福井 和彦 / 畑中 侑

株式会社ミガロ.

システム事業部 システム2課

# Webサービスを利用して機能UP!
## ―既存機能に住所検索を追加

Delphi/400 で Web サービスを活用する。
この仕組みと方法を、住所検索の実装を通して紹介する。



略歴　福井　和彦
1972 年 03 月 20 日生
1994 年大阪電気通信大学工学部卒
2001 年 04 月株式会社ミガロ . 入社
2001 年 04 月システム事業部配属

現在の仕事内容
主に Delphi/400 を使用した受託開発で、要件確認から納品・フォローに至るまでのシステム開発全般に携わる。また、Delphi/400 の導入支援やセミナーの講師なども担当。

略歴　畑中　侑
1983 年 07 月 06 日生
2006 年京都産業大学法学部卒
2006 年 04 月株式会社ミガロ . 入社
2006 年 04 月システム事業部配属

現在の仕事内容
システム受託開発に携わって５年目。ミガロ . に入社し初めてプログラムを作成するも、現在は担当顧客を持ち、小規模から中規模案件のリーダーや大規模案件のサブリーダーを務めるに至る。

## 1.はじめに

基幹システムにおいて、郵便番号や住所を入力するケースは少なくない。例えば、取引先マスターを見てみると、そこには郵便番号や住所といった情報を持っており、請求書や納品書への印字に利用することが多い。また発注業務においては、納入場所の指定など、郵便番号を含めた住所情報などの入力が多かれ少なかれ求められる。

ただし、これらの情報を入力していくのはけっこうな手間ではある。特にクライアント / サーバー型の基幹システムの場合、すべて手入力をしていることの方が多いのではないだろうか。もし、郵便番号から住所を検索できる機能を実装することができれば、入力の手間を大きく軽減することが実現できる。

## 2.住所検索実装の課題

クライアント / サーバー型のシステムで住所検索を実装しようとした場合、郵便番号データのマスターファイルが必要となる。全国の郵便番号と住所情報は膨大な量になるが、日本郵政（郵便事業株式会社）のホームページよりダウンロード可能で、それを IBM i へ取り込むプログラムを作成することで、マスターファイルは作成できる。

だが、市町村合併等による郵便番号情報の変更に対応して、日本郵政ホームページのデータも月１回のペースで更新されている。このため、住所検索をシステムに実装したとしても、住所情報を保持し常に最新に保つことが課題となる。

しかし、これまで住所情報をすべて手入力で対応してきたのであれば、この“あると便利な機能”を、さらに手間を掛けて新たに実装するのは難しい選択であろう。

## 3.Webサービスを利用する

そこで本稿では、Web サービスを利用した住所検索の実装方法についてご紹介したい。「Web サービス」とは、インターネット技術を利用した、さまざまなサイトで提供されているサービスである。

今回題材としている住所検索についても、Web サービスを提供しているサイトがある。このサービスを利用することで、前述の郵便番号のファイルや仕組みの追加等を必要とせず、また最新データに保つためにメンテナンスに時間を割くこともなく、住所検索を実現することができるのである。

## 4.仕組みについて

今回は例として、「郵便番号検索 API (※)」という Web サービスを利用して、住所検索の実装方法の説明を進めていく。

まず「郵便番号検索 API」を利用して住所検索を行う仕組みだが、Web サービスを提供している Web サーバーに対して、既定の書式の URL をリクエストする。そして、検索結果を XML 形式で取得し、検索結果を画面へ反映する流れ

図1

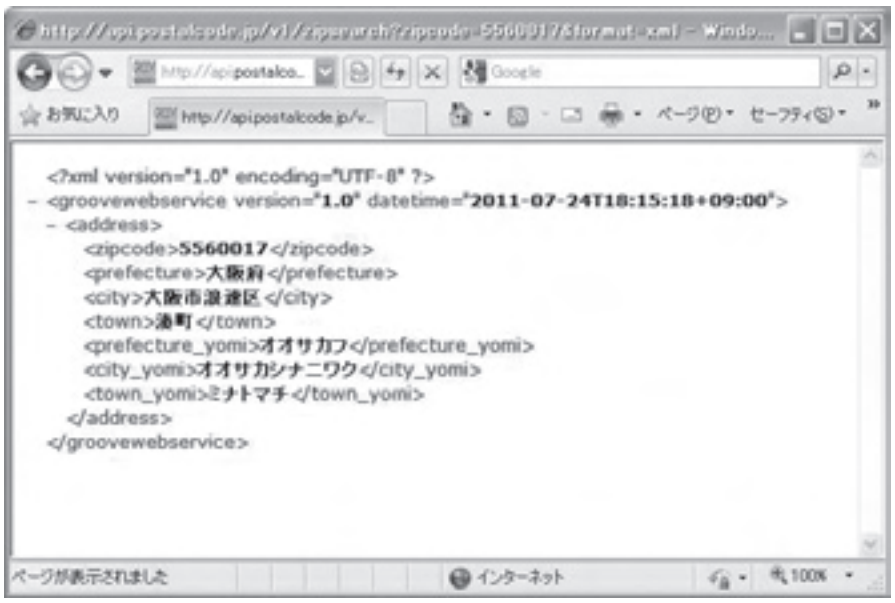



図2

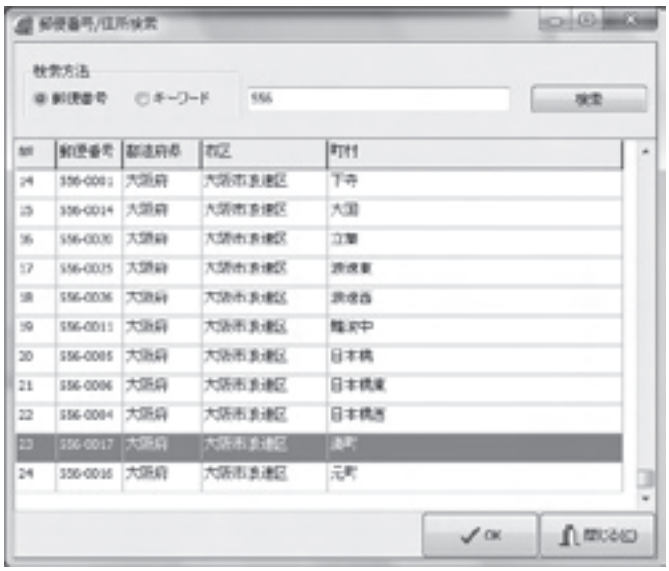



図3


<使用コンポーネント一覧>

| | コンポーネント | プロパティ Name |
|---|---|---|
| a | TPanel | Panel1 |
| b | TRadioGroup | RadioGroup1 |
| c | TEdit | Edit1 |
| d | TBitBtn | BitBtn1 |
| e | TStringGrid | StringGrid1 |
| f | TPanel | Panel2 |
| g | TBitBtn | BitBtn2 |
| h | TBitBtn | BitBtn3 |
| i | TXMLDocument | XMLDocument1 |
| j | TIdHTTP | IdHTTP1 |

となる。

次の例は、郵便番号 556-0017 で検索する場合の URL である。これをブラウザで実行すると、【図 1】のような XML が取得できる。

この仕組みを利用した住所検索の作成手順について、具体的に説明していきたい。

【例】郵便番号“556-0017”で検索
http://api.postalcode.jp/v1/zipsearch?zipcode=5560017&format=xml

※利用規約について

最近の Web サービスは、今回紹介する郵便番号検索をはじめ、地図表示から、ショッピング、オークション、商品検索までその種類も範囲も多種多様である。利用するにあたっては、Web サービスを提供している提供元がその利用方法を定めている。それが利用規約である。提供元の利用規約をよく読み、理解したうえで使用していただきたい。

本稿で取り上げている、グルーブテクノロジー提供の「郵便番号検索 API」についても同様で、クレジット表記および利用規約の同意が必要である。

【郵便番号検索 API】
http://groovetechnology.co.jp/index.html

# 5.画面イメージと実装機能

今回作成する住所検索のサンプル画面のイメージは、【図 2】となる。また、使用するコンポーネントは、【図 3】を参照していただきたい。

明細表示用コンポーネントとして、TStringGrid を使用している。このため、明細にデータを表示させるための項目転送ロジックを記述する必要がある。

そして、実装する検索機能としては、「郵便番号検索」と「キーワード検索」の 2 種類の方法を実装する。これらの機能は、Web サービス「郵便番号検索 API」で提供されている機能である。

**●郵便番号検索**

指定する郵便番号が 3 桁または 7 桁の場合のみ、検索結果が返ってくる。例えば郵便番号を 3 桁で指定した場合は、最初の 3 桁がその指定と一致する郵便番号が、検索結果として返ってくる。

**●キーワード検索**

指定したキーワードが、“都道府県”“市区”“町村”のいずれかに含まれている検索結果が返ってくる。

また、検索結果を呼び出し元画面へ反映する方法としては、OK ボタンの押下時に明細で選択されている郵便番号と住所を、フォームの property 経由で呼び出し元画面へ渡す仕組みとしている。

フォームの property 設定については、【ソース 1】を参照していただきたい。

# 6.実装方法

ここから、各機能の実装方法について、順番に紹介していく。

今回利用する Web サービスは、URL を指定すると、XML で情報を返してくれるサービスである。

### XML情報とXMLスキーマ

Web サービスで提供される XML には、定義されたタグが設定されている。【図 4】

これをスキーマ定義というが、XML 情報（図 1）を見るとわかるように、どんな情報が存在するかは容易に確認することができる。

ただ、プログラムから利用するにあたり、この文字列を解析して、プログラムを作成するとなると大変である。Delphi/400 ではこの解析を自動で行い、ユニットとして作成する機能がある。それが「XML データバインディング」である。

この機能はウィザード形式になっており、XML ファイルを指定することで、XML 情報を簡単に扱うことのできるユニットが自動生成される仕組みである。開発環境の 1 機能として、Delphi/400 に備わっている。

この機能を利用すれば、文字列の XML 情報をプログラムで解析する必要がなくなり、自動生成されたユニットを利用することで、プロパティとして簡単にデータにアクセスできる。

XML データバインディングの手順は、【図 5】【図 6】を参照いただきたい。

また今回、XML データバインディングで使用する XML ファイルについては、グルーブテクノロジーの Web サイトより取得することが可能である。取得方法は【図 7】を参照いただきたい。

実際に作成されたユニットを見てほしい。XML 情報で確認したタグごとに設定された項目が、自動でプロパティとしてアクセスできるようになっていることがわかるだろう。【ソース 2】

また、グローバル関数として XML 情報を引数に持ち、プロパティに格納するメソッドも自動で生成される。今回はこれらを利用することで簡単に実現することができる。【ソース 3】

### コンポーネント

Delphi/400 で利用するにあたり、ポイントとなるコンポーネントは 2 つある。

URL を指定して結果を取得する TIdHTTP コンポーネントと、XML を扱うための TXMLDocument コンポーネントである。

### コーディング解説

ここから、コーディングの解説を行っていく。【ソース 4】

最初に、XML データバインディングで自動生成されたユニットを、ユニット参照する。

①サンプル画面に表示されているように、画面上に“検索方法”として設けた RadioGroup の ItemIndex により、Web サービスで設定されている URL 定型文に検索キーを埋め込む。（ソースサンプルでは、URL 定型文を const 定義として記述）

**●検索方法＝郵便番号の場合**

URL 定型文と郵便番号とで、リクエスト URL を作成し、変数“URL”に格納する。

【例】
http://api.postalcode.jp/v1/zipsearch?zipcode=556&format=xml&ie=Shift_JIS&oe=Shift_JIS

図4

| タグ | 説明 |
|---|---|
| address | 住所の郵便番号の情報です。zipcode, prefecture, city, town, prefecture_yomi, city_yomi, town_yomiが設定されます。 |
| office | 事業所の個別郵便番号の情報です。zipcode, prefecture, city, town, street, office_name, office_name_yomiが設定されます。 |
| zipcode | 郵便番号が、「xxxyyyy」の形式で設定されます。 |
| prefecture | 都道府県名が設定されます。 |
| city | 市区町村名が設定されます。 |
| town | 町域名が設定されます。 |
| office_name | 大口事業所等名が設定されます。 |
| street | 大口事業所等名の小字名、丁目、番地等が設定されます。 |
| prefecture_yomi | 都道府県名のよみ(カタカナ)が設定されます。 |
| city_yomi | 市区町村名のよみ(カタカナ)が設定されます。 |
| town_yomi | 町域名のよみ(カタカナ)が設定されます。 |
| office_name_yomi | 大口事業所等名のよみ(カタカナ)が設定されます。 |

図5

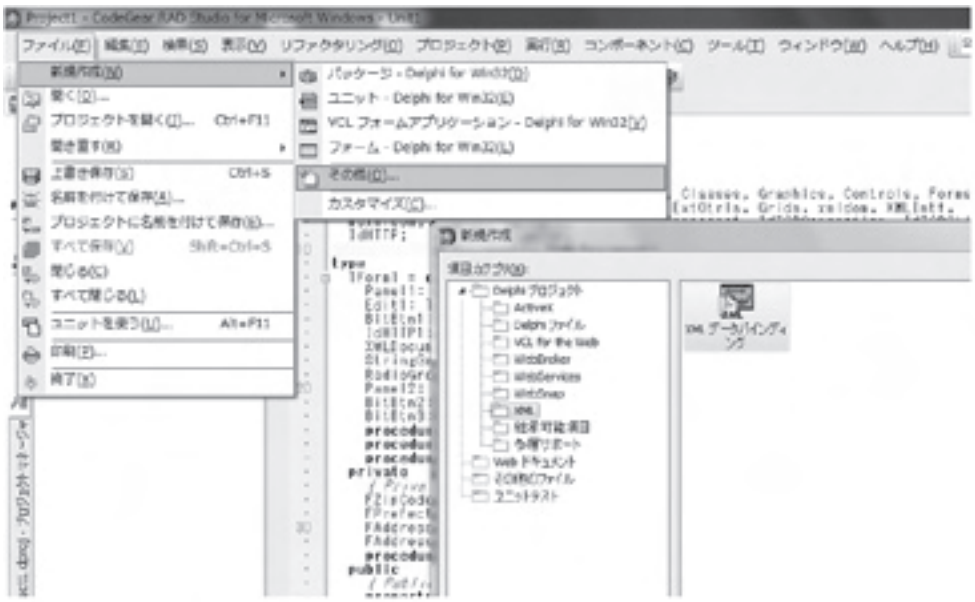
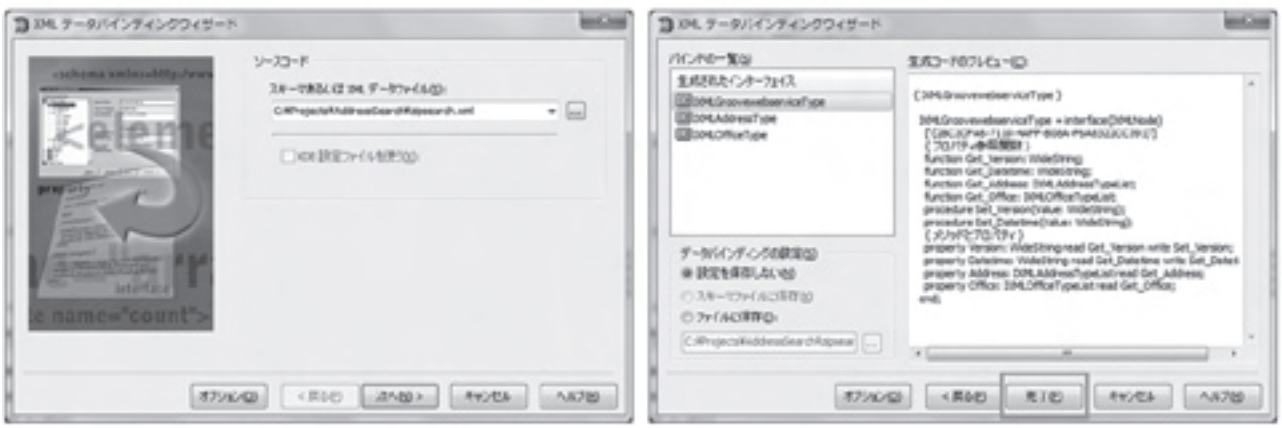


図6



●検索方法＝キーワードの場合

URL 定型文とキーワードとで、リクエスト URL を作成し、変数“URL”に格納する。

【例】

http://api.postalcode.jp/v1/zipsearch?zipcode= 湊町 &format=xml&ie=Shift_JIS&oe=Shift_JIS

② TIdHTTP コンポーネントの Get メソッドを、変数 URL で呼び出し、戻り値として得られる XML 情報の文字列を、TXMLDocument コンポーネントの XML プロパティにセットする。

③ XML データバインディングにより、自動生成されたユニットの関数 GetgrooveWebservice の引数に、TXMLDocument を指定する。

④関数 GetgrooveWebservice は、郵便番号などを配列のプロパティとして持つ。そのプロパティの値を取得し、取得件数分を画面に表示する。

またここで、前述の TStringGrid へ表示するための項目転送ロジックを記述することになる。

# 7.既存機能への組み込み

既存機能に郵便番号や住所を入力する画面があれば、完成した住所検索を組み込み、動作を確認していただきたい。

今回は住所検索を呼び出し、検索結果を既存画面へ反映させる機能であった。さらに、既存機能の郵便番号入力時に、住所を直接取得できるようにすることも可能である。

後者は、既存機能の郵便番号入力時にチェックとして、【ソース 5】の関数 GetAddress を組み込むことで実装が可能である。

なお、別ユニットのため、呼び出し元画面では uses 節への追加、関数を記述する前のフォームの Create、および呼び出し語のフォームの Release といった別途記述が必要である。

**コーディング解説**

①戻り値の初期化を行う。（それ以降の記述は、前章の検索画面で解説したものと重複するので割愛する）

②引数となる郵便番号で検索した結果、取得できない、あるいは結果が単一でない場合は、そのまま処理を終了する。

③検索した結果が単一である場合は、検索結果をプロパティに格納し、戻り値に True をセットする。

呼び出し元画面では、戻り値に応じて、True の場合はプロパティから必要な情報を画面にセットし、False の場合は、エラーとして処理するか検索画面を表示させる。こういった制御を行えば、ユーザーにもわかりやすいと思われる。

# 8.最後に

Delphi/400 で Web サービスを利用することができるということは、ご存知の方も多いだろう。しかし、具体的にどのサービスをどのように利用すればよいのか、わからない方も少なくないと思う。

今回“既存機能に住所検索を追加する”と利用目的を明確にし、具体的な事例の 1 つとして、Web サービスを利用した機能追加を紹介してきた。本稿を通して、Web サービスの利用方法について、少しでもイメージしていただけたのであれば幸いである。

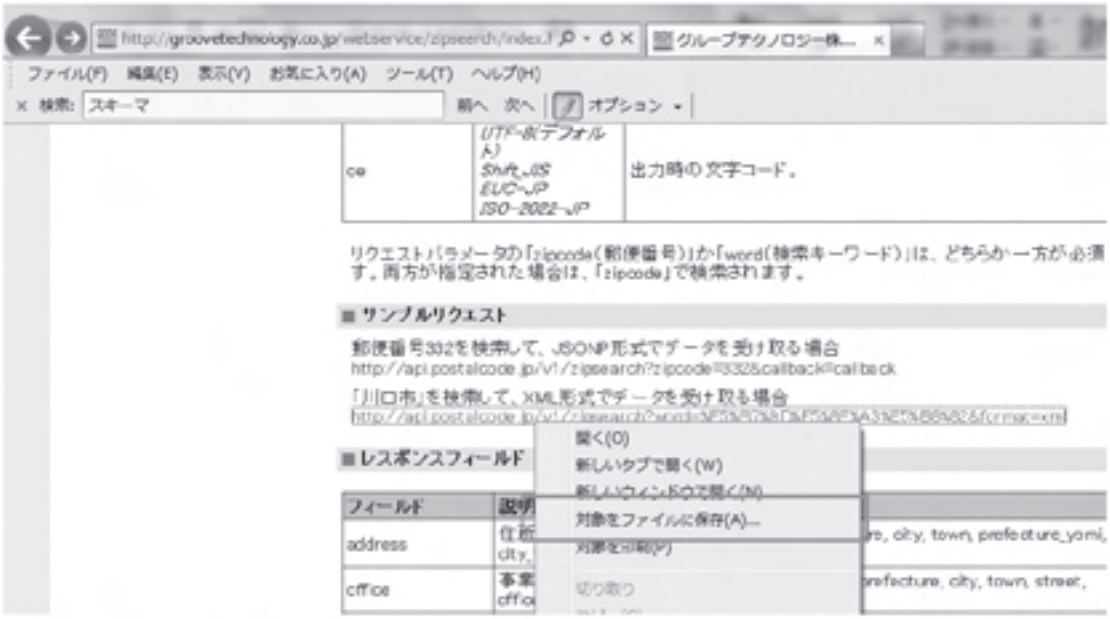
なお、Web サービスを利用する場合、サービスを提供する側の都合により仕様変更やサービスの停止が起きる可能性を認識しておく必要がある。したがって、Web サービスを利用した機能を実装した場合、定期的な Web サービス提供サイトの状況確認が必要となる。

このようなリスクはあるものの、昨今の Web サービスでは、他にもさまざまなサービスが提供されている。これらサービスを上手に利用していくことは、利便性の高いシステムを構築していくうえで手助けの 1 つになるのではないかと思う。

M

畑中 侑
現在の仕事内容（詳細）
システム受託開発に携わって 5 年目。ミガロ . に入社し初めてプログラムを作成するも、現在は担当顧客を持ち、小規模から中規模案件のリーダーや大規模案件のサブリーダーを務めるに至る。
Delphi/400 や RPG などのプログラム開発経験を積みながらスキルを磨きつつ、お客様のご要望に耳を傾け、一歩一歩提案力をつけるための修行中の毎日。

図7




参考URL(グループテクノロジー):http://groovetechnology.co.jp/webservice/zipsearch/index.html

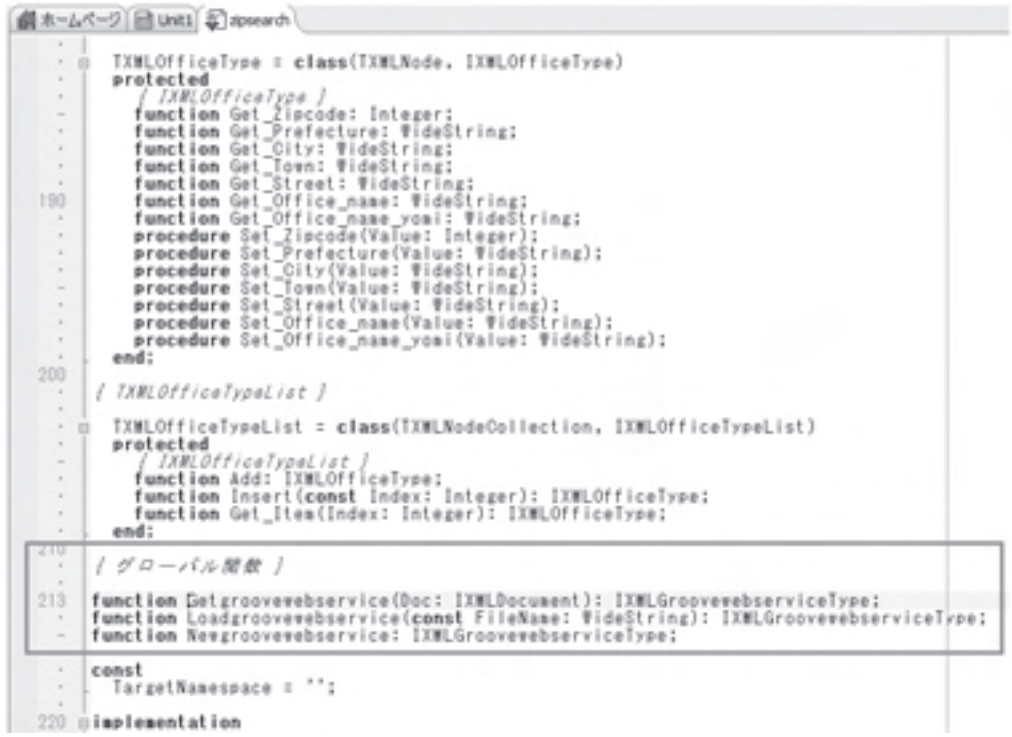
ソース1

```
private
  { Private declarations }
  FZipCode: String;
  FPrefecture: String;
  FAddress1: String;
  FAddress2: String;
  procedure ListClear;                    //-------StringGridの初期化
public
  { Public declarations }
  property ZipCode: String read FZipCode write FZipCode;                    //---プロパティ 郵便番号
  property Prefecture: String read FPrefecture write FPrefecture;           //---プロパティ 都道府県
  property Address1: String read FAddress1 write FAddress1;                 //---プロパティ 住所１
  property Address2: String read FAddress2 write FAddress2;                 //---プロパティ 住所２
  function GetAddress(AZipCode: String): Boolean;                           //---------(郵便)住所検索処理
end;
```

ソース2

```
unit zipsearch;
interface

uses xmldom, XMLDoc, XMLIntf;
type

{ 前方宣言 }

  IXMLGroovewebserviceType = interface;
  IXMLAddressType = interface;
  IXMLAddressTypeList = interface;
  IXMLOfficeType = interface;
  IXMLOfficeTypeList = interface;

{ IXMLGroovewebserviceType }

  IXMLGroovewebserviceType = interface(IXMLNode)
    ['{D99F2E98-49A2-4FA3-83E6-E0A72D3F3AE2}']
    { プロパティ参照関数 }
    function Get_Version: WideString;
    function Get_Datetime: WideString;
    function Get_Address: IXMLAddressTypeList;
    function Get_Office: IXMLOfficeTypeList;
    procedure Set_Version(Value: WideString);
    procedure Set_Datetime(Value: WideString);
    { メソッドとプロパティ }
    property Version: WideString read Get_Version write Set_Version;
    property Datetime: WideString read Get_Datetime write Set_Datetime;
    property Address: IXMLAddressTypeList read Get_Address;
    property Office: IXMLOfficeTypeList read Get_Office;
  end;

{ IXMLAddressType }

  IXMLAddressType = interface(IXMLNode)
    ['{0785D88E-2C51-43AE-A040-79995063DE20}']
    { プロパティ参照関数 }
    function Get_Zipcode: Integer;
    function Get_Prefecture: WideString;
    function Get_City: WideString;
    function Get_Town: WideString;
    function Get_Prefecture_yomi: WideString;
    function Get_City_yomi: WideString;
    function Get_Town_yomi: WideString;
    procedure Set_Zipcode(Value: Integer);
    procedure Set_Prefecture(Value: WideString);
    procedure Set_City(Value: WideString);
    procedure Set_Town(Value: WideString);
    procedure Set_Prefecture_yomi(Value: WideString);
    procedure Set_City_yomi(Value: WideString);
    procedure Set_Town_yomi(Value: WideString);
    { メソッドとプロパティ }
    property Zipcode: Integer read Get_Zipcode write Set_Zipcode;
    property Prefecture: WideString read Get_Prefecture write Set_Prefecture;
    property City: WideString read Get_City write Set_City;
    property Town: WideString read Get_Town write Set_Town;
    property Prefecture_yomi: WideString read Get_Prefecture_yomi write Set_Pref
    property City_yomi: WideString read Get_City_yomi write Set_City_yomi;
    property Town_yomi: WideString read Get_Town_yomi write Set_Town_yomi;
  end;

{ IXMLAddressTypeList }
```

ソース3



```
  IXMLOfficeType = class(TXMLNode, IXMLOfficeType)
  protected
    { IXMLOfficeType }
    function Get_Zipcode: Integer;
    function Get_Prefecture: WideString;
    function Get_City: WideString;
    function Get_Town: WideString;
    function Get_Street: WideString;
    function Get_Office_name: WideString;
    function Get_Office_name_yomi: WideString;
    procedure Set_Zipcode(Value: Integer);
    procedure Set_Prefecture(Value: WideString);
    procedure Set_City(Value: WideString);
    procedure Set_Town(Value: WideString);
    procedure Set_Street(Value: WideString);
    procedure Set_Office_name(Value: WideString);
    procedure Set_Office_name_yomi(Value: WideString);
  end;

{ IXMLOfficeTypeList }

  IXMLOfficeTypeList = class(TXMLNodeCollection, IXMLOfficeTypeList)
  protected
    { IXMLOfficeTypeList }
    function Add: IXMLOfficeType;
    function Insert(const Index: Integer): IXMLOfficeType;
    function Get_Item(Index: Integer): IXMLOfficeType;
  end;

{ グローバル関数 }

function Getgroovewebservice(Doc: IXMLDocument): IXMLGroovewebserviceType;
function Loadgroovewebservice(const FileName: WideString): IXMLGroovewebserviceType;
function Newgroovewebservice: IXMLGroovewebserviceType;

const
  TargetNamespace = '';

implementation
```

ソース4


```
var
  Form1: TForm1;

implementation

uses zipsearch;

const
  cSearch_URL = 'http://api.postalcode.jp/v1/zipsearch?';
  cSearch_Type1 = 'zipcode=';
  cSearch_Type2 = 'word=';
  cSearch_P1 = '&format=xml&ie=Shift_JIS&oe=Shift_JIS';

{$R *.dfm}

・・・　省　略

{--------------検索ボタン押下時--------------}
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  i: Integer;
  URL: String;
  XMLResultList: IXMLGroovewebserviceType;
begin
  //StringGridをクリアする
  ListClear;

  //入力チェック
  if (Edit1.Text = '') then
  begin
    //エラーメッセージ
    MessageDlg('検索条件を入力してください。', mtError, [mbYes], 0);
    Exit;
  end;

  //郵便番号での検索
  if RadioGroup1.ItemIndex = 0 then
  begin
    //郵便番号をセット
    URL := cSearch_URL + cSearch_Type1 + (Edit1.Text) + cSearch_P1;
  end;
  //キーワードでの検索
  if RadioGroup1.ItemIndex = 1 then
  begin
    //キーワードをセット
    URL := cSearch_URL + cSearch_Type2 + (Edit1.Text) + cSearch_P1;
  end;

  //送信処理を行い、レスポンスのXMLをXMLDocument1に格納
  XMLDocument1.XML.Text := IdHTTP1.Get(URL);

  // XMLDocument1のXMLから結果セットを取得
  XMLResultList :=  Getgroovewebservice(XMLDocument1);

  //受け取ったXMLデータから住所リストをStringGridへセット
  for i := 0 to XMLResultList.Address.Count - 1 do
  begin
    //行追加
    if i <> 0 then
    begin
      StringGrid1.RowCount := StringGrid1.RowCount + 1;
    end;

    //リストへセット
    StringGrid1.Cells[0, i + 1] := IntToStr(i + 1);

    StringGrid1.Cells[1, i + 1] := FormatFloat('000-0000',
                                   XMLResultList.Address.Items[i].Zipcode);

    StringGrid1.Cells[2, i + 1] := XMLResultList.Address.Items[i].Prefecture;

    StringGrid1.Cells[3, i + 1] := XMLResultList.Address.Items[i].City;

    StringGrid1.Cells[4, i + 1] := XMLResultList.Address.Items[i].Town;
  end;
end;
```

ソース5

```
{--------------[関数]住所情報取得--------------}
function TForm1.GetAddress(AZipCode: String): Boolean;
var
  URL: String;
  XMLResultList: IXMLGroovewebserviceType;
begin
  //初期処理
  Result := False;

  //郵便番号をセット
  URL := cSearch_URL + cSearch_Type1 + (AZipCode) + cSearch_P1;

  //送信処理を行い、レスポンスのXMLをXMLDocument1に格納
  XMLDocument1.XML.Text := IdHTTP1.Get(URL);

  // XMLDocument1のXMLから結果セットを取得
  XMLResultList :=  Getgroovewebservice(XMLDocument1);

  //取得件数を確認
  if XMLResultList.Address.Count <> 1 then
  begin
    //1件以外（取得できないor複数ある）の場合、処理終了
    Exit;
  end;

  //受け取ったXMLデータからプロパティへセット
  FZipCode    := FormatFloat('000-0000',
                 XMLResultList.Address.Items[0].Zipcode);  //---プロパティ 郵便番号
  FPrefecture := XMLResultList.Address.Items[0].Prefecture;;  //---プロパティ 都道府県
  FAddress1   := XMLResultList.Address.Items[0].City;  //---プロパティ 住所1
  FAddress2   := XMLResultList.Address.Items[0].Town;  //---プロパティ 住所2
  Result := True;
end;
```