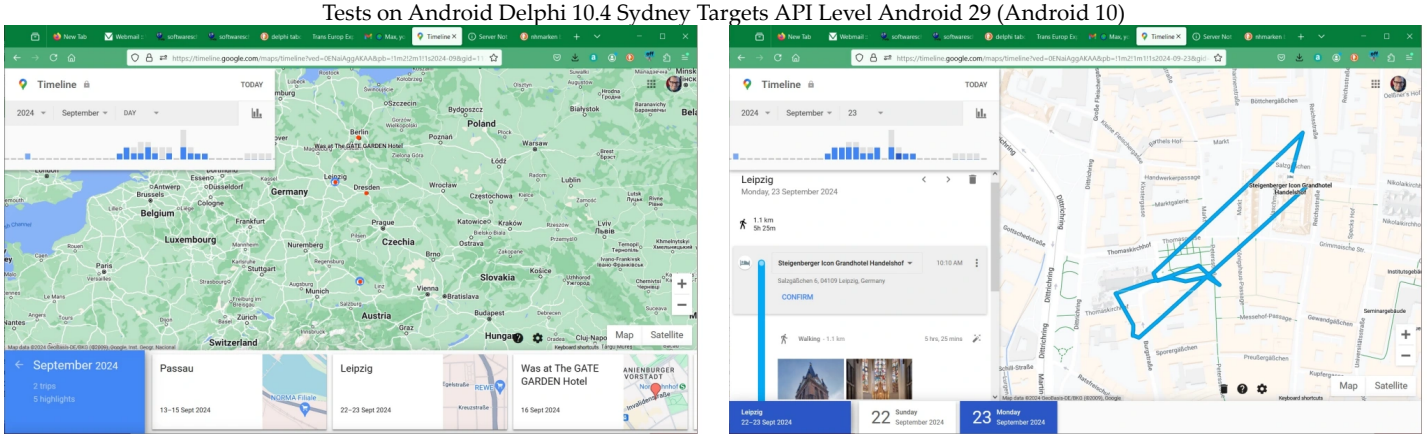
Delphi form as a browser plugin with a forthcoming web assembly

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

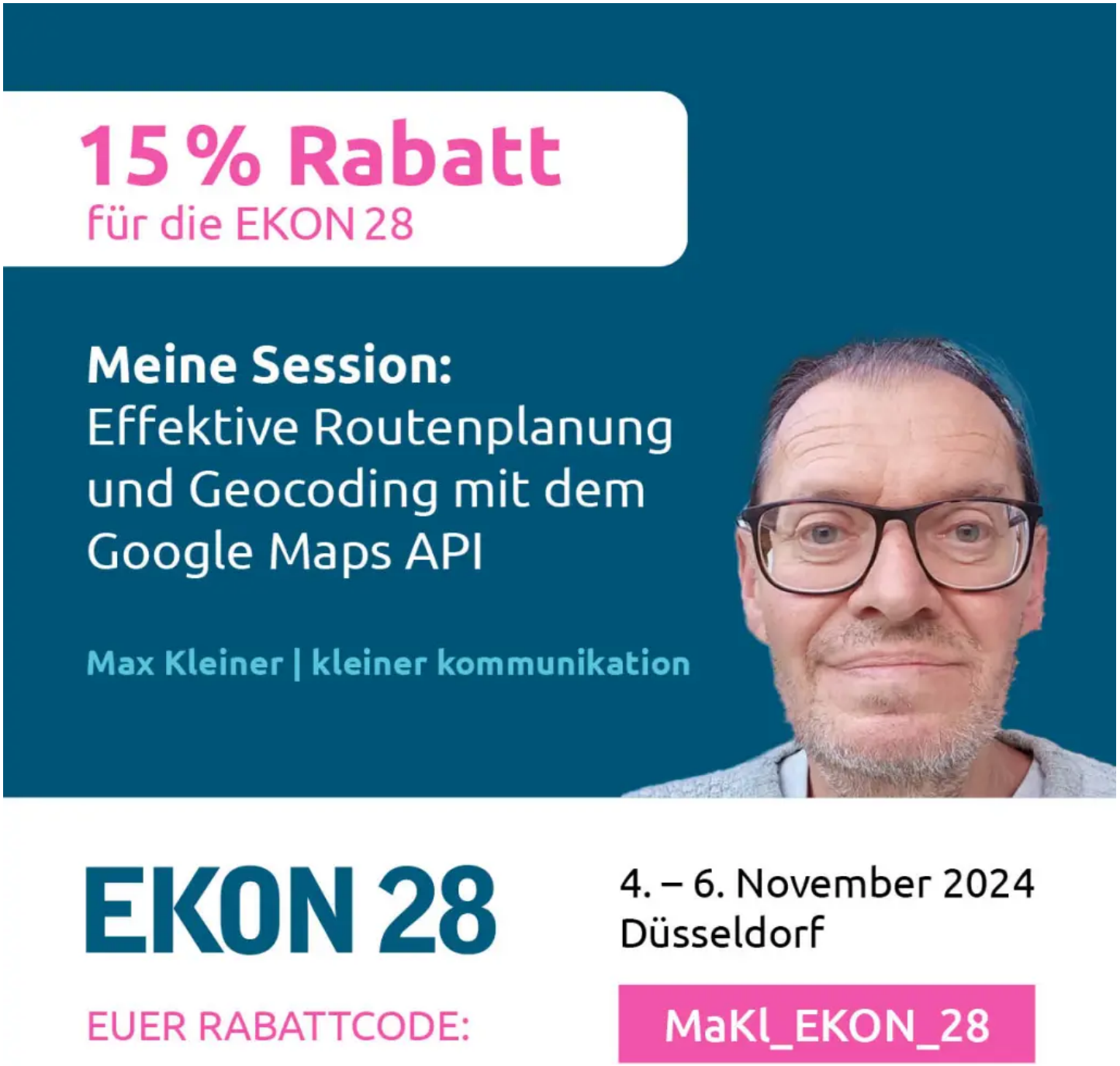
EKON28/1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas/download (https://sourceforge.net/projects/maxbox/files/Examples/EKON/EKON28/1274_GoogleDirForm2GeocodeDirectionsGeneral2request62.pas/download)



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

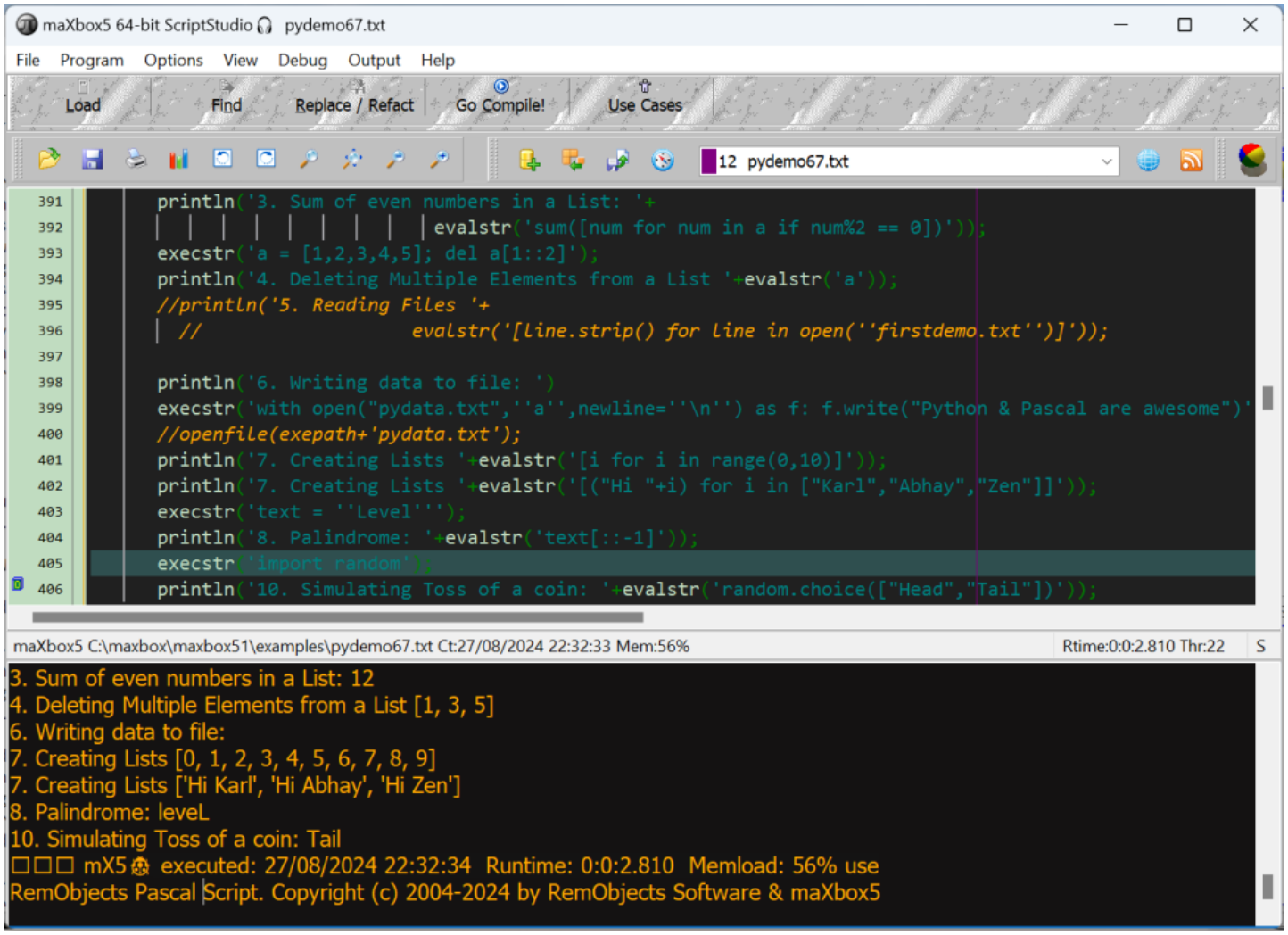


Timeline as an output from directions tracker



EKON 28 will be xenomorph

Trefft 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.



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, THttpRequestC, 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 (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, MaKl_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:

```
Const
  UrlWeatherReport25=
     'https://api.openweathermap.org/data/2.5/weather?q=%s&units=metric&APPID (https://api.openweathermap.org/d
                          '=55013bf3d09cfb0619989a00ed5bed_____yourAPIKey';
  GEOCoordREX3 =
               '"lon":([\d\.-]+).*"lat":([\d\.-]+).*"temp":([\d\.-]+).*';
```

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

```
function GetGeoWeather(const location: string;
                               const UrlRestAPI: string): string;
 var lStrm: TStringStream;
begin
  lStrm:= TStringStream.Create('');
  try
     try
       HTTPGet(Format(UrlRestAPI,[location]),lStrm);
     except
        //if something wrong try using a backup server.
        //writeln('htmlback: '+GetURLAsString('http://api.openweathermap.org (http://api.openweathermap.org)')
       writeln('OWeather_Map Exception: '+Gethtm(UrlWeatherReport25))
     end;
     lStrm.Seek(0,0);
     result:= +(lStrm.ReadString(lStrm.Size));
  finally
     lStrm.Free;
  end;
end;
```

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

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

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

```
procedure ShowGeoWeather;
begin
  //sr:= GetGeoWeather(removespaces('Melbourne,AU'),UrlWeatherReport25);
  sr:= GetGeoWeather(removespaces('Bern, CH'),UrlWeatherReport25);
  writeln(formatjson(sr))
  // writeln('GEO_Weather_Report5: '+sr);
  lonf:= GEOCoord2PointX2(sr).x
  latf:= GEOCoord2PointX2(sr).y
  PrintF('Lon: %.4f - Lat: %.4f ',[lonf, latf])
  writ('geoweather coords: '+format('Lon: %.4f -Lat: %.4f -temp:%.2f°',
                                                [lonf,latf,temp]));
end;
```

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!

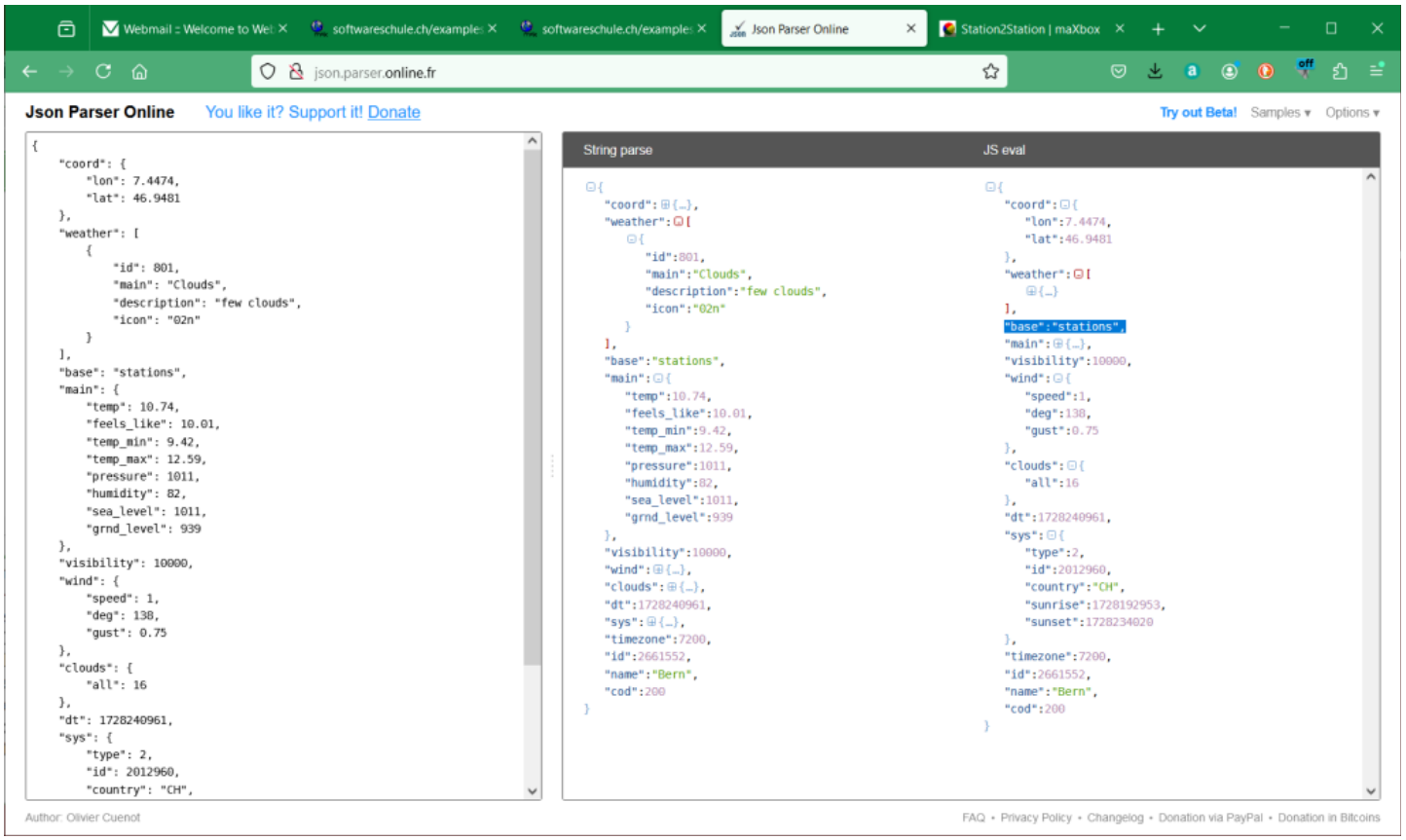
This next with a JSON Parser:

```
function GEOCoord2PointX3(apistr: string): TFloatPoint;
   begin
     setdecimalseparator('.')
     with TMcJsonItem.Create do begin
       AsJSON:= apistr;
       result.x:= at(0,'lat').asnumber;
       result.y:= at(0,'lon').asnumber;
       temp:= at(3,'temp').asnumber;
       free;
     end;
   end;
```

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

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

```
{
    "coord": {
        "lon": 7.4474,
        "lat": 46.9481
    },
    "weather": [
        {
            "id": 500,
            "main": "Rain",
            "description": "light rain",
            "icon": "10n"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 11.03,
        "feels_like": 10.33,
        "temp_min": 10.27,
        "temp_max": 12.74,
        "pressure": 1011,
        "humidity": 82,
        "sea_level": 1011,
        "grnd_level": 939
    },
    "visibility": 10000,
    "wind": {
        "speed": 0.81,
        "deg": 142,
        "gust": 0.47
    },
    "rain": {
        "1h": 0.52
    },
    "clouds": {
        "all": 60
    },
    "dt": 1728239017,
    "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:

```
const TESTJSON =
'[                                                '+LF+
'  {                                              '+LF+
'    "_id": "56af331efbeca6240c61b2ca",           '+LF+
'    "index": 120000,                             '+LF+
'    "guid": "bedb2018-c017-429E-b520-696ea3666692", '+LF+
'    "isActive": false,                                '+LF+
'    "balance": "$2,202,350",                          '+LF+
'    "object": {                                       '+LF+
'        "name": "amx",                           '+LF+
'        "lastname": "lang"                       '+LF+
'    }                                            '+LF+
'  }                                              '+LF+
']                                                ';

writ(regexFindall(TESTJSON,
   '(?:\"|\'')(?<key>[^"]*)(?:\"|\'')(?=:)(?:\:\s*)(?:\"|\'')?(?<value>true|false|[0-9a-zA-Z\+\-\,\.\$]*)'));

>>> 8 Matches [ ]:
[ "_id": "56af331efbeca6240c61b2ca ][ "index": 120000, ]
[ "guid": "bedb2018-c017-429E-b520-696ea3666692 ]
[ "isActive": false ][ "balance": "$2,202,350 ][ "object":  ]
[ "name": "am ][ "lastname": "lang ]

with TRegEx.Create(sr,[rroIgnoreCase]) do try
   mats:= Matches(TESTJSON);
   //for group in matches do
   for m:= 0 to mats.count-1 do begin
     for g:= 1 to mats[it].groups.count-1 do
       writ(format('matchgrp[%d,%d]: %s ',[m,g,mats[m].groups[g].value]));
   end;
  writ('match group[direct] balance: '+mats[4].groups[2].value);
 finally Free;
end;

>>> 8 matches each 3 groups
matchgrp[0,1]: _id
matchgrp[0,2]: 56af331efbeca6240c61b2ca
matchgrp[1,1]: index
matchgrp[1,2]: 120000,
matchgrp[2,1]: guid
matchgrp[2,2]: bedb2018-c017-429E-b520-696ea3666692
matchgrp[3,1]: isActive
matchgrp[3,2]: false
matchgrp[4,1]: balance
matchgrp[4,2]: $2,202,350
matchgrp[5,1]: object
matchgrp[5,2]:
matchgrp[6,1]: name
matchgrp[6,2]: amx
matchgrp[7,1]: lastname
matchgrp[7,2]: lang
match group[direct] balance: $2,202,350
mX5 executed: 06/10/2024 16:49:24  Runtime: 0:0:3.482  Memload: 57% use
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