The Mystery of IStream

Max Kleiner

3 min read · 6 days ago

For clarity and safety, the IStream object defined in the Delphi Visual Component Library (VCL) is a dynamic object that doesn't have any methods, procedures, functions, events etc… until these methods are defined in your code. The IStream Object will publish these methods correctly for external COM objects expecting an IStream interface.

For example you want to convert an OleVariant from an external COM to a memory stream, its kind of a mystery stream:

```
function getmemstreamfromIStream(avariant: variant): TMemorystream;
 var instream: IStream; astream: TStream;
 begin
   instream:= IUnknown(avariant) as IStream;
   astream:= TOleStream.Create(instream);
   result:= astream as TMemorystream;
 end;
```

That will not work because its an invalid class typecast exception, the solution will be a copy from:

```
function getMemStreamfromIStream2(avariant: variant): TMemorystream;
 var instream: IStream; ostream: TStream;
 begin
    instream:= IUnknown(avariant) as IStream;
    ostream:= TOleStream.Create(instream);
    result:= TMemorystream.Create;
    try
      result.CopyFrom(OStream, OStream.Size);
    finally
      OStream.Free;
    end;
 end;
```

You could use most any TStream derived object within your own code implementation, including THandleStream or TFileStream. THandleStream provides access to the file handle variable that Windows API require for many core file read/write API calls.

That code works because of a new feature introduced in D2010, namely the ability to recover a reference to the object that implements an interface. Note though that if the `IStream` is implemented by something other than your Delphi code, then the cast will fail.

As another solution is the load of an IStream from an OLE response stream as unknown variant type to a well known TMemoryStream in order to save the response stream to a file (in our example a binary QR-code image file as a png graphic):

```
Const
  URLGoogleQRCODE='https://chart.apis.google.com/chart?chs=%dx%d&cht=qr&chld=%s
  AFILENAME= 'mX5QRCode5.png';
  QDATA= 'https://maxbox4.wordpress.com/';

Type TQrImage_ErrCorrLevel=(L,M,Q,H);

Function QRCcodeOle(Wid,Hei:Word; C_Level,apath:string; const Data:string): str
var
  httpReq,hr: Olevariant;  instream: IStream;
  jo: TJSON; strm :TMemoryStream;
begin
  httpReq:= CreateOleObject('WinHttp.WinHttpRequest.5.1');
  //jo:= TJSON.Create();
  hr:= httpReq.Open('GET',
                     format(URLGoogleQRCODE,[Wid,Hei,C_Level,HTTPEncode(Data)
  httpReq.setRequestheader('content-type','application/octet-stream');
  //httpReq.setRequestheader('Authorization','Bearer '+ CHATGPT_APIKEY2);
  if hr= S_OK then HttpReq.Send();
  strm:= TMemoryStream.create;
  If HttpReq.Status = 200 Then begin
    try
```

```
          strm:= getMemStreamfromIStream2(HttpReq.responsestream);
          //getmemStreamfromIStream2file(hrstream, apath);
          writeln('responsestream size: '+itoa(strm.size));
          strm.savetoFile(apath)
          openFile(apath);
       except
          writeln('EHTTPex: '+ExceptiontoString(exceptiontype, exceptionparam));
       finally
          strm.free;
          httpreq:= unassigned;
       end;
     end;
   end;
   //https://stackoverflow.com/questions/15441014/how-do-i-load-an-istream-into-a-
```

And the call of the function, but it doesn't returns anything for the moment, cause we store the result direct in a file (better would be to get back a boolean of success or fail):

```
   //6. Call of the OLE WinHttp Class
   writeln('back of OLE call: '+
                  QRCcodeOle(150,150,'Q',ExePath+'\examples\'+AFILENAME, QDATA));
   writeln('SHA1 '+Sha1(ExePath+'examples\'+AFILENAME));  //}
```

Another option is to write a `TStream`-derived class that accesses the `IStream` internally (similar to how the RTL's `TStreamAdapter` class wraps a `TStream` so it can be passed around as an `IStream`).

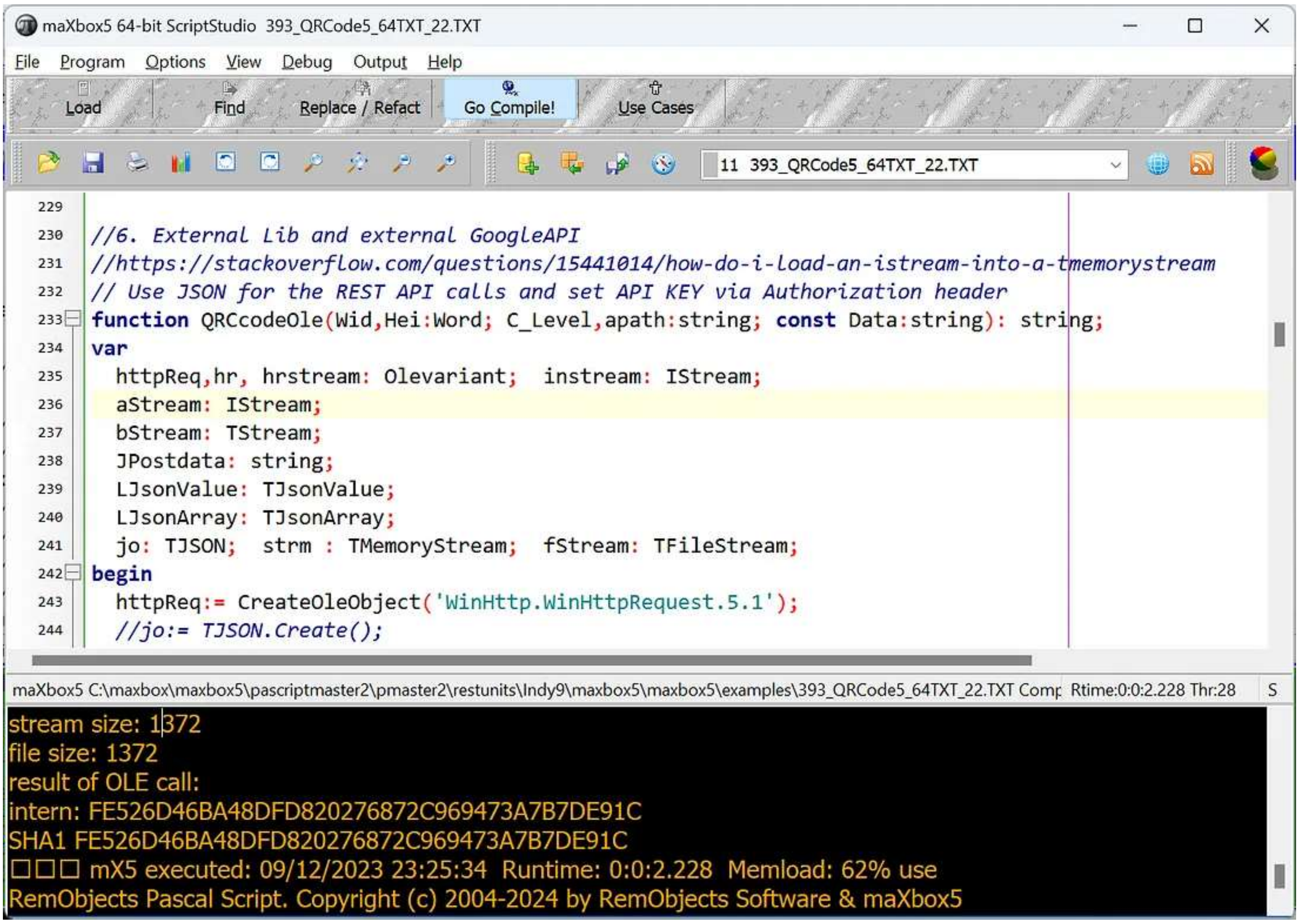
Using the Google Chart Tools / Image Charts (aka Chart API) you can easily generate QR codes, this kind **of** images are a special **type of** two-dimensional barcodes. They are also known **as** hardlinks **or** physical world hyperlinks.

QDATA= ‘https://maxbox4.wordpress.com/';

I was probably not aware of `TOleStream` at the time I wrote this answer. Looking at `TOleStream` now, I notice that it does not support 64-bit streams. This code does. Other than that, this code is almost identical to the code that `TOleStream` uses, with one only exception being that this code's implementation of the `Size` property getter is more optimized than `TOleStream`'s implementation is, and this code implements the `Size` property setter whereas `TOleStream` does not.

So we can combine the invoke call from HttpReq.responsestream to get a file in one function:

```
function getmemStreamfromIStream2File(avariant: variant;
                                          apath: string): Tmemorystream;
 var instream: IStream; ostream: TStream;
 begin
    instream:= IUnknown(avariant) as IStream;
    ostream:= TOleStream.Create(instream);
    result:=  Tmemorystream.Create;
    try
      result.CopyFrom(OStream, OStream.Size);
      result.SaveToFile(apath)
    finally
      OStream.Free;
    end;
 end;
```


maXbox5 64-bit ScriptStudio 393_QRCode5_64TXT_22.TXT
File Program Options View Debug Output Help
Load
Find
Replace / Refact
Go Compile!
Use Cases
11 393_QRCode5_64TXT_22.TXT

```
//6. External Lib and external GoogleAPI
//https://stackoverflow.com/questions/15441014/how-do-i-load-an-istream-into-a-tmemorystream
// Use JSON for the REST API calls and set API KEY via Authorization header
function QRCcodeOle(Wid,Hei:Word; C_Level,apath:string; const Data:string): string;
var
  httpReq,hr, hrstream: Olevariant;  instream: IStream;
  aStream: IStream;
  bStream: TStream;
  JPostdata: string;
  LJsonValue: TJsonValue;
  LJsonArray: TJsonArray;
  jo: TJSON;  strm : TMemoryStream;  fStream: TFileStream;
begin
  httpReq:= CreateOleObject('WinHttp.WinHttpRequest.5.1');
  //jo:= TJSON.Create();
```

maXbox5 C:\maxbox\maxbox5\pascriptmaster2\pmaster2\restunits\Indy9\maxbox5\maxbox5\examples\393_QRCode5_64TXT_22.TXT Comp Rtime:0:0:2.228 Thr:28 S

```
stream size: 1372
file size: 1372
result of OLE call:
intern: FE526D46BA48DFD820276872C969473A7B7DE91C
SHA1 FE526D46BA48DFD820276872C969473A7B7DE91C
□□□ mX5 executed: 09/12/2023 23:25:34  Runtime: 0:0:2.228  Memload: 62% use
RemObjects Pascal Script. Copyright (c) 2004-2024 by RemObjects Software & maXbox5
```

maXbox5 QRCode Tutorial

Get Images as Response Stream

Conclusion: There are probably better ways, but I would create a TOleStream, which is designed as an IStream wrapper; then you can use the CopyFrom() method of your MemoryStream to load and save the data!

Qr Code Streaming API

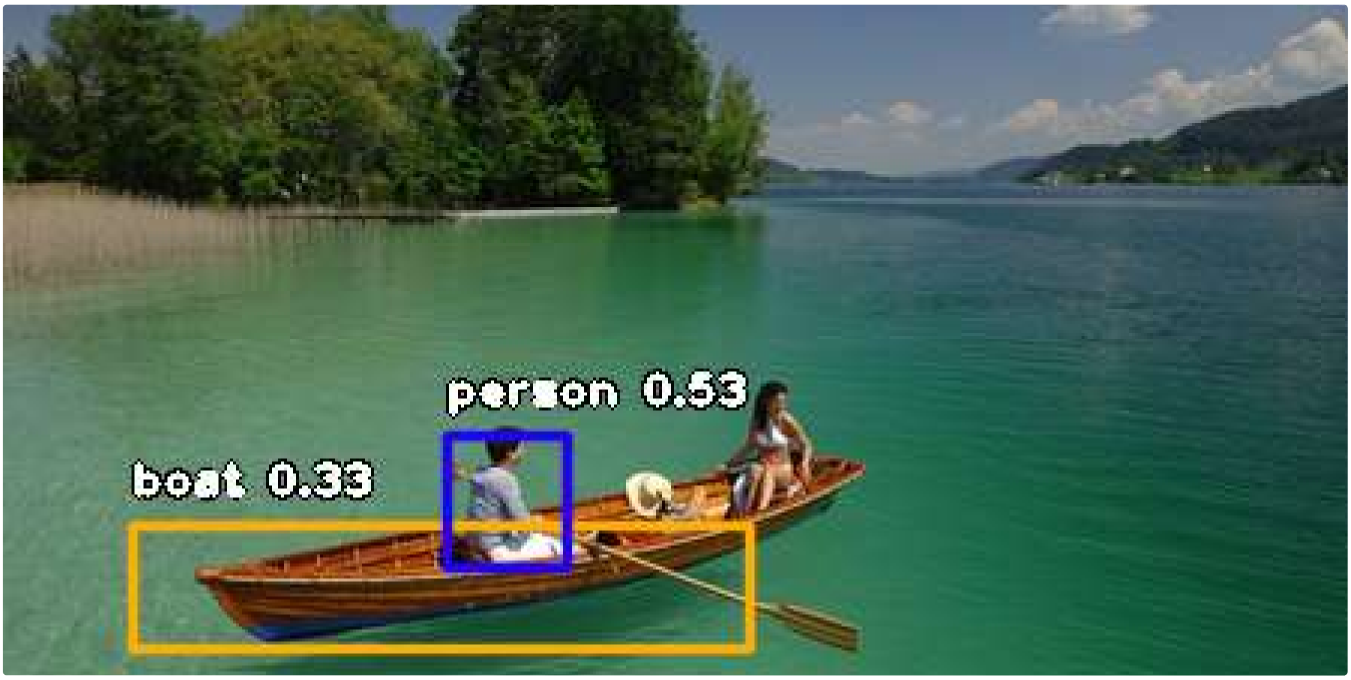
Edit profile

Written by Max Kleiner

20 Followers

Max Kleiner's professional environment is in the areas of OOP, UML and coding - among other things as a trainer, developer and consultant.

More from Max Kleiner



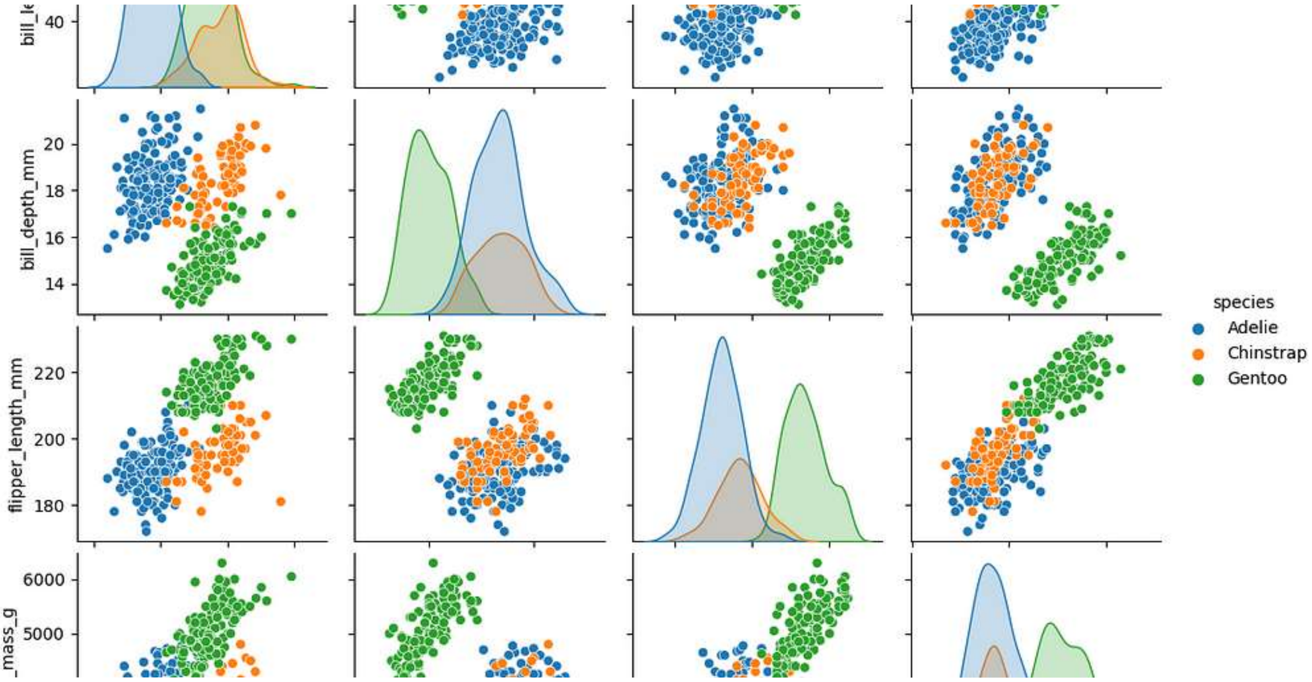
Max Kleiner

Object Detection

//

6 min read · Jul 31, 2020

16



Max Kleiner in Nerd For Tech

Antarctica Learning

A Machine Learning Lesson

6 min read · Nov 22, 2022

16

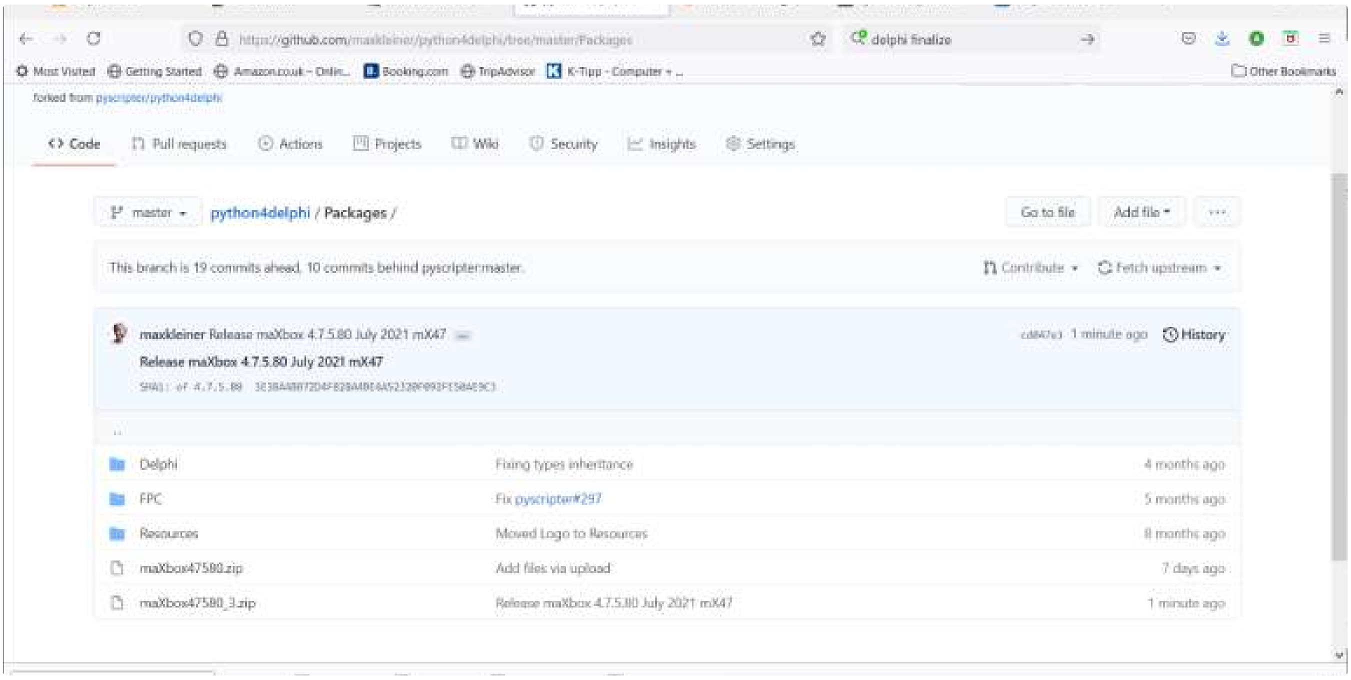
Max Kleiner

Tutorials Overview 2014–2022

In this chapter I want to describe all my 92 tutorials in coding examples:

13 min read · Jan 31, 2022

4



https://github.com/maxkleiner/python4delphi/tree/master/Packages

delphi finalize

forked from pyscripter/python4delphi

python4delphi / Packages /

This branch is 19 commits ahead, 10 commits behind pyscripter:master.

maxkleiner Release maXbox 4.7.5.80 July 2021 mX47 … cd847e3 1 minute ago History

Release maXbox 4.7.5.80 July 2021 mX47

SHA1: of 4.7.5.80 3E38A4B872D4F828A4BE6A52328F093FE5B4E9C3

..

Delphi	Fixing types inheritance	4 months ago
FPC	Fix pyscripter#297	5 months ago
Resources	Moved Logo to Resources	8 months ago
maXbox47580.zip	Add files via upload	7 days ago
maXbox47580_3.zip	Release maXbox 4.7.5.80 July 2021 mX47	1 minute ago

Max Kleiner

Python4maXbox Code

maXbox4exe Digitally signed by maXbox4exe Date: 2023.12.16 10:04:20 +01'00'