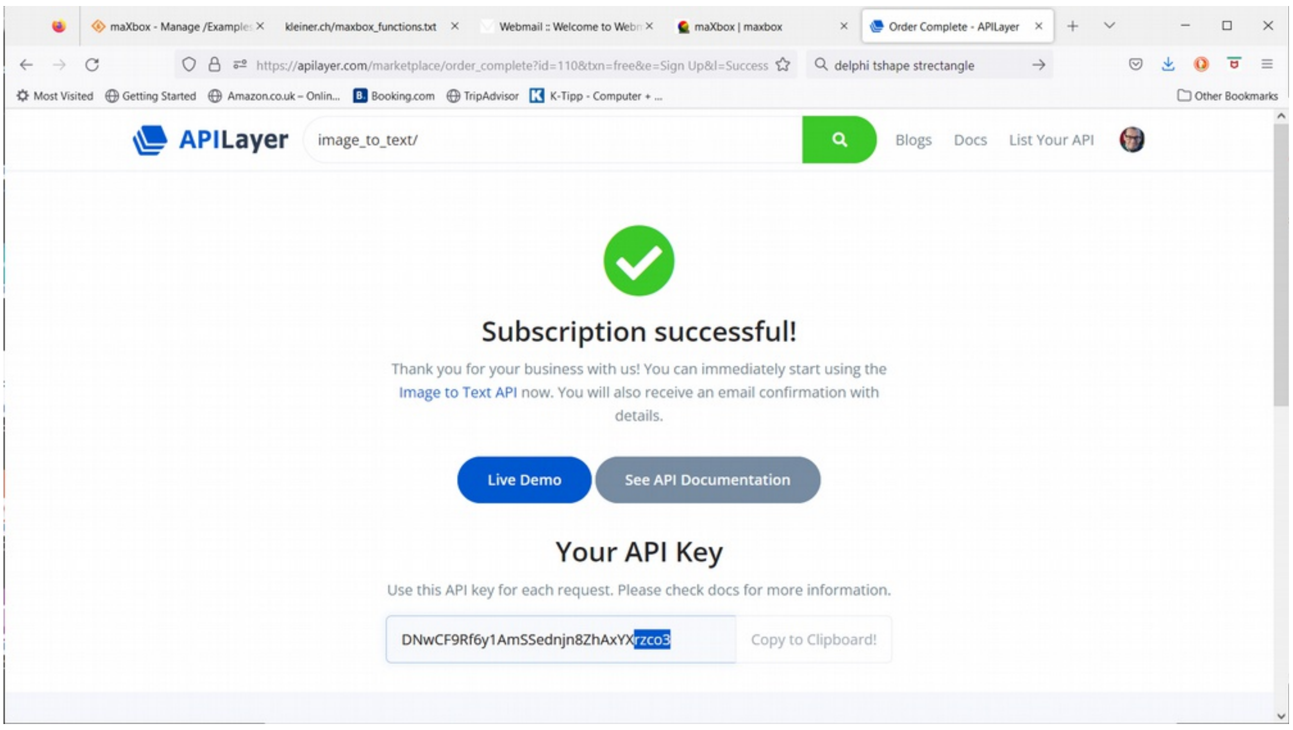
Image to Text API

maXbox Starter 103 – Text recognition of characters in images.

"A picture is worth a thousand words.
An interface is worth a thousand pictures.".

This API recognizes and reads a text embedded in pictures or photos. Image to Text API uses a neural net (LSTM) based OCR engine which is focused on line recognition, but also supports recognizing the character patterns. It supports both handwriting and printed materials as well as street maps.
APILayer is an API marketplace where also your API can reach a broader audiences, but first you need an API-key for free:

The result of a simple subscription will be the screenshot below:



Subscription successful!

Thank you for your business with us! You can immediately start using the Image to Text API now. You will also receive an email confirmation with details.

Your API Key

Use this API key for each request. Please check docs for more information.

DNwCF9Rf6y1AmSSednjn8ZhAxYXrzco3

Pic: 1176_apilayer_reg.png

This register allows you a monthly usage of 30 successful calls. Almost all API's has a free plan to subscribe. Looking at the following book-cover, it will extract the text information easily, even though the cover has shadows and positioned with angle.

We use *WinHttp.WinHttpRequest*, *JSONObjects* and *TGraphics* library with loading and testing the REST-client. Also we pass the API-key as a request-header, so get a key first at: https://apilayer.com/marketplace

You can also use a powerful 'OCR' feature (text in picture recognition) to extract text from an image during the conversion process. In this case, you will get an editable text document as a result that you can adjust and modify as you need.

The data represents is JSON data with all the text extracted and even the language of the text to scan is auto detected. Before we dive into code this is the main part of the script:

```
function Image_to_text_API2(AURL, url_imgpath, aApikey: string): string;
var httpq: THttpConnectionWinInet;
    rets: TStringStream;
    heads: TStrings; iht:IHttpConnection2;
begin
  httpq:= THttpConnectionWinInet.Create(true);
  rets:= TStringStream.create('');
  heads:= TStringlist.create;
   try
     heads.add('apikey='+aAPIkey);
     iht:= httpq.setHeaders(heads);
     httpq.Get(Format(AURL,[url_imgpath]),rets);
     if httpq.getresponsecode=200 Then result:= rets.datastring
       else result:='Failed:'+
              itoa(Httpq.getresponsecode)+Httpq.GetResponseHeader('message');
   except
     writeln('EWI_HTTP: '+ExceptiontoString(exceptiontype,exceptionparam));
   finally
     httpq:= Nil;
     heads.Free;
     rets.Free;
   end;
end;
```

The main part function opens a connection, invokes the API and results a stream which we convert to a datastring.
Image2Text or Image to Text live demo is providing an API service on its APILayer publication platform. Live Demo feature allows you to test the API within your browser; no need to install or code anything. You can modify all the parameters as you like and interact with the API from many languages. The API export format is JSON, e.g. our book cover see below:

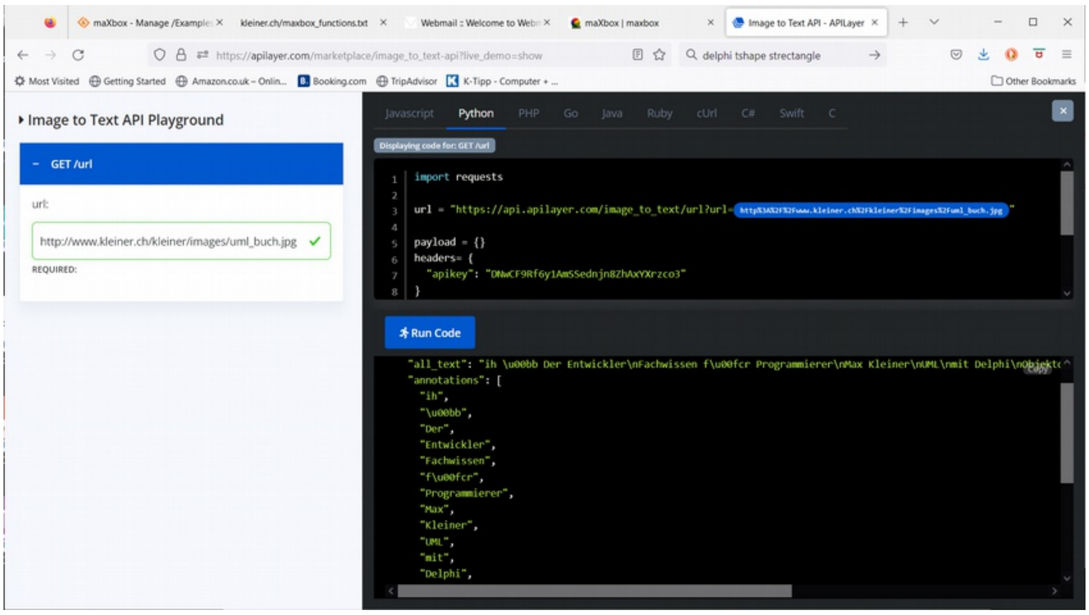


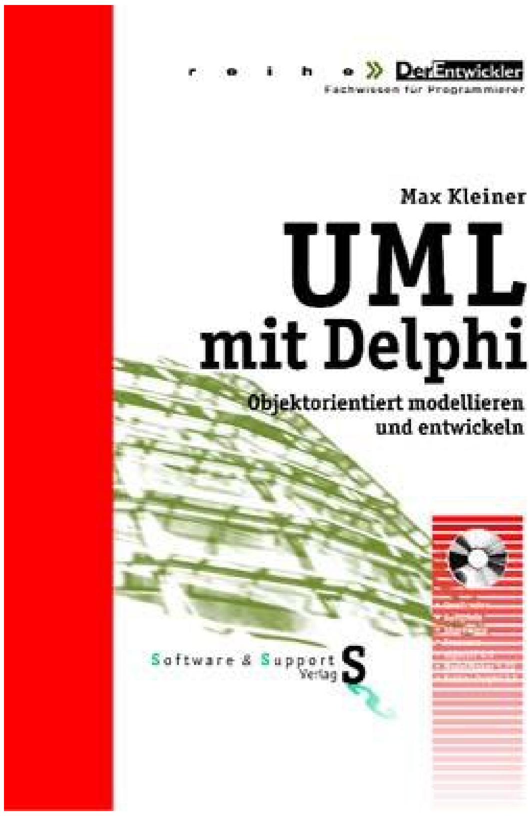
Image to Text API Playground
GET /url
url:
http://www.kleiner.ch/kleiner/images/uml_buch.jpg
REQUIRED:
Javascript
Python
PHP
Go
Java
Ruby
cUrl
C#
Swift
C
Displaying code for: GET /url
Run Code

```
import requests

url = "https://api.apilayer.com/image_to_text/url?url=http%3A%2F%2Fwww.kleiner.ch%2Fkleiner%2Fimages%2Fuml_buch.jpg"

payload = {}
headers= {
  "apikey": "DNwCF9Rf6y1AmSSednjn8ZhAxYXrzco3"
}
```

```
"all_text": "ih \u00bb Der Entwickler\nFachwissen f\u00fcr Programmierer\nMax Kleiner\nUML\nmit Delphi\nObjekt
"annotations": [
  "ih",
  "\u00bb",
  "Der",
  "Entwickler",
  "Fachwissen",
  "f\u00fcr",
  "Programmierer",
  "Max",
  "Kleiner",
  "UML",
  "mit",
  "Delphi",
```

Pic: 1176_apilayer_livedemo.png

The published result datasets are based on LSTM in combination with a OCR. LSTM stands for Long Short-Term Memory and is a type of Recurrent Neural Network(RNN). Talking about RNN, it is a network that works on the present input by taking into consideration the previous output (feedback) and storing in its memory as memory cells for a short period of time (short-term memory). For example take our book-cover as input:



r e i h e » Der Entwickler
Fachwissen für Programmierer

Max Kleiner
UML
mit Delphi
Objektorientiert modellieren
und entwickeln

Software & Support Verlag

Pic: 1176_uml_buch.jpg

LSTMs have feedback connections and cells which make them different to more traditional feed-forward neural networks with the still existing vanishing gradient problem. This property enables LSTMs to process entire sequences of data (e.g. time series, handwriting or sentences) without treating each point in the sequence independently, but rather, retaining useful information about previous data in the sequence like "Objektorientiert", "modellieren", "und", "entwickeln" as a context. The output of the call

```
writeln(Image_to_text_API2(URL_APILAY, URLIMAGEPATH4,
                              'DNwCF9Rf6y1AmSSednjn8ZhAxYXr----'));
```

is the JSON datastring in about Runtime: 0:0:3.859:

{"lang": "de", "all_text": "ih \u00bb Der Entwickler\nFachwissen f\u00fcr Programmierer\nMax Kleiner\nUML\nmit Delphi\nObjektorientiert modellieren\nund entwickeln\nSoftware & Support", "annotations": ["ih", "\u00bb", "Der", "Entwickler", "Fachwissen", "f\u00fcr", "Programmierer", "Max", "Kleiner", "UML", "mit", "Delphi", "Objektorientiert", "modellieren", "und", "entwickeln", "Software", "&", "Support"]}

Also the well known Tesseract 4.0 (like OmniPage) added a new OCR engine based on LSTM neural networks.

The API can also be triggered with this few lines of P4D code:

```
procedure PyCode(imgpath: string);
begin
  with TPythonEngine.Create(Nil) do begin
  pythonhome:= 'C:\Users\max\AppData\Local\Programs\Python\Python36-32\';
  try
    loadDLL;
    ExecString('import requests');
    ExecStr('url= "https://api.apilayer.com/image_to_text/url?
                                                      url='+imgpath+'"');
    ExecStr('payload = {}');
    ExecStr('headers= {"apikey": "dy5L70eQx72794XBZ8sewEgYTZR85----"}');
    Println(EvalStr('requests.request("GET",url, headers=headers,
                                                   data=payload).text'));
  except
    raiseError;
  finally
    unloadDLL;
    free;
  end;
 end;
end;
```

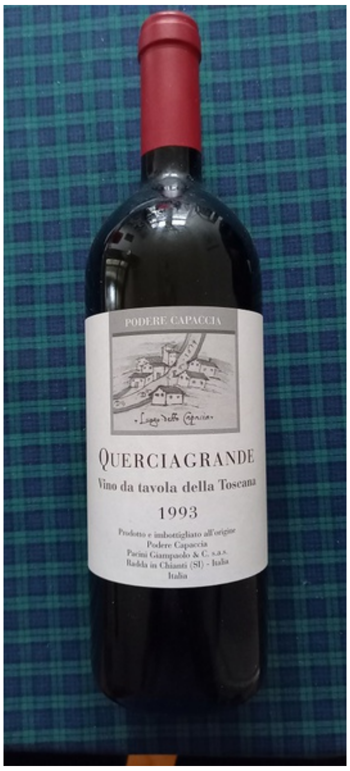
When you fail with a restricted call or an invalid key you get a bunch of exceptions like the following:

wininet_error: Unauthorized (401). or {"message":"Invalid authentication credentials"}

The fact that error code is not "one of the expected return values" tells for the versions that the error comes from an underlying layer and this API just passes it up on internal failure.
To shine a bit more light on those errors a function exists to convert the *ErrorCode* to a string for better understanding:

```
function GetWinInetError(ErrorCode:Cardinal): string;
const
   winetdll = 'wininet.dll';
var
  Len: Integer;
  Buffer: PChar;
begin
  Len:= FormatMessage(
  FORMAT_MESSAGE_FROM_HMODULE or FORMAT_MESSAGE_FROM_SYSTEM or
  FORMAT_MESSAGE_ALLOCATE_BUFFER or FORMAT_MESSAGE_IGNORE_INSERTS or
  FORMAT_MESSAGE_ARGUMENT_ARRAY,
  Pointer(GetModuleHandle(winetdll)), ErrorCode,0, @Buffer,SizeOf(Buffer),nil);
  try
    while (Len > 0) and {$IFDEF UNICODE}(CharInSet(Buffer[Len - 1],[#0..#32,
        '.'])) {$ELSE}(Buffer[Len- 1] in [#0..#32, '.']) {$ENDIF} do Dec(Len);
    SetString(Result, Buffer, Len);
  finally
    LocalFree(HLOCAL(Buffer));
  end;
end;
```

Our final example is an interesting one. What about an old wine bottle (1993) with shapes (and grapes :-)), an old historic painting from 1619, dates, symbols and position angles_:

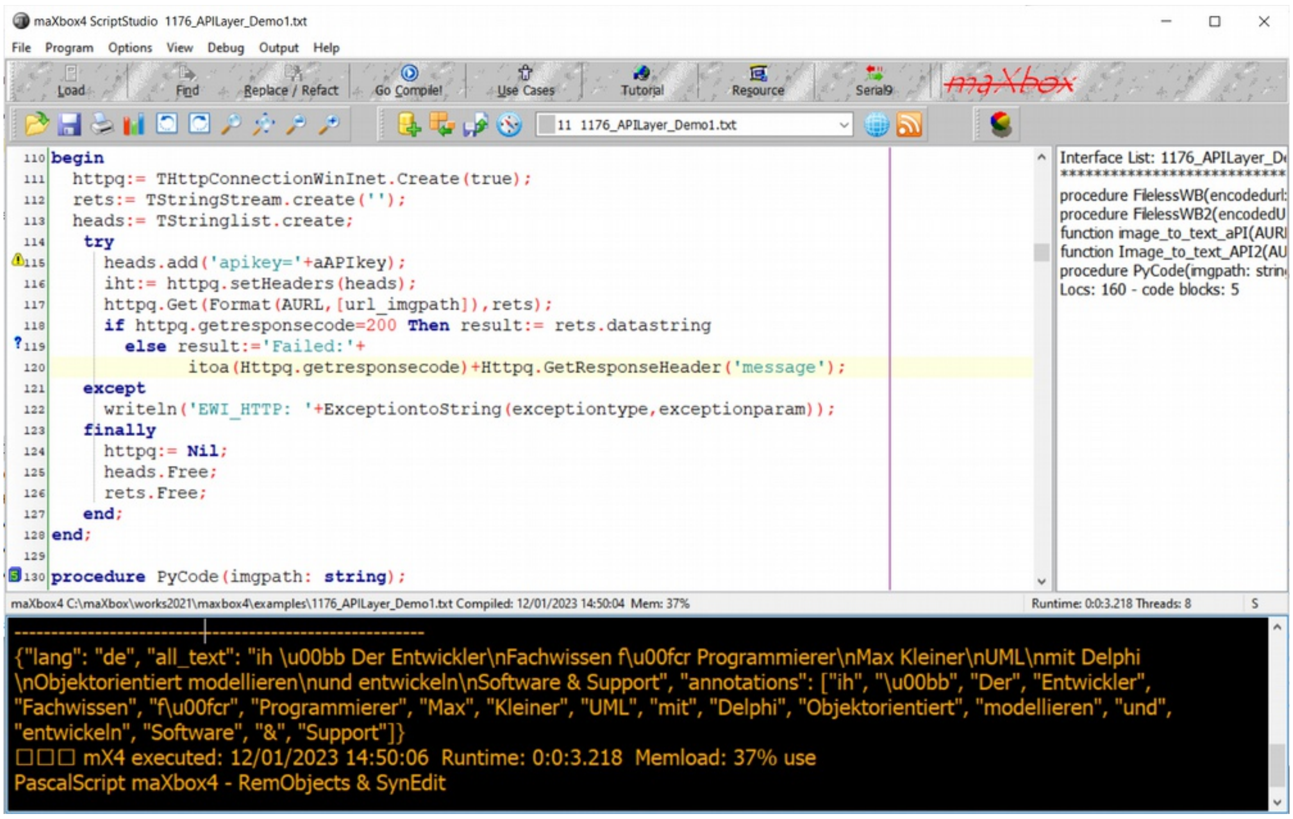


pic: 1176_wine_test.jpg
https://my6code.files.wordpress.com/2022/12/wine_1993_20221230_141947.jpg?w=768

And the result is convincing, also the fact that the year in the label image was recognized correctly as 1619:

{"lang": "it", "all_text": "DS\nPODERE CAPACCIA\n1619\nLuggo deffo apacia\nQUERCIAGRANDE\nVino da tavola della Toscana\n1993\nProdotto e imbottigliato all'origine\nPodere Capaccia\nPacini Giampaolo & C. s.a.s.\nRadda in Chianti (SI) - Italia\nItalia", "annotations": ["DS", "PODERE", "CAPACCIA", "1619", "Luggo", "deffo", "apacia", "QUERCIAGRANDE", "Vino", "da", "tavola", "della", "Toscana", "1993", "Prodotto", "e", "imbottigliato", "all'origine", "Podere", "Capaccia", "Pacini", "Giampaolo", "&", "C.", "s.a.s.", "Radda", "in", "Chianti", "(", "SI", ")", "-", "Italia", "Italia"]}

Any missing or incomplete data is difficult to find without knowing the original. But on the other side, it is very easy to use since users just need to screenshot the part they wish to convert and then copy the text after.
Furthermore the API access is provided in a REST-like interface (Representational State Transfer) exposing database resources or pre-trained models in a JSON format with content-type in the Response Header.
Note: If a programming language is not listed in the Code Example from the live demo, you can still make API calls by using a HTTP request library written in our programming language, as we did with GET or POST.


maXbox4 ScriptStudio 1176_APILayer_Demo1.txt
File Program Options View Debug Output Help
Load
Find
Replace / Refact
Go Compile!
Use Cases
Tutorial
Resource
Serial9
11 1176_APILayer_Demo1.txt

```
begin
  httpq:= THttpConnectionWinInet.Create(true);
  rets:= TStringStream.create('');
  heads:= TStringlist.create;
   try
     heads.add('apikey='+aAPIkey);
     iht:= httpq.setHeaders(heads);
     httpq.Get(Format(AURL,[url_imgpath]),rets);
     if httpq.getresponsecode=200 Then result:= rets.datastring
       else result:='Failed:'+
             itoa(Httpq.getresponsecode)+Httpq.GetResponseHeader('message');
   except
     writeln('EWI_HTTP: '+ExceptiontoString(exceptiontype,exceptionparam));
   finally
     httpq:= Nil;
     heads.Free;
     rets.Free;
   end;
end;

procedure PyCode(imgpath: string);
```

```
Interface List: 1176_APILayer_De
****************************
procedure FilelessWB(encodedurl:
procedure FilelessWB2(encodedU
function image_to_text_aPI(AURI
function Image_to_text_API2(AU
procedure PyCode(imgpath: strin
Locs: 160 - code blocks: 5
```

```
maXbox4 C:\maXbox\works2021\maxbox4\examples\1176_APILayer_Demo1.txt Compiled: 12/01/2023 14:50:04 Mem: 37%     Runtime: 0:0:3.218 Threads: 8     S
```

```
------------------------------------------------------------
{"lang": "de", "all_text": "ih \u00bb Der Entwickler\nFachwissen f\u00fcr Programmierer\nMax Kleiner\nUML\nmit Delphi
\nObjektorientiert modellieren\nund entwickeln\nSoftware & Support", "annotations": ["ih", "\u00bb", "Der", "Entwickler",
"Fachwissen", "f\u00fcr", "Programmierer", "Max", "Kleiner", "UML", "mit", "Delphi", "Objektorientiert", "modellieren", "und",
"entwickeln", "Software", "&", "Support"]}
□□□ mX4 executed: 12/01/2023 14:50:06  Runtime: 0:0:3.218  Memload: 37% use
PascalScript maXbox4 - RemObjects & SynEdit
```

Pic: 1176_apilayer_demo_mX4.png

Conclusion:
The Image to Text API from APILayer detects and extracts text from images using state-of-the-art optical character recognition (OCR) algorithms in combination with a neural network called LSTM. It can detect texts of different sizes, fonts, and even handwriting or difficult numbers.

Reference:

https://apilayer.com/marketplace/image_to_text-api
https://apilayer.com/docs

https://my6.code.blog/2022/09/02/webpostdata/
http://www.kleiner.ch/kleiner/images/uml_buch.jpg

Doc and Tool: https://maxbox4.wordpress.com

Script Ref: 1176_APILayer_Demo1.txt

Appendix: A Delphi REST client API to consume REST services written in any programming language with a class from maXbox4 integration:
https://github.com/fabriciocolombo/delphi-rest-client-api/blob/master/src/HttpConnectionWinInet.pas

The API it is designed to work with Delphi 7 or later; newer versions takes advantage of Generics Methods.
https://github.com/fabriciocolombo/delphi-rest-client-api

Max Kleiner 12/01/2023

maXbox4exe
Digital unterschrieben von maXbox4exe
Datum: 2023.01.13 08:49:21 +01'00'