Webservice-Client mit Lazarus

Michael Fuchs

Lazarusforum 2019

14.09.2019

Übersicht

- Was sind Webservices?
- Wie kann man Webservices mit Lazarus benutzen?
- Beispiel - Programm zur Anbindung eines XML-RPC-Webservices

Definition Webservice

Ein Webservice ermöglicht die Maschine-zu-Maschine-Kommunikation auf Basis von HTTP oder HTTPS über Rechnernetze wie das Internet. Dabei werden Daten ausgetauscht und auf entfernten Computern Funktionen aufgerufen. Jeder Webservice besitzt einen Uniform Resource Identifier (URI), über den er eindeutig identifizierbar ist, sowie eine Schnittstellenbeschreibung in maschinenlesbarem Format (als XML-Artefakt, z. B. WSDL), die definiert, wie mit dem Webservice zu interagieren ist. Die Kommunikation kann (muss aber nicht) über Protokolle aus dem Internetkontext wie HTTP laufen und kann XML- oder JSON-basiert sein.

Quelle: https://de.wikipedia.org/wiki/Webservice

Definition Webservice

„Ein Webservice ist wie eine Webseite, nur für Computer anstatt für Menschen.“

Quelle: mein Azubi

Webservice & Co

- **Webservices**
  - SOAP
  - XML-RPC
  - JSON-RPC
  - REST
  - Web API
  - ...
- **Transportprotokoll**
  - HTTP/ HTTPS
  - FTP, SMTP, ...
  - Disketten(?) → RFC 1149
- **Schnittstellenbeschreibung**
  - WSDL
  - WADL
  - ...

Lazarus und Webservices

- **Web Service Toolkit**
  - zur Anbindung an WSDL-Webservices
  - erzeugt automatisch Model- und Service-Klassen
- **FpHttpClient, Synapse, LNet, Indy**
  - Datentransport zu HTTP-Webservices
- **FpJson**
  - Parsen von JSON-Daten
- **FCL-XML und Laz2_XML**
  - Parsen von XML-Daten
- **...**

Beispiel-Projekt

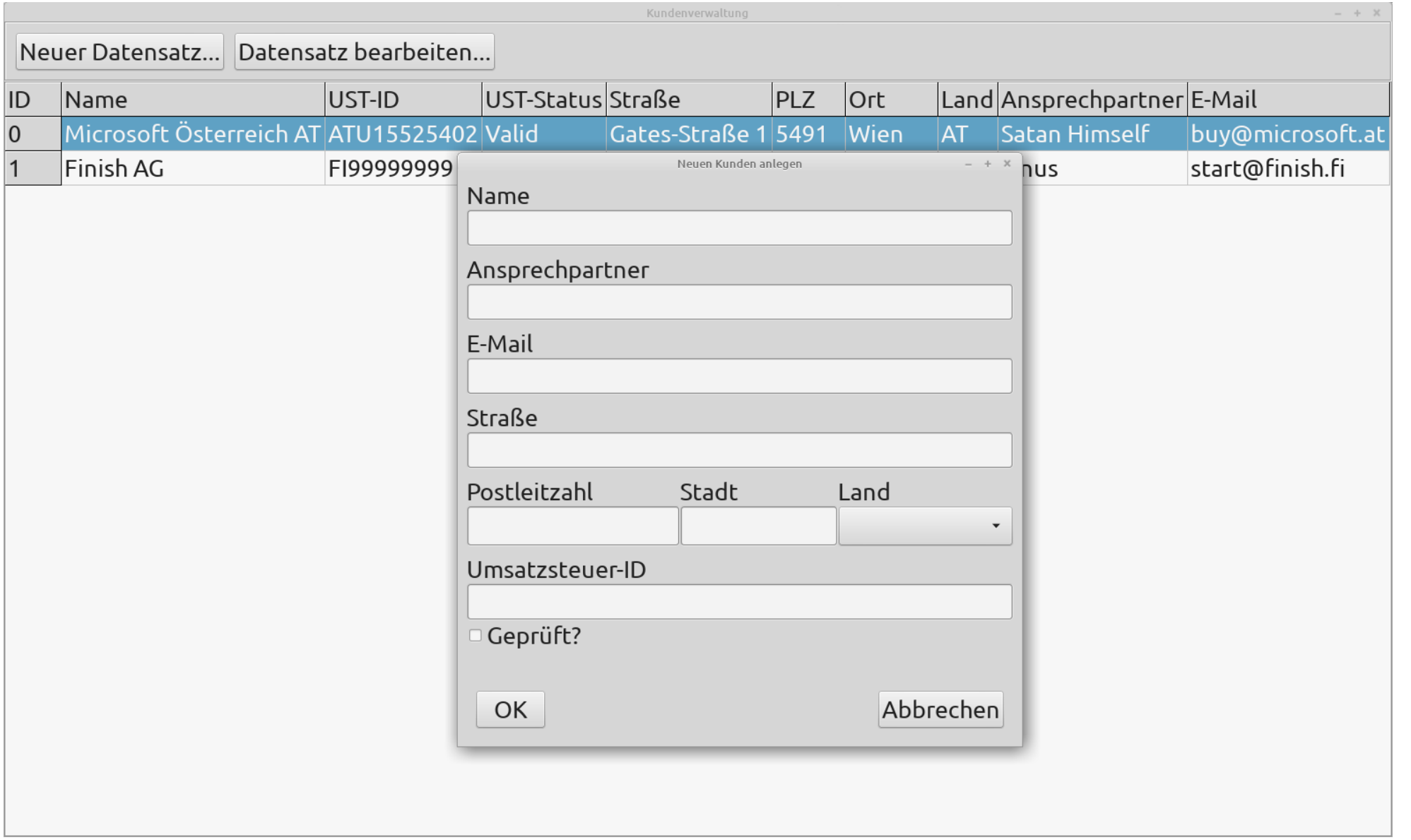
„Hey Katrin, das Anlegen eines neuen Geschäftskunden aus der Europäischen Union ist zu kompliziert. Der Sachbearbeiter muss von Hand auf der Webseite des Bundeszentralamt für Steuern die Umsatzsteuer-ID des Kunden prüfen.
Kann man das nicht automatisieren? “

Als Ergänzung zum Bestätigungsverfahren über das Internetformular bietet das Bundeszentralamt für Steuern (BZSt) eine Alternative an, die sich insbesondere an Unternehmen mit vielen Bestätigungsanfragen richtet.

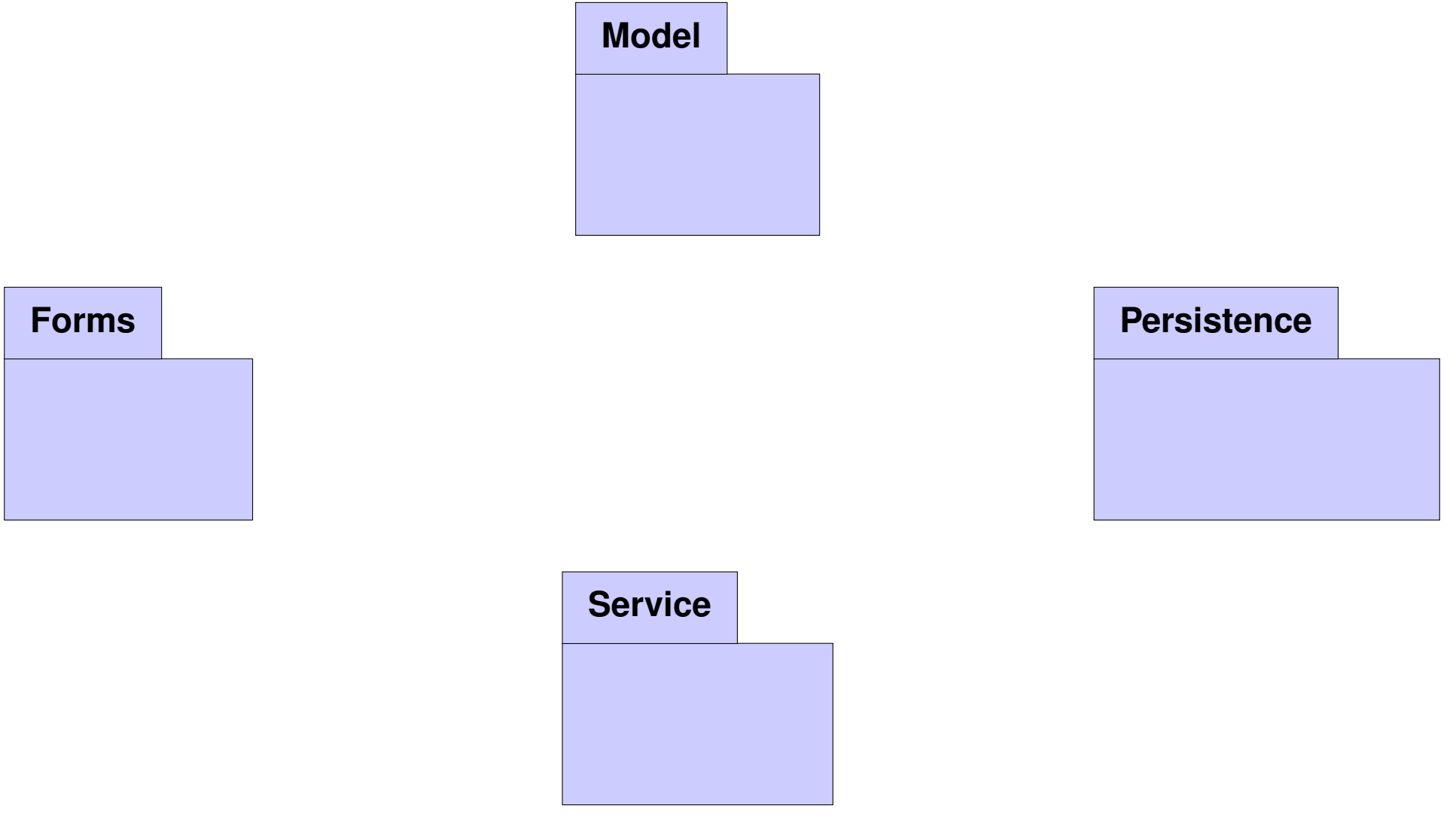
Über die sog. XML-RPC-Schnittstelle wird Unternehmen die Möglichkeit gegeben, die Prüfung von ausländischen Umsatzsteuer-Identifikationsnummern (USt-IdNrn.) in die eigenen Softwaresysteme einzubinden und die USt-IdNrn. automatisiert abzufragen.

Die Einbindung der XML-RPC-Schnittstelle muss durch das Unternehmen in Eigenregie erfolgen und setzt entsprechende Programmierkenntnisse voraus. Das Bundeszentralamt für Steuern kann hierbei keine Unterstützung oder Hilfe leisten.

Quelle: https://evatr.bff-online.de/eVatR/xmlrpc/


Kundenverwaltung
Neuer Datensatz...
Datensatz bearbeiten...
ID
Name
UST-ID
UST-Status
Straße
PLZ
Ort
Land
Ansprechpartner
E-Mail
0
Microsoft Österreich AT
ATU15525402
Valid
Gates-Straße 1
5491
Wien
AT
Satan Himself
buy@microsoft.at
1
Finish AG
FI99999999
start@finish.fi
Neuen Kunden anlegen
Name
Ansprechpartner
E-Mail
Straße
Postleitzahl
Stadt
Land
Umsatzsteuer-ID
Geprüft?
OK
Abbrechen

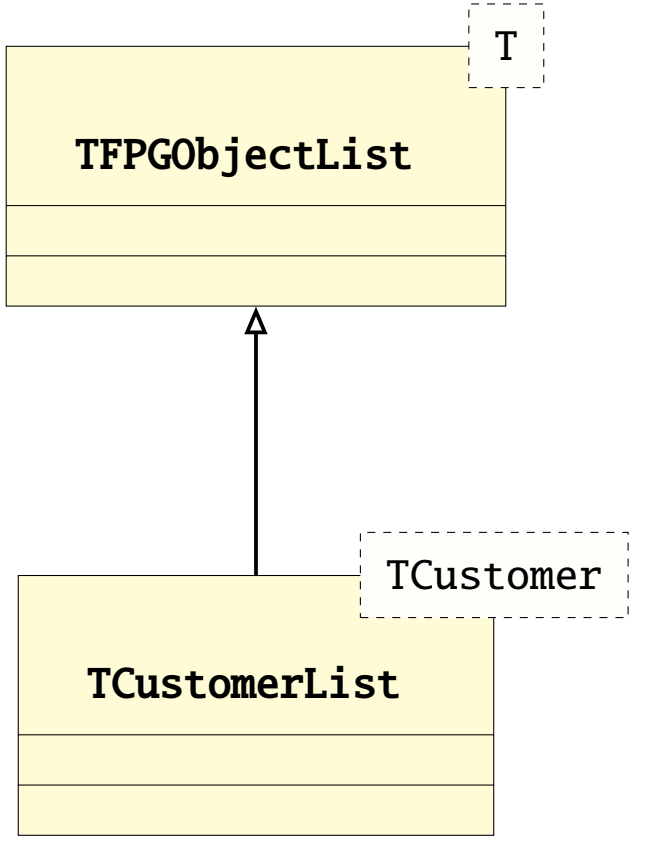
Packages



Model

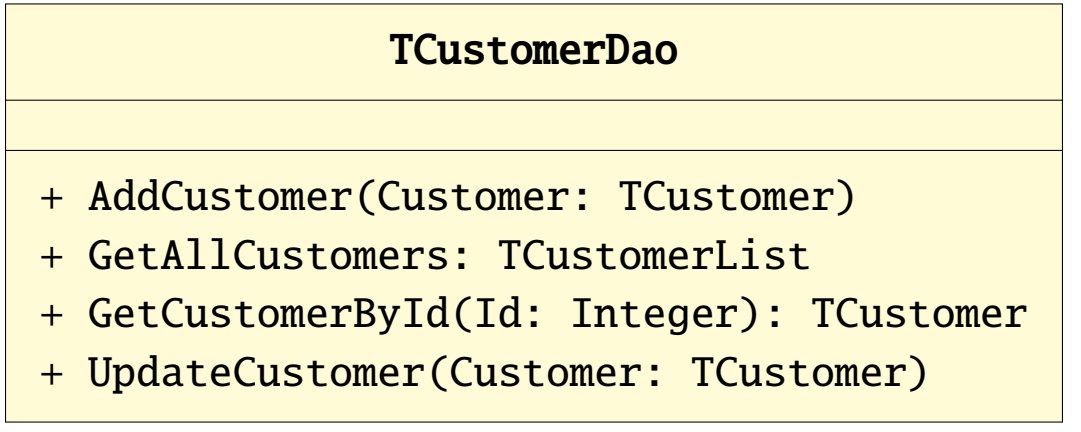
TCustomer

+ Id: Integer
+ City: String
+ ContactName: String
+ CountryCode: String
+ Email: String
+ Name: String
+ Street: String
+ VatIdNumber: String
+ VatStatus: Integer
+ ZipCode: String

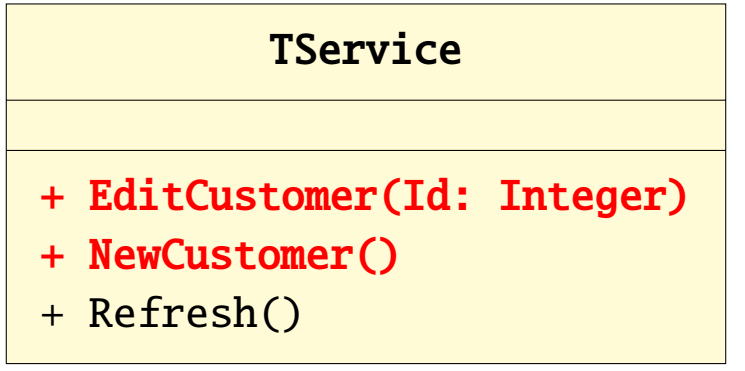



```
VAT_STATUS_UNKNOWN = 0
VAT_STATUS_VALID   = 1
VAT_STATUS_INVALID = 2
```

Persistenz


TCustomerDao
+ AddCustomer(Customer: TCustomer)
+ GetAllCustomers: TCustomerList
+ GetCustomerById(Id: Integer): TCustomer
+ UpdateCustomer(Customer: TCustomer)

Service



TService.NewCustomer

```
procedure TService.NewCustomer;
var
  Customer: TCustomer;
begin
  try
    Customer := TCustomer.Create;
    BindCustomerToForm(Customer);
    CustomerForm.Caption := 'Neuen Kunden anlegen';
    if (CustomerForm.ShowModal = mrOK) then begin
      RetrieveCustomerDataFromForm(Customer);
      Dao.AddCustomer(Customer);
    end;
  finally
    FreeAndNil(Customer);
  end;
  Refresh;
end;
```

TService.EditCustomer

```
procedure TService.EditCustomer(Id: Integer);
var
  Customer: TCustomer;
begin
  try
    Customer := Dao.GetCustomerById(Id);
    if Assigned(Customer) then begin
      BindCustomerToForm(Customer);
      CustomerForm.Caption := 'Kunden bearbeiten';
      if (CustomerForm.ShowModal = mrOK) then begin
        RetrieveCustomerDataFromForm(Customer);
        Dao.UpdateCustomer(Customer);
      end;
    end;
  finally
    FreeAndNil(Customer);
  end;
  Refresh;
end;
```

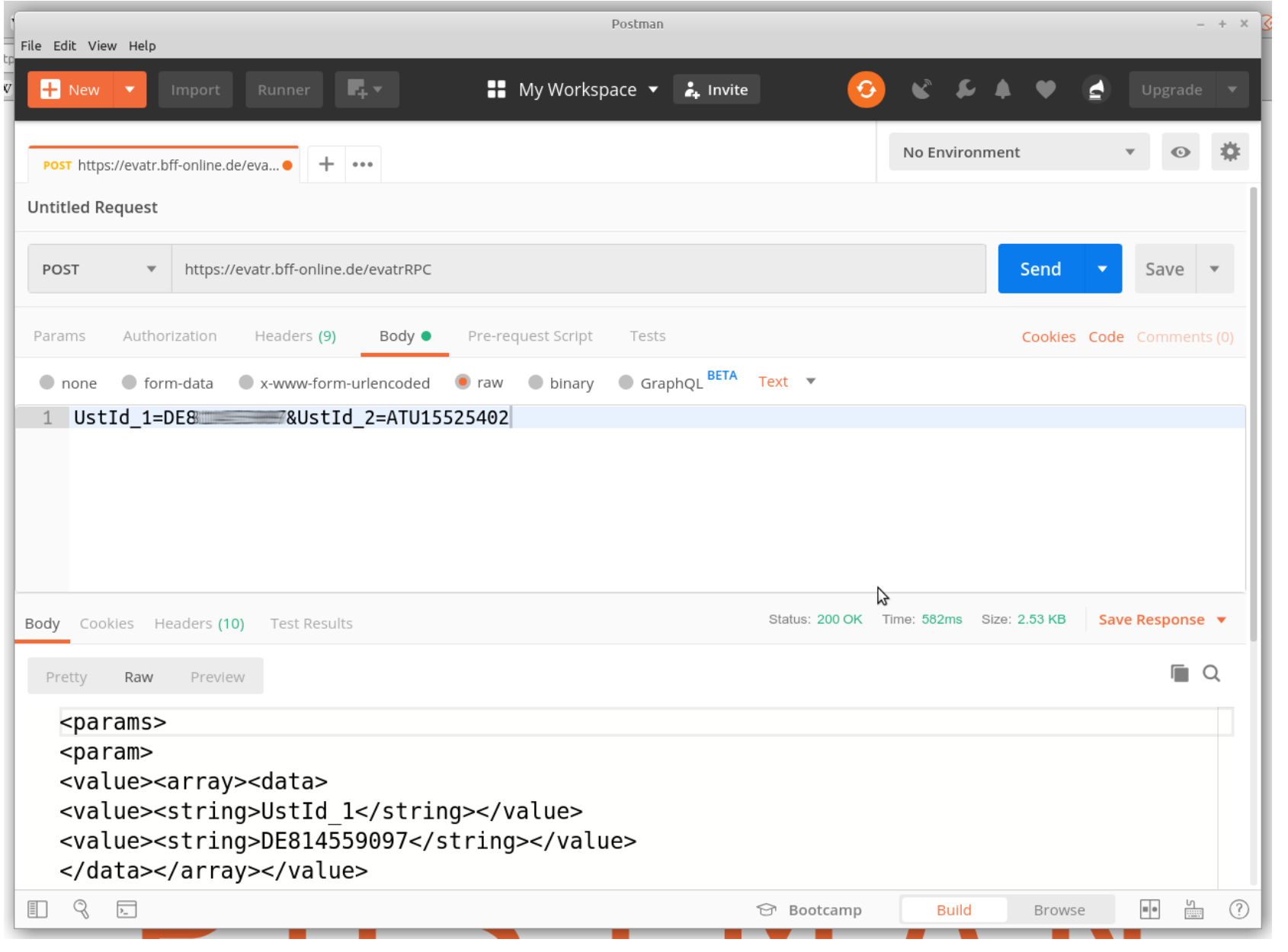
Webservice-Beschreibung

- Server-URL:
  - https://evatr.bff-online.de/
- Funktion mit Parametern:
  - evatrRPC(UstId_1, UstId_2, Firmenname, Ort, PLZ, Strasse, Druck)

Parameter	Datentyp	Case Sensitive	Beschreibung	Pflicht
UstId_1	String	X	Ihre deutsche UstIdNr	X
UstId_2	String	X	ausländische UStIdNr	X
Firmenname	String	—	Name & Rechtsform der Firma	—
Ort	String	—	Ort der Firma	—
PLZ	String	—	Postleitzahl der Firma	—
Strasse	String	—	Strasse und Hausnummer	—
Druck	String	—	ja = Bestätigungsmitteilung	—

Webservice-Test

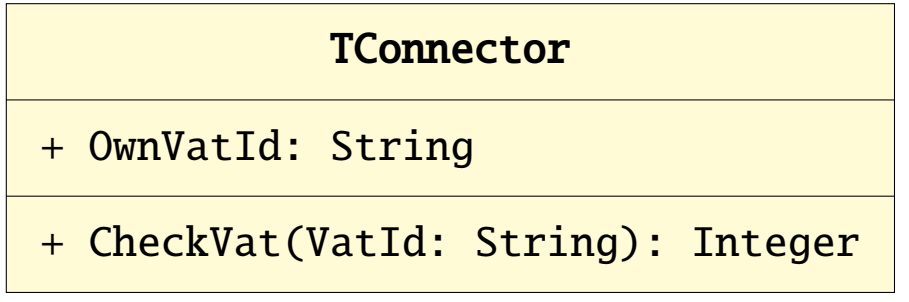
- URL:
  - https://evatr.bff-online.de/evatrRPC
- HTTP - Methode:
  - POST
- BODY:
  - UstId_1=DEXXXXXXXXX&UstId_2=ATU15525402
-

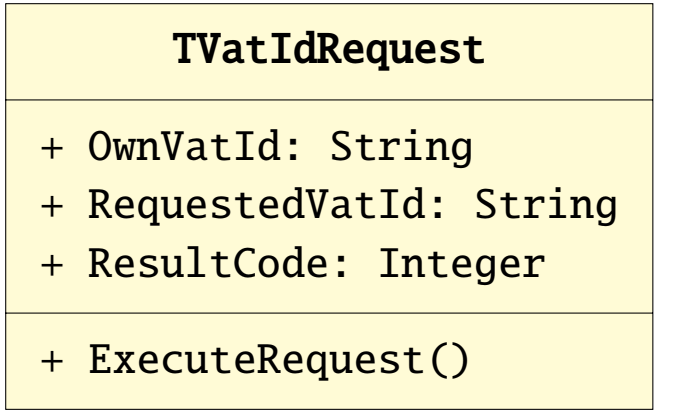

Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
POST https://evatr.bff-online.de/eva...
No Environment
Untitled Request
POST
https://evatr.bff-online.de/evatrRPC
Send
Save
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
Comments (0)
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
BETA
Text
1 UstId_1=DE8[redacted]&UstId_2=ATU15525402
Body
Cookies
Headers (10)
Test Results
Status: 200 OK
Time: 582ms
Size: 2.53 KB
Save Response
Pretty
Raw
Preview
<params>
<param>
<value><array><data>
<value><string>UstId_1</string></value>
<value><string>DE814559097</string></value>
</data></array></value>
Bootcamp
Build
Browse

```
<params>
<param>
<value><array><data>
<value><string>UstId_1</string></value>
<value><string>DEXXXXXXXXX</string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>ErrorCode</string></value>
<value><string>200</string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>UstId_2</string></value>
<value><string>ATU15525402</string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Druck</string></value>
<value><string>nein</string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Erg_PLZ</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Ort</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Datum</string></value>
<value><string>12.09.2019</string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>PLZ</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Erg_Ort</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Uhrzeit</string></value>
<value><string>20:46:33</string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Erg_Name</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Gueltig_ab</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Gueltig_bis</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Strasse</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Firmenname</string></value>
<value><string></string></value>
</data></array></value>
</param>
<param>
<value><array><data>
<value><string>Erg_Str</string></value>
<value><string></string></value>
</data></array></value>
</param>
</params>
```

```
<params>
  ...
  <param>
    <value>
      <array>
        <data>
          <value>
            <string>ErrorCode</string>
          </value>
          <value>
            <string>200</string>
          </value>
        </data>
      </array>
    </value>
  </param>
 ...
```

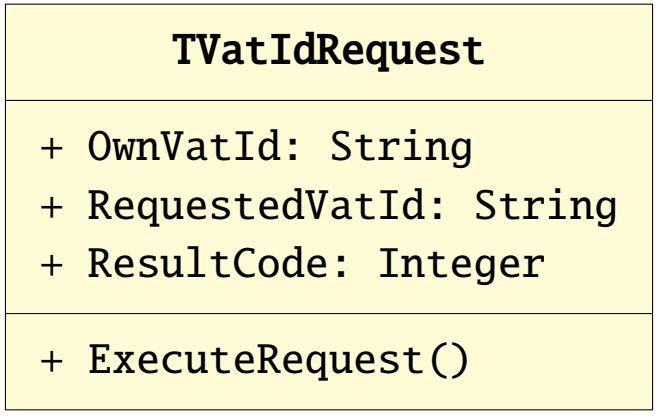
Entwicklung des Connectors





- **Pro**
  - einmalige Intialisierung
- **Kontra**
  - eventuell zusätzliches Rückgabeobjekt notwendig

- **Pro**
  - kein separates Rückgabeobjekt notwendig
- **Kontra**
  - Initialisierung bei jedem Request notwendig


TVatIdRequest
+ OwnVatId: String
+ RequestedVatId: String
+ ResultCode: Integer
+ ExecuteRequest()

Exkurs: ADR

Architecture Decision Records

- Dokumentation von Architekturentscheidungen
- enthält:
  - Problem
  - Entscheidungsvarianten
  - gewählte Variante
  - Begründung
  - Datum
  - Beteiligte
- Teil der Architektur-Dokumentation

Beispiel ADR

#42 :: Connector zum VAT-Webservice

2019-09-13: Katrin

Variante 1: Ein zentraler Connector, der eine CheckVatId-Methode enthält, die einen Request durchführt.

Variante 2: Ein Objekt, welches für exakt einen Request benutzt wird.

Ausgewählt wurde Variante 2, weil bei späteren Erweiterungen zusätzliche Daten direkt in der Request-Klasse gespeichert werden können. Bei Variante 1 wäre eine zusätzliche Rückgabe-Klasse notwendig.
Der erhöhte Initialisierungsaufwand ist bei der erwarteten Menge an Requests vernachlässigbar.

Implementation des Connectors

```
type
  TVatIdRequest = class(TObject)
    private
      FOwnVatId: String;
      FRequestedVatId: String;
      FResultCode: Integer;
    private
      function BuildRequestBody: TStream;
      procedure ExtractResultCode(Document: TXMLDocument);
      procedure ParseResponse(Response: TStream);
    public
      constructor Create(AnOwnVatId, ARequestedVatId: String);
    public
      property OwnVatId: String read FOwnVatId;
      property RequestedVatId: String read FRequestedVatId;
      property ResultCode: Integer read FResultCode write FResultCode;
    public
      procedure ExecuteRequest;
  end;
```

```
procedure TVatIdRequest.ExecuteRequest;
var
  Client: TFPHTTPClient;
  Request, Response: TStream;
begin
  FResultCode := 999;
  try
    Response := TMemoryStream.Create;
    Request := BuildRequestBody;
    Client := TFPHTTPClient.Create(nil);
    Client.RequestBody := Request;
    try
      Client.Post(CONNECTOR_URL, Response);
        if Client.ResponseStatusCode = 200 then ParseResponse(Response);
    except
      // do nothing
    end;
  finally
    FreeAndNil(Client);
    FreeAndNil(Response);
    FreeAndNil(Request);
  end;
end;
```

```
function TVatIdRequest.BuildRequestBody: TStream;
var
  BodyString: String;
begin
  BodyString := Format('UstId_1=%s&UstId_2=%s', [FOwnVatId, FRequestedVatId]);
  Result := TMemoryStream.Create;
  Result.Write(BodyString[1], BodyString.Length);
  Result.Seek(0, soFromBeginning);
end;
```

```
procedure TVatIdRequest.ParseResponse(Response: TStream);
var
  XmlDocument: TXMLDocument;
begin
  Response.Seek(0, soFromBeginning);
  try
    ReadXMLFile(XmlDocument, Response);
    ExtractResultCode(XmlDocument);
  finally
    FreeAndNil(XmlDocument);
  end;
end;
```

```
procedure TVatIdRequest.ExtractResultCode(Document: TXMLDocument);
var
  i: Integer;
  CurrentNode: TDOMNode;
  s: String = '';
begin
  for i := 0 to (Document.DocumentElement.ChildNodes.Count - 1)  do begin
    CurrentNode := Document.DocumentElement.ChildNodes.Item[i];
    if CurrentNode.FirstChild.FirstChild.FirstChild
                  .FirstChild.FirstChild.FirstChild
                  .NodeValue = 'ErrorCode' then begin
      s := CurrentNode.FirstChild.FirstChild.FirstChild
                      .ChildNodes[1].FirstChild.FirstChild.NodeValue;
    end;
  end;
  if not s.IsEmpty then
    FResultCode := StrToInt(s);
end;
```

Einbau in den Service

```
procedure TService.CheckVatIdOfCustomer(Customer: TCustomer);
var
  Request: TVatIdRequest;
begin
  try
    Request := TVatIdRequest.Create(OwnVatId, Customer.VatIdNumber);
    Request.ExecuteRequest;
    if (Request.ResultCode = 200) then
      Customer.VatIdStatus := VAT_STATUS_VALID
    else
      Customer.VatIdStatus := VAT_STATUS_INVALID;
  finally
    FreeAndNil(Request);
  end;
end;
```

```
procedure TService.NewCustomer;
var
  Customer: TCustomer;
begin
  try
    Customer := TCustomer.Create;
    BindCustomerToForm(Customer);
    CustomerForm.Caption := 'Neuen Kunden anlegen';
    if (CustomerForm.ShowModal = mrOK) then begin
      RetrieveCustomerData(Customer);
      CheckVatIdOfCustomer(Customer);
      Dao.AddCustomer(Customer);
    end;
  finally
    FreeAndNil(Customer);
  end;
  Refresh;
end;
```

22:59
erster Test → VALID

23:01
zweiter Test → INVALID

Logging Unit (1 / 2)

```
unit WsClientExample.Logging;
{$MODE ObjFpc}
{$H+}

interface

uses
  Classes, SysUtils, EventLog;

  procedure LogDebug(const Fmt: String; Args: array of const);

(* ... *)
```

Logging Unit (2 / 2)

```
implementation

var
  Log: TEventLog;

procedure LogDebug(const Fmt: String; Args: array of const);
begin
  Log.Debug(FormatString, Arguments);
end;

initialization
  Log := TEventLog.Create(nil);
  Log.FileName := 'app.log';
  Log.AppendContent := True;
  Log.LogType := ltFile;
  Log.Active := True;

finalization
  FreeAndNil(Log);

end.
```

```
(* ... *)
ReadXMLFile(XmlDocument, Response);
ExtractResultCode(XmlDocument);
LogDebug('Webservice returns resultcode %d. ', [ResultCode]);
(* ...*)
```

23:17
neuer Test → INVALID

„Webservice returns resultcode 217.“

- Katrin bittet einen Kollegen um Hilfe
- **Rubber-Ducking**
  - Schritt-für-Schritt-Erklärung des Problems
  - → führt häufig direkt zur Lösung

neuer Test → VALID

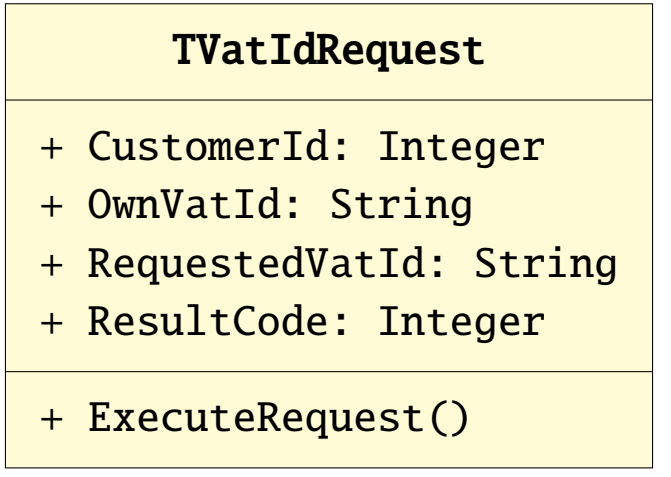
„Webservice returns resultcode 200.“

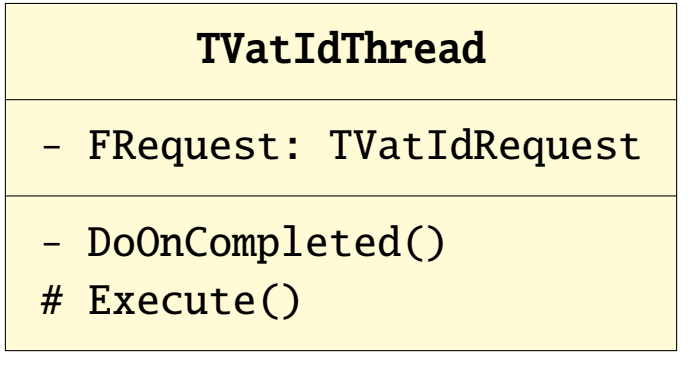
Was bedeutet eigentlich der Code 217?

„Bei der Verarbeitung der Daten ist ein Fehler aufgetreten. Ihre Anfrage kann deshalb nicht bearbeitet werden. “

Über diese Schnittstelle können Sie sich täglich, in der Zeit zwischen **05:00 Uhr** und **23:00 Uhr**, die Gültigkeit einer ausländischen Umsatzsteuer-Identifikationsnummer (USt-IdNr.) bestätigen lassen.

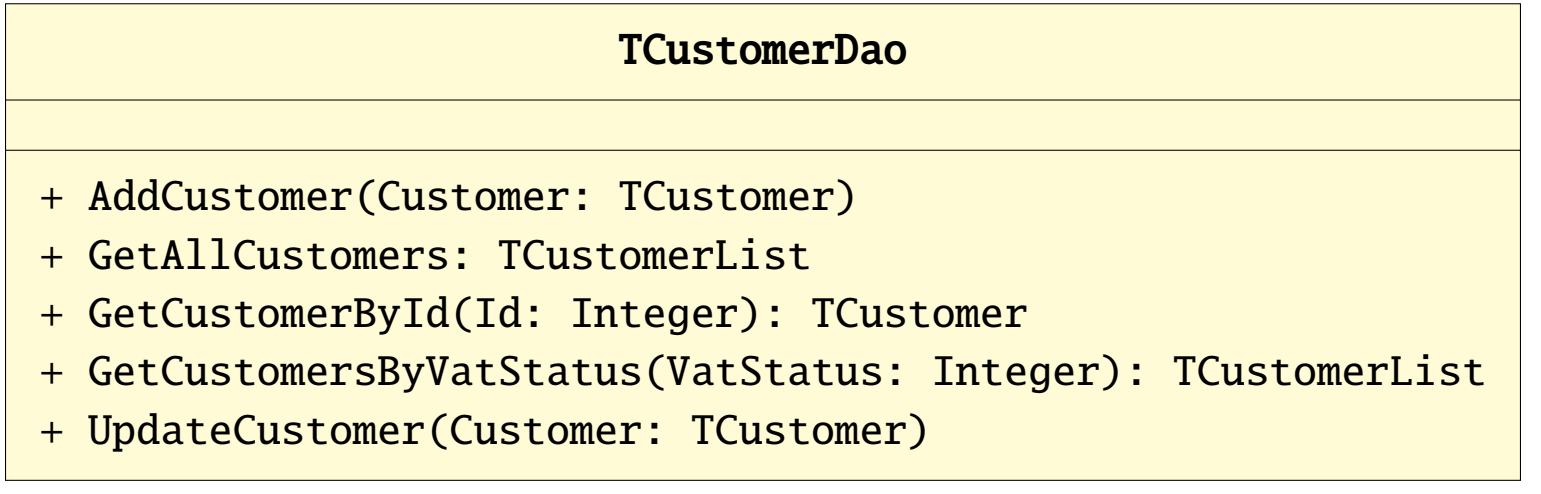
Lösungsideen

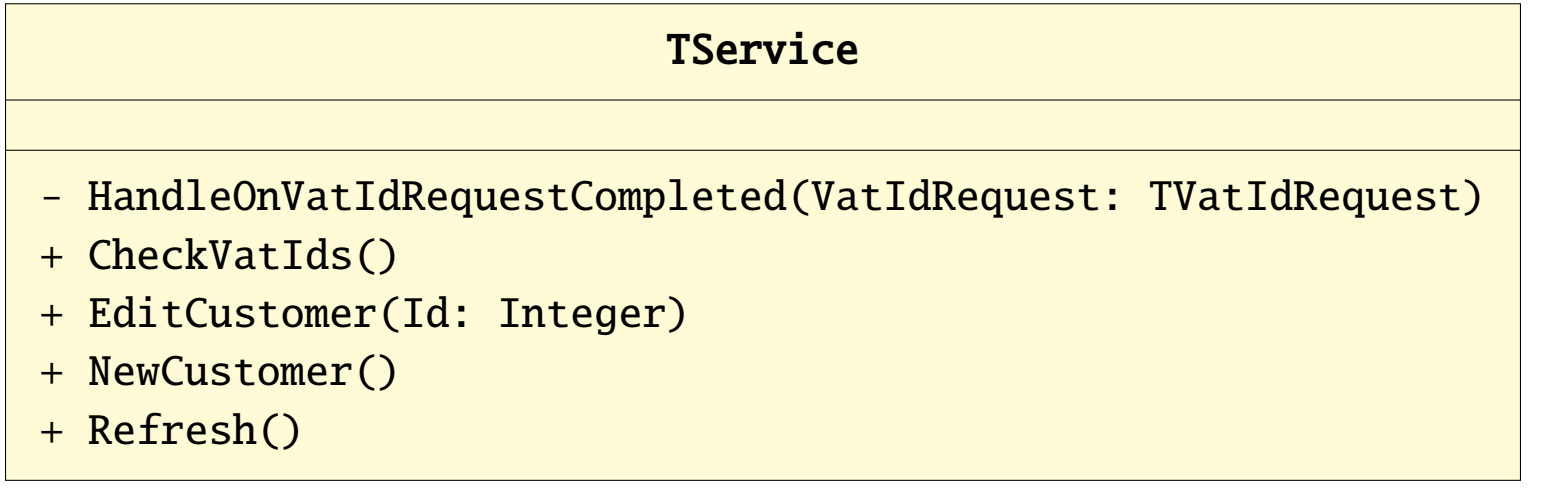

TVatIdRequest
+ CustomerId: Integer
+ OwnVatId: String
+ RequestedVatId: String
+ ResultCode: Integer
+ ExecuteRequest()


TVatIdThread
- FRequest: TVatIdRequest
- DoOnCompleted()
Execute()

```
type
  TVatIdRequestCompletedEvent = procedure(Request: TVatIdRequest) of object;

  TVatIdThread = class(TThread)
    private
      FOnCompleted: TVatIdRequestCompletedEvent;
      FRequest: TVatIdRequest;
    private
      procedure DoOnCompleted;
    protected
      procedure Execute; override;
    public
      constructor Create(ARequest: TVatIdRequest;
                         AOnCompleted: TVatIdRequestCompletedEvent);
    end;
```

```
constructor TVatIdThread.Create(ARequest: TVatIdRequest;
                                AOnCompleted: TVatIdRequestCompletedEvent);
begin
  inherited Create(True);
  FreeOnTerminate := True;
  FRequest := ARequest;
  FOnCompleted := AOnCompleted;
end;

procedure TVatIdThread.Execute;
begin
  FRequest.ExecuteRequest;
  Synchronize(@Self.DoOnCompleted);
end;

procedure TVatIdThread.DoOnCompleted;
begin
  if Assigned(FOnCompleted) then
    FOnCompleted(FRequest);
end;
```


TCustomerDao
+ AddCustomer(Customer: TCustomer)
+ GetAllCustomers: TCustomerList
+ GetCustomerById(Id: Integer): TCustomer
+ GetCustomersByVatStatus(VatStatus: Integer): TCustomerList
+ UpdateCustomer(Customer: TCustomer)


TService
- HandleOnVatIdRequestCompleted(VatIdRequest: TVatIdRequest)
+ CheckVatIds()
+ EditCustomer(Id: Integer)
+ NewCustomer()
+ Refresh()

```
procedure TService.CheckVatIds;
var
  Customers: TCustomerList;
  o: TCustomer;
begin
  if IsWebserviceRunning then begin
    try
      Customers := Dao.GetCustomersByVatStatus(VAT_STATUS_UNKNOWN);
      for o in Customers do begin
        ValidateVatId(o);
      end;
    finally
      FreeAndNil(Customers);
    end;
  end;
end;

function IsWebserviceRunning: Boolean;
begin
  Result := HourOf(Now) in [5..22];
end;
```

```
procedure TService.ValidateVatId(Customer: TCustomer);
var
  CurrentRequest: TVatIdRequest;
  CurrentThread: TVatIdThread;
begin
  if not(Customer.VatIdNumber.IsEmpty) then begin
    CurrentRequest := TVatIdRequest.Create(Customer.Id, OwnVatId,
                                           Customer.VatIdNumber);
    CurrentThread := TVatIdThread.Create(CurrentRequest,
                                         @Self.HandleOnVatIdRequestCompleted);
    CurrentThread.Start;
  end;
end;
```

```
procedure TService.HandleOnVatIdRequestCompleted(VatIdRequest: TVatIdRequest);
var
  Customer: TCustomer;
begin
  LogDebug('Handle VAT-ID request for Customer %d with result code %d. ',
           [VatIdRequest.CustomerId, VatIdRequest.ResultCode]);
  try
    Customer := Dao.GetCustomerById(VatIdRequest.CustomerId);
    if Assigned(Customer) then begin
      Customer.VatStatus := GetStatusByErrorCode(VatIdRequest.ResultCode);
      Dao.UpdateCustomer(Customer);
    end;
  finally
    FreeAndNil(VatIdRequest);
    FreeAndNil(Customer);
  end;
  Refresh;
end;
```