****

maXbox Starter 94

****

Work **with** Post API Services

Max Kleiner

//Zwei Worte werden Dir im Leben viele Türen öffnen - "ziehen" und "stossen".
Two words will open many doors **for** you **in** life - "pull" **and** "push".

https://github.com/LibreTranslate/LibreTranslate#mirrors

The Question **is**: What **is** the easiest way **to do** an HTTPS POST request **in** Delphi? Im **not** having problems **with** making HTTP POST requests, but how can I **do it using** SSL **with** Request Headers? Ive searched around **and** havent found anything clear that explains this well enough.

Essentially, a POST **or** GET API microservice architecture **is** a method **of** developing software applications **as** a suite **of** independently deployable, small, modular services **or** building blocks **in** which each service runs a unique process **and** communicates through a well-defined, lightweight mechanism **to** serve a business goal.

Such a microservice can be
- a socket server
- a data logger
- signal sensor detector
- language translator
- sentiment analysis

So the short answer **is** simple, use a COM-**Object with** flexible late binding:

```
function getPostTranslateLibre(feedstream: string): string;
var
  Url,API_KEY, source: string;
  jo, locate: TJSONObject;
  httpReq,hr: Olevariant;
  strm: TStringStream;
begin
  httpReq:= CreateOleObject('WinHttp.WinHttpRequest.5.1');
  // Open the HTTPs connection.
  try
    //hr:= httpReq.Open('POST','https://libretranslate.com/detect', false);
    hr:= httpReq.Open('POST','https://libretranslate.pussthecat.org/detect', false);
    httpReq.setRequestheader('user-agent',
          'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0');
    httpReq.setRequestheader('content-type','application/x-www-form-urlencoded');
    //httpReq.setRequestheader('X-RapidAPI-Host','nlp-translation.p.rapidapi.com');

    //httpReq.setRequestheader('X-RapidAPI-Key','...333');

    if hr= S_OK then HttpReq.Send('q='+HTTPEncode(feedstream));
     /// Send HTTP Post Request & get Responses.

    If HttpReq.Status = 200 Then
       result:= HttpReq.responseText
    Else result:= 'Failed at getting response:'+itoa(HttpReq.Status)+HttpReq.responseText;
     //writeln('debug response '+HttpReq.GetAllResponseHeaders);
  finally
    httpreq:= unassigned;
  end;
end;
```

This Post API **of** the example language detector **is** free **for** the moment **and** shows the proot **of** concept. Free **and** Open Source Machine Translation API, entirely self-hosted. Unlike other APIs, **it** doesnt rely **on** proprietary providers such **as** Google **or** Azure **to** perform translations. Instead, its translation engine **is** powered by the open source Argos Translate **library**.

LibreTranslate supports per-user limit quotas, e.g. you can issue API keys to users so that they can enjoy higher requests limits per minute (if you also set --req-limit). By default all users are rate-limited based on --req-limit, but passing an optional api_key parameter to the REST endpoints allows a user to enjoy higher request limits.

Then we add some business goal to the service:

- a socket server as a time and temp server
- a data logger to store climate samples
- signal sensor to get temperature and others
- a language translator/detector to fullfill a sentiment analysis

The idea behind POST microservices is that some types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.

First example is the main of a http-server:

```
 begin   //@main
  //TWebServerCreate;
  with TIdHTTPServer.Create(Nil) do begin
    sr:= GetIPfromHost(getHostName)  //'172.16.10.80';
    Bindings.Add.IP:= sr;
    Bindings.Add.Port:= 8080;
    OnCommandGet:= @HTTPServerCommandGet;
    Active:= True;
    try
      Writeln('Hello world/Web server start at: '+sr);
      ShowMessageBig('maXbox Hello WorldWeb server at: '+sr+#1310+
                         '  Press OK to quit webserver!'+#13);
    finally
      writeln('SocketServer stop: '+timetoStr(now)); //TWebServerDestroy;
      Active:= False;
      Free;
    end;
  end;
```

Another answer is the use of a compiled early binding library, for example the ALHttpClient Base Class. TALHttpClient is a ancestor of class like TALWinInetHttpClient or TALWinHttpClient:

http://sourceforge.net/projects/alcinoe/

```
function TALHTTPClient_Post5HTTPSTranslate(aUrl: AnsiString;
                             aPoststring: string;
                             aResponseContentStream: TStringStream;
                             aResponseContentHeader: TALHTTPResponseHeader): string;
Var OldContentLengthValue: AnsiString;
   LHttpClient: TALWininetHttpClient;
   FRequestHeader: TALHTTPRequestHeader;
   aPostDataStrings: TALStrings; aPostDataStream: TStream;
begin
  LHttpClient:= TALWininetHttpClient.create;
  LHttpClient.Url:= aUrl;
  LHttpClient.RequestMethod:= HTTPmt_Post; //HTTPrm_Post;
  LHttpClient.RequestHeader.UserAgent:=USERAGENT;
  LHttpClient.RequestHeader.ContentType:='application/x-www-form-urlencoded';
  //LHttpClient.RequestHeader.CustomHeaders:=
  //LHttpClient.RequestHeader.RawHeaderText:=
  //           'content-type: application/x-www-form-urlencoded'; //+CRLF+
             //'X-RapidAPI-Host: nlp-translation.p.rapidapi.com'+CRLF+
             //'X-RapidAPI-Key: "df61a35825msh..."';
  try
     aPostDataStrings:= TALStringlist.create;
     aPostDataStrings.add('q='+HTTPEncode(apoststring));
     writeln('postman '+aPostDataStrings.strings[0]+' '+apoststring)
```

```
    try
      result:= LHttpClient.PostUrlEncoded(aUrl, aPostDataStrings, true); //overload;
      aPostDataStream:= TStringStream.create('');
      aResponseContentStream:= TStringStream.create('');
      //result:= aResponseContentHeader.ReasonPhrase;
    except
      writeln('E: '+ExceptiontoString(exceptiontype, exceptionparam));
      //writeln('E: '+aResponseContentHeader.ReasonPhrase+'-'+aResponseContentHeader.Rawheadertext+
         //          '--- '+aResponseContentStream.datastring);
    end;
  finally
    LHttpClient.Free;
    aPostDataStrings.Free;
    aPostDataStream.Free;
    aResponseContentStream.Free;
  end;
end;
```

You can configure user-agent or content-type with type safety and debug possibilities.

To analyze the sentiment of some text for example, do an HTTP POST to http://text-processing.com/api/sentiment/ with form encoded data containg the text you want to analyze.

You’ll get back a JSON object response with 2 attributes:

label: will be either pos if the text is determined to be positive, neg if the text is negative, or neutral if the text is neither pos nor neg.

probability: an object that contains the probability for each label. neg and pos will add up to 1, while neutral is standalone. If neutral is greater than 0.5 then the label will be neutral. Otherwise, the label will be pos or neg, whichever has the greater probability.

```
BBC-News Sentiment of 2022-04-12 20:01:52.917443
0: Donbas: Battle in east Ukraine expected to be bloody and decisive: -0.7906
1: Brooklyn shooting: Sixteen injured in New York City subway station: -0.9001
2: Johnny Depp and Amber Heard: Heard giving 'performance of her life': 0.34
3: Marine Le Pen says she opposes sanctions on Russian gas: 0.6124
4: Sepp Blatter and Michel Platini to go on trial in June to face corruption charges: -0.2732
5: Ukraine war: Putin says Russian invasion will achieve 'noble' aims: -0.296
6: Kinahan Cartel: US sanctions cartel leader with links to Tyson Fury: -0.4404
7: Ukraine War: US 'deeply concerned' at report of Mariupol chemical attack: -0.5994
8: Ukraine war: Desperate mother writes details on toddler's back: -0.8934
9: Britney Spears says she is pregnant after conservatorship ends: -0.7845
10: El Salvador: Whip-wielding demons kick off Easter week: -0.1027
11: Ukraine: Our parents wouldn’t leave Bucha, then Russia moved in: -0.4063
12: Grieving Russians can't believe talk of war crimes in Ukraine: -0.926
13: Ukraine conflict: 'Russian soldiers raped me and killed my husband': -0.9705
14: Hidden wealth of one of Putin’s 'inner circle' revealed: 0.4939
15: Ukraine round-up: Austria pessimistic after Putin talks: -0.4404
16: Zelensky asked if he'll give Russia any part of Ukraine: 0.3378
17: Could Marine Le Pen win the French elections?: 0.5859
18: Falklands War: 'The UK is still usurping our land': -0.7506
19: Ukraine war: The foreign fighters supporting the Ukrainian army: -0.4939
20: Spanish police seize huge haul of illegal stuffed animals: -0.3182
2022-04-12 18:01:52.917443
```

Source of the script at:
http://www.softwareschule.ch/examples/sentiment4.txt

Ref: https://github.com/frantic/delphi-tdd-example/blob/master/src/RssModel.pas
     http://text-processing.com/docs/sentiment.html
     https://stackoverflow.com/questions/3885703/post-method-winhttprequest-multipart-form-data

Every day we interact with many websites during web browsing. To get any web resource using a web browser we generally fire a HTTP request to the server.

As developers, we should know what we are sending to the server from our browser using a HTTP request and what we are getting from the server as the HTTP response.
In the first code snippet we get the header with

```
  writeln('debug response '+HttpReq.GetAllResponseHeaders);
```

Our objective here is to capture the following things during HTTP request and HTTP response:

- Request Headers
- Getting Cookie information
- Request body °
- Response headers °
- Response body °

HTTP/HTTPS consists of request-response pairs: the request from your computer to the server and the response from the server or the middleware framework.
For generic sockets the request-response consists of the entire contents of the inbound and outbound streams. This is not always so useful for sockets and may be improved in future. If you need to do a lot of socket level debugging you may want to consider using Ethereal.
Sequence view lets your view the requests in the sequence that they occur:

```
back from langdetext: [{"confidence":98.0,"language":"en"}]
debug response Connection: keep-alive
Date: Thu, 14 Apr 2022 07:19:22 GMT
Content-Length: 38
Content-Type: application/json
Server: nginx
Vary: Accept-Encoding
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Authorization, Content-Type
Access-Control-Allow-Methods: GET, POST
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Authorization
Access-Control-Max-Age: 1728000
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer
Content-Security-Policy: default-src 'self' http: https: data: blob: 'unsafe-inline'; frame-ancestors 'self';
Permissions-Policy: interest-cohort=()
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload


back from langdetext in bad case: [{"confidence":98.0,"language":"es"}]

E: Exception: BAD REQUEST (400) - 'https://libretranslate.pussthecat.org/detect'.
E: BAD REQUEST- HTTP/1.1 400 BAD REQUEST
Server: nginx
Date: Thu, 14 Apr 2022 07:19:23 GMT
Content-Type: application/json
Content-Length: 49
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Authorization, Content-Type
Access-Control-Allow-Methods: GET, POST
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Authorization
Access-Control-Max-Age: 1728000
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer
Content-Security-Policy: default-src 'self' http: https: data: blob: 'unsafe-inline'; frame-ancestors 'self';
Permissions-Policy: interest-cohort=()
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

With Event-handlers or a delegate you do have the flexibility to act as a microservice. The OnCommandGet() event can be changed with a lot of use cases at design or at runtime as well.

In this example like in Object Pascal or C#, you can think of a delegate as a pointer (or a reference) to a method. This is useful because the pointer can be passed around as a value like in above case @HTTPServerCommandGet;:

```
 procedure HTTPServerCommandGet(AContext: TIdPeerThread;
              ARequestInfo: TIdHTTPRequestInfo; ARespInfo: TIdHTTPResponseInfo);
  begin
    ARespInfo.ResponseNo:= 200;
    ARespInfo.ContentType:= 'text/plain';
    ARespInfo.ContentText:= 'Hi IBZ 2022 TimeServe at: '
                                  +DateTimeToInternetStr(Now,true);
  end;
```

The central concept of a delegate is its signature, or shape:

```
 HTTPServerCommandGet(AContext: TIdPeerThread;
              ARequestInfo: TIdHTTPRequestInfo; ARespInfo: TIdHTTPResponseInfo);
```

To do this, we create specific methods for the code we want to be executed. The glue between the event and the methods (event handlers) to be executed are the delegates.

The common definition of microservices generally relies upon each microservice providing an API endpoint, often but not always a stateless REST API which can be accessed over HTTP(S) just like a standard webpage. This method for accessing microservices make them easy for developers to consume as they only require tools and methods many developers are already familiar with.

This is how get get the TMP36 sensor value from Arduino:

```
 function connectAndGetValue: string;
 begin
   with TBlockSerial.Create do begin
     Config(9600,8,'N',1,true,false);
     Connect(COMPORT);
     result:= RecvString(1800)   //com timeout
     CloseSocket;
     Free;
   end;
 end;
```

Then the result is pushed to a web socket in a timer mode with another delegate:

```
   arTimer:= TTimer.Create(Self);
   arTimer.Enabled:= true;
   arTimer.Interval:= 2000;
   arTimer.OnTimer:= @eventActTimer;

  procedure eventActTimer(sender: TObject);
  begin
    tmpval:= connectAndGetValue;
    writeln(datetimetostr(now)+' C°: '+tmpval+'° >'+aremoteIP)
    aremoteIP:= '';
  end;
```

Be aware of the remoteIP:

```
 Exception: Could not bind socket. Address and port are already in use.
   PrintF('Command %s received: %s of temperature C°: %s',
           [RequestInfo.Command,thread.connection.Socket.binding.PeerIP,tmp2]);
```

This is a common newbie mistake. You are creating two bindings, one bound to 127.0.0.1:DefaultPort, and one bound to 0.0.0.0:50001. You need one binding instead, that is bound to 127.0.0.1:50001 instead.

```
with HTTPServer1.Bindings.Add do begin
  IP:= '127.0.0.1';
  Port:= 50001;
end;
```

In its simplest forms, we can call now the service from a browser or a desktop app like a web- or win form. At least the client call:

```
procedure TDataFormbtnHTTPSendGetClick(Sender: TObject);
  var
    HTTPClient: TIdHTTP;
    responseStream: TMemoryStream;
  begin
    HTTPClient:= TIdHTTP.Create(Nil);
    responseStream:= TMemoryStream.Create;
    try
      try
        HTTPClient.Get1('http://127.0.0.1:8080',responseStream);
        responseStream.Seek(0, soFromBeginning);
        SetLength(Sr, responseStream.Size);
        responseStream.Read(Sr, responseStream.Size);
        writeln('response: '+sr)
      except
        //on e : Exception do begin
        Showmessage('Could not send get request to localhost, port 8080');
      end;
      //end;
    finally
      //@FreeAndNil(HTTPClient);
      HTTPClient.Free;
      HTTPClient:= Nil;
      responseStream.Free;
    end;
  end;
```

Or take another old concept from cryptography RSA. Encode and decode can bee seen as two micro services with different use cases:
We have public and private keys, each including of two values.
For the public key the values are n of p*q, the so called "modulus", and E, a well known encrypting integer prime with the value: Const E = 65537;.

The private key values are also n, the same modulus that appears in the public key, and d, a big number which can decrypt any message encrypted using the public key.

There are obviously two cases:

1. Encrypting with public key, and then decrypting with private key.
   For a message or data
2. Encrypting with private key, and then decrypting with public key.
   For a digital signature

Conclusion:
The idea of separating applications into smaller parts is nothing new; there are other programming paradigms which address this same concept, such as Service Oriented Architecture (SOA) or POST-Services. What may be new are some of the tools and techniques used to deliver on the promise of microservices like Docker, OpenStack, Postman, Swagger, RapidAPI or OpenShift.
Simplify API development for users, teams, and enterprises with the Swagger open source and professional toolset. Find out how Swagger can help you design and document your APIs at scale.

Each service should be independently developed and deployed. No coordination should be needed with other service teams if no breaking API changes have been made. Each service is effectively it's own product with it's own codebase and lifecycle.

```
                    ----
                   /~@@~\,
 _______ . _\_\___/\ __ /\___|_|_ . _______
/ ____  |=|      \  <_+>  /      |=|  ____ \
~|    |\|=|======\\_____//======|=|/|    |~
 |_   |    \      |     |      /    |    |
  \==-|     \     |  mX4 |     /     |----|~~)
  |   |      |    |     |    |      |____/~/
  |   |       \____\____/____/      /    / /
  |   |         {-----------}      /____/ /
  |___|        /~~~~~~~~~~~~~\    |_/~|_|/
   \_/        [/~~~~~||~~~~~\]     /__|\
   | |         |     ||||    |    (/|[[\)
   [_]        |     |  |     |
              |_____|  |_____|
              (_____)  (_____)
              |     |  |     |
              |     |  |     |
              |/~~~\|  |/~~~\|
              /|___|\  /|___|\
             <_______><_______>

 A microservice architecture shifts around complexity. Instead of a single complex system, you have a bunch of simple services with complex interactions.


 Ref: http://www.softwareschule.ch/maxbox.htm

  ..\examples\210_RSA_crypto_complete8hybrid.txt
  ..\examples\750_ibz_cryptomem_RSA_proof_64.txt
  ..\examples\749_helloWebServer3_tempsensor3.txt
  ..\examples\749_helloWebServer3.txt
  ..\examples\sentiment4.txt
  ..\examples\1121_sentiment_api3_bbc_newsfeed4rec21.txt

 Doc:

  https://rapidapi.com/hub

  http://text-processing.com/demo/sentiment/

  https://opensource.com/resources/what-are-microservices

  https://sourceforge.net/projects/alcinoe/

  http://www.vinaysahni.com/best-practices-for-building-a-microservice-architecture

  http://www.softwareschule.ch/download/maxbox_functions.txt

  https://www.academia.edu/31112544/Work_with_microservice_maXbox_starter48.pdf

  There are only 10 types of people: those who understand binary and those do not.

  Appendix:
  TWinApiDownload = class(TObject)
  private
    fEventWorkStart : TEventWorkStart;
    fEventWork : TEventWork;
    fEventWorkEnd : TEventWorkEnd;
    fEventError : TEventError;
    fURL : string;
    fUserAgent : string;
    fStop : Boolean;
    fActive : Boolean;
    fCachingEnabled : Boolean;
    fProgressUpdateInterval : Cardinal;
    function GetIsActive : Boolean;
```

```
  public
    constructor Create;
    destructor Destroy; override;
    function CheckURL(aURL: string) : Integer;
    function Download(Stream : TStream) : Integer; overload;
    function Download(var res : string) : Integer; overload;
    function ErrorCodeToMessageString(aErrorCode : Integer) : string;
    procedure Stop;
    procedure Clear;
    property UserAgent : string read fUserAgent write fUserAgent;
    property URL : string read fURL write fURL;
    property DownloadActive : Boolean read GetIsActive;
    property CachingEnabled : Boolean read fCachingEnabled write fCachingEnabled;
    property UpdateInterval:Cardinal read fProgressUpdateInterval write fProgressUpdateInterval;
    property OnWorkStart : TEventWorkStart read fEventWorkStart write fEventWorkStart;
    property OnWork : TEventWork read fEventWork write fEventWork;
    property OnWorkEnd : TEventWorkEnd read fEventWorkEnd write fEventWorkEnd;
    property OnError : TEventError read fEventError write fEventError;
  end;

C:\maXbox\works2021\maxbox4>py
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import http.client
>>>
>>> conn = http.client.HTTPSConnection("nlp-translation.p.rapidapi.com")
>>> payload = "text=Hello%20World&to=es&from=en"
>>> headers = {
...     'content-type': "application/x-www-form-urlencoded",
...     'X-RapidAPI-Host': "nlp-translation.p.rapidapi.com",
...     'X-RapidAPI-Key': "df61a35825msh66c9514de953a7ap192bcfjsn16a3d1018ce3"
...     }
>>> conn.request("POST", "/v1/translate", payload, headers)
>>>
>>> res = conn.getresponse()
>>> data = res.read()
>>> print(data.decode("utf-8"))
{"message":"You are not subscribed to this API."}
>>>

               _od#HMM6&*MMMH::-_
          _dHMMMR??MMM? ""| `"'-?Hb_
       .~HMMMMMMMMHMMM#M?        `*HMb.
     ./?HMMMMMMMMMMM"*"""           &MHb.
    /'|MMMMMMMMMMM'             -   `*MHM\
   /  |MMM'MMHHM''                   .MMMHb
  |   9HMMP   .Hq,                   TMM'MMH
 /     |MM\,H-""&&6\__               `MMMMMMb
|       `""HH#,       \             - MMMMMMM|
|           `HoodHMM###.              `9MMMMMH
|              .MMMMMMMM##\             `*"?HM
|         ..  ,HMMMMMMMMMMMo\.              |M
|             |MMMMM'MMMMMMM'MNHo           |M
|              ?MMMMMMM'MMMMMMMM*           |H
|.              `#MMMMMMMM'MMMM'           .M|
 \                `MMMMMMMMMMMM*           |P
 `\                MMMMMMMMT"'            ,H
  `\              `MMM'MMH?              ./
    \.            |MMMH#"               ,/
     `\.          |MMP'               ./'
       `~\        `HM:.-    .       ,/'
          "-\_       '_\ .      _.-"
              "-\-#odMM\_,oo==-"

```

```

https://github.com/LibreTranslate/LibreTranslate#mirrors
-----_____
                   _____------           __      ----_
            ___----             ___------              \
               ----________        ----                 \
                           -----__    |             _____)
                                __-                /     \
                    _______-----    ___--          \    /)\
              ------_______      ---____            \__/  /
                           -----__    \ --    _          /\
                                  --__--__     \_____/   \_/\
                                          ----|   /          |
                                              |  |___________|
                                              |  | ((_(_)| )_)
                                              |  \_((_(_)|/(_)
                                              \             (
                                               \_____________)

```