```
{ * 外部api支持，百度翻译服务http支持                                          * }
{ * https://github.com/PassByYou888/CoreCipher                                 * }
{ * https://github.com/PassByYou888/ZServer4D                                   * }
{ * https://github.com/PassByYou888/zExpression                                 * }
{ ****************************************************************************** }
unit BaiduTranslateAPI;

interface

uses Classes, CoreClasses, PascalStrings, UnicodeMixedLib, MemoryStream64, JsonDataObjects;

var
  // 使用下列地址申请百度翻译
  // http://api.fanyi.baidu.com

  // 申请api使用资格是免费的，待申请完成后
  // 在开发者控台首页，会看见appid和密钥，粘贴到下面即可

  // 百度翻译接口可以用于客户端，也可以用于FMX
  // 但是在客户端直接使用翻译容易造成密钥丢失，以及无法记录用户的翻译历史和加速翻译，并且翻译量容易超标，造成不可控制的被扣费

  // 百度的公共测试密钥
  BaiduTranslate_Appid: string = '2015063000000001';
  BaiduTranslate_Key  : string = '12345678';

  // 接口百度翻译的最大安全线程
  // 系统每个线程会占用1-4M(linux X64下是8M)的堆栈空间，合理安排后台资源
  // 线程堆栈的技术摘要请自行参考下列介绍
  // http://blog.csdn.net/cherish_2012/article/details/45073399
  BaiduTranslate_MaxSafeThread: Integer = 100; // 100是豪华配置的linux云主机

type
  // 将来百度翻译api新增语种后，自行在下列添加进去即可
  TTranslateLanguage = (
    tL_auto, // 自动
    tL_zh,   // 中文
    tL_en,   // 英语
    tL_yue,  // 粤语
    tL_wyw,  // 文言文
    tL_jp,   // 日语
    tL_kor,  // 韩语
    tL_fra,  // 法语
    tL_spa,  // 西班牙语
    tL_th,   // 泰语
    tL_ara,  // 阿拉伯语
    tL_ru,   // 俄语
    tL_pt,   // 葡萄牙语
    tL_de,   // 德语
    tL_it,   // 意大利语
    tL_el,   // 希腊语
    tL_nl,   // 荷兰语
    tL_pl,   // 波兰语
    tL_bul,  // 保加利亚语
    tL_est,  // 爱沙尼亚语
    tL_dan,  // 丹麦语
    tL_fin,  // 芬兰语
    tL_cs,   // 捷克语
    tL_rom,  // 罗马尼亚语
    tL_slo,  // 斯洛文尼亚语
    tL_swe,  // 瑞典语
    tL_hu,   // 匈牙利语
    tL_cht,  // 繁体中文
    tL_vie); // 越南语

  TTranslateCompleteProc = reference to procedure(UserData: Pointer; Success: Boolean; sour, dest: TPascalString);

procedure BaiduTranslateWithHTTP(UsedSSL: Boolean;
  sourLanguage, desLanguage: TTranslateLanguage;
  text: TPascalString;
  UserData: Pointer;
  OnResult: TTranslateCompleteProc);

threadvar BaiduTranslateTh: Int64;

implementation

uses SysUtils, IdHTTP;
```

```
type
  THTTPGetTh = class;

  THTTPSyncIntf = class
    th: THTTPGetTh;
    url: string;
    HTTP: TIdHTTP;
    m64: TMemoryStream64;
    UserData: Pointer;
    OnResult: TTranslateCompleteProc;

    procedure AsyncGet;
  end;

  THTTPGetTh = class(TThread)
    syncIntf: THTTPSyncIntf;
    procedure execute; override;
  end;

procedure THTTPSyncIntf.AsyncGet;
var
  js       : TJsonObject; // 高频率翻译这种东西，我们要用JsonDataObjects，它的缺陷是不支持fpc，但是它稳定和快速
  ja       : TJsonArray;
  I        : Integer;
  Success  : Boolean;
  sour, dst: TPascalString;
begin
  try
      HTTP.Get(url, m64);
  except
    th.Synchronize(
      procedure
      begin
        OnResult(UserData, false, '', '');
      end);
    exit;
  end;
  m64.Position := 0;
  js := TJsonObject.Create;
  js.LoadFromStream(m64, TEncoding.UTF8, True);

  Success := false;
  sour := '';
  dst := '';

  try
    // 对翻译结果做安全检查
    if js.IndexOf('trans_result') >= 0 then
      begin
        ja := js.A['trans_result'];
        if ja.Count > 0 then
          begin
            for I := 0 to ja.Count - 1 do
              begin
                Success := True;
                if I = 0 then
                  begin
                    sour := ja[I].S['src'];
                    dst := ja[I].S['dst'];
                  end
                else
                  begin
                    sour.Append(#13#10 + ja[I].S['src']);
                    dst.Append(#13#10 + ja[I].S['dst']);
                  end;
              end;
          end;
      end;
  except
    DisposeObject([js]);
    th.Synchronize(
      procedure
      begin
        OnResult(UserData, false, '', '');
      end);
    exit;
  end;

  th.Synchronize(
```

```
    procedure
    begin
      OnResult(UserData, Success, sour, dst);
    end);

  DisposeObject([js]);

  DisposeObject([m64]);
end;

procedure THTTPGetTh.execute;
begin
  inc(BaiduTranslateTh);
  FreeOnTerminate := True;
  try
      syncIntf.AsyncGet;
  finally
    // 安全回收
    DisposeObject(syncIntf);
    dec(BaiduTranslateTh);
  end;
end;

function TranslateLanguage2Token(t: TTranslateLanguage): TPascalString; inline;
begin
  // 我已备注各个翻译标记
  case t of
    tL_auto: Result := 'auto'; // auto会根据http的请求编码决定翻译语言，其它不会
    tL_zh: Result := 'zh';     // 中文
    tL_en: Result := 'en';     // 英语
    tL_yue: Result := 'yue';   // 粤语
    tL_wyw: Result := 'wyw';   // 文言文
    tL_jp: Result := 'jp';     // 日语
    tL_kor: Result := 'kor';   // 韩语
    tL_fra: Result := 'fra';   // 法语
    tL_spa: Result := 'spa';   // 西班牙语
    tL_th: Result := 'th';     // 泰语
    tL_ara: Result := 'ara';   // 阿拉伯语
    tL_ru: Result := 'ru';     // 俄语
    tL_pt: Result := 'pt';     // 葡萄牙语
    tL_de: Result := 'de';     // 德语
    tL_it: Result := 'it';     // 意大利语
    tL_el: Result := 'el';     // 希腊语
    tL_nl: Result := 'nl';     // 荷兰语
    tL_pl: Result := 'pl';     // 波兰语
    tL_bul: Result := 'bul';   // 保加利亚语
    tL_est: Result := 'est';   // 爱沙尼亚语
    tL_dan: Result := 'dan';   // 丹麦语
    tL_fin: Result := 'fin';   // 芬兰语
    tL_cs: Result := 'cs';     // 捷克语
    tL_rom: Result := 'rom';   // 罗马尼亚语
    tL_slo: Result := 'slo';   // 斯洛文尼亚语
    tL_swe: Result := 'swe';   // 瑞典语
    tL_hu: Result := 'hu';     // 匈牙利语
    tL_cht: Result := 'cht';   // 繁体中文
    tL_vie: Result := 'vie';   // 越南语
    else
      Result := 'auto'; // auto会根据http的请求编码决定翻译语言，其它不会
  end;
end;

procedure BaiduTranslateWithHTTP(UsedSSL: Boolean; sourLanguage, desLanguage: TTranslateLanguage; text: TPascalString; U
serData: Pointer; OnResult: TTranslateCompleteProc);
var
  salt    : Integer;
  httpurl : TPascalString;
  soursign: TPascalString;
  lasturl : TPascalString;
  intf    : THTTPSyncIntf;
  th      : THTTPGetTh;
begin
  salt := umlRandomRange(32767, 1024 * 1024 * 2);
  soursign := BaiduTranslate_Appid + text + intToStr(salt) + BaiduTranslate_Key;

  if UsedSSL then
      httpurl := 'http://api.fanyi.baidu.com/api/trans/vip/translate'
  else
      httpurl := 'https://api.fanyi.baidu.com/api/trans/vip/translate';
```

```
  lasturl.text := httpurl + '?' +
    'q=' + umlURLEncode(text) +
    '&from=' + TranslateLanguage2Token(sourLanguage) +
    '&to=' + TranslateLanguage2Token(desLanguage) +
    '&appid=' + BaiduTranslate_Appid +
    '&salt=' + intToStr(salt) +
    '&sign=' + umlStringMD5Char(soursign);
  intf := THTTPSyncIntf.Create;
  intf.th := THTTPGetTh.Create;
  intf.th.syncIntf := intf;
  intf.url := lasturl;
  intf.HTTP := TIdHTTP.Create(nil);
  intf.m64 := TMemoryStream64.Create;
  intf.UserData := UserData;
  intf.OnResult := OnResult;
  intf.th.Suspended := false;
end;

end.
```