Geokodierte Adressdatenbank

Franz Fiala

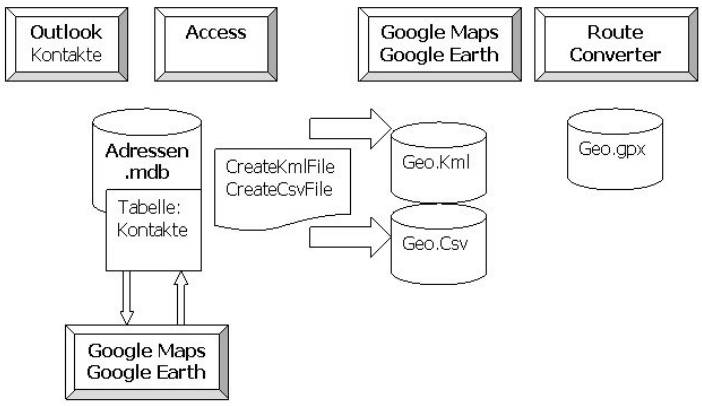
In **PCNEWS**-113, S.16 wurde gezeigt, wie man eigene Landkarten mit Positionsmarken, Skizzen und Bildern interaktiv mit Google-Tools herstellen kann. (*"Aktivitäten in Google Maps darstellen"*).

Die meisten PC-User haben aber ihre Adressen in einer Datenbank gespeichert. Nein? Also, wenn Sie Outlook verwenden und Ihre *Kontakte* Einträge enthalten, dann sind diese bereits in einer Datenbank gespeichert. Diese Outlook-Datenbank ist sogar erstaunlich flexibel, weil sie es erlaubt, Felder beliebigen Typs nachträglich einzusetzen. Anderseits ist sie aber nicht relational, mit der Folge, dass Adressfelder und Telefonnummern mehrfach vorkommen.

Will man auch eine geografische Darstellung, muss man in Google Maps oder Google Earth eine Positionsmarke einfügen oder man nimmt das Navi zur Hand und tippt mühsam Adressen ab, damit man die wichtigsten Ziele bereits vor der Fahrt eingetragen hat.

Daher müssten alle Adressen auf der Datenbank händisch in Google-Maps oder Google-Earth oder in das Navi übertragen werden. Eine Änderung in der Adressdatenbank muss immer auch in der Geo-Anwendung vorgenommen werden. Das sind zwei völlig unabhängige Verfahren und die beiden Adressdarstellungen, jene in der Datenbank und jene in der Geo-Anwendung hängen in keiner Weise zusammen und werden sich im Laufe der Zeit auseinander entwickeln.

Gesucht ist daher ein Weg, die Adressdatenbank durch geografische Koordinaten zu ergänzen und dann diese Daten in ein gängiges Navigationsformat zu übertragen, damit man es in Google Maps oder Google Earth oder in ein portables Navigationsgerät übertragen kann.



Arbeitsschritte zur Erzeugung Navi-kompatibler Geo-Datenbanken

Die Arbeitsschritte sind:

- Daten aus Outlook in Datenbank **Adresse.mdb** exportieren
- Tabelle um die Spalten **LAT**, **LNG** erweitern (nur beim ersten Mal erforderlich)
- Geodaten eintragen
- Geodaten und Ortsbezeichnungen in Datei **Kontaktexy.kml** exportieren mit Google Earth geokodieren und die Koordinaten in der Tabelle nachtragen.
- Datenbank **Adresse.mdb** in Outlook importieren und Eingabemaske anpassen
- Geodaten mit RouteConverter in gewünschtes Navi-Format umwandeln

Leider sind in der Outlook-Kontakt-Datenbank gleich mehrere Adressangaben und daher ist etwa ein Ort mehrfach enthalten (**Ortgeschäftlich**, **Ortprivat**, **WeitererOrt**). Hier empfiehlt es sich zu überlegen, ob man eventuell die Datenbank reduziert, nur eine dieser Adressangaben verwendet oder eben die geografischen Angaben für jeden dieser Orte vorsieht (**LAT1**, **LNG1**, **LAT2**, **LNG2**, **LAT3**, **LNG3**).

Für Outlook-Benutzer

Wenn Sie keine Access-Datenbank sondern Outlook zum Speichern Ihrer Kontaktadressen verwenden, müssen Sie zuerst Ihre Kontaktdaten in eine Access-Datenbank exportieren, dann sinngemäß wie hier beschrieben weiterarbeiten und nach Abschluss des Geokodierens diese ergänzte Datenbank wieder in Outlook importieren.

Die Befehlsfolgen gelten für Outlook 2003.

Kontakte exportieren

Outlook -> Wechseln zu -> Kontakte

Datei -> Importieren/Exportieren...

Exportieren in eine Datei -> Weiter

Microsoft Access -> Weiter

Aus diesem Ordner exportieren: Kontakte -> Weiter

Exportiere Datei speichern unter: **Adressen.mdb** *-> Weiter*

Felder zuordnen... -> Weiter

Grundsätzlich könnte man beim Export die Bezeichnungen der Felder mit *"Felder zuordnen..."* verändern. Wir lassen diese Bezeichnungen aber gleich.

Die weitere Beschreibung des Geokodierens ist wie in diesem Artikel und bezieht sich auf die exportierte Datenbank **Adressen.mdb**.

Geokodierte Kontakte importieren

Outlook-User müssen die geokodierte Access-Datenbank wieder in Outlook zurück übertragen.

Outlook -> Wechseln zu -> Kontakte

Datei -> Importieren/Exportieren...

Importieren aus anderen Programmen oder Dateien -> Weiter

Microsoft Access -> Weiter

zu importierende Datei auswählen -> *Duplikate durch importiere Elemente ersetzen*

Zielordner *Kontakte* wählen -> *Weiter*

Felder zuordnen... -> Fertig stellen

Wie beim Exportieren lassen wir die Feldzuordnung unverändert.

Eingabeformular anpassen

Wo sind jetzt diese beiden neuen Angaben **LAT** und **LNG**? In einem offenen Kontakt findet man sie nicht!

Wenn man diese beiden Feldern auch in Outlook bearbeiten will, muss man das Eingabeformular anpassen:

Extras -> Formulare -> Ein Formular entwerfen... -> Kontakt

Das Formular wird im Entwurfsmodus angezeigt und ein Fenster *Feldauswahl* öffnet sich. Hier wählt man Benutzerdefinierte Felder und zieht die beiden Felder **LAT** und **LNG** auf die Entwurfsfläche.

Extras -> Formular -> Formular veröffentlichen

Suchen in Kontakte -> "Geo"

Um dieses Formular für die Eingabe zu verwenden, muss man eingeben

Extras -> Formular -> Formular auswählen -> Suchen in: "Kontakte" -> "Geo"

Adressdatenbank

Eine Adressdatenbank kann beispielsweise folgende Felder haben:

`ANREDE, TITEL, VNAME, FNAME, STRASSE, PLZ ORT, LAND, EMAIL, TELEFON`

Die hervorgehobenen Felder sind für die geografische Position relevant. Als Datenbank benutzen wir Microsoft Access 2003, das ein Teil von Microsoft Office ist, die hier verwendete Beispieldatenbank hat den Namen `ADRESSEN.MDB`. Die einzige Tabelle heißt `KONTAKTE`. Alle Beispieldaten finden sich bei der Online-Version dieses Artikel.

Neue Felder in der Adressdatenbank

Mit jeder postalischen Neuerung, müssen Adressdatenbanken reorganisiert werden. Bereits Geschichte sind Änderungen nach Einführung der Postleitzahlen, der E-Mail-Adressen und auch der Handy-Rufnummern.

Seit es digitale Landkarten gibt, ist ein neuer Aspekt dazugekommen: die exakte Position auf einer Landkarte. Dazu benötigt man zwei zusätzliche Felder im Gleikommaformat: `LAT` (*latitude*) und `LNG` (*longitude*) für die geografische Breite und Länge. Die Namen der neuen Spalten wurden in Anlehnung an die in den Google APIs verwendeten Variablennamen gewählt. Man kann diese Spalten aber auch einfach `BREITE` und `LAENGE` nennen.

Wir bearbeiten den Entwurf der Tabelle `KONTAKTE` und fügen zwei Spalten `LAT` und `LNG` mit dem Felddatentyp `Zahl` und Feldgröße `Double` hinzu.

Es wird für eine endgültige Version notwendig sein, neben diesen Feldern für die Koordinaten auch noch ein oder mehrere `Ja/Nein`-Felder vorzusehen, die es dann erlauben, nur eine bestimmte Teilmenge an Adressen für den Export in das Navigationssystem zu übernehmen.

Wenn die Datenbank über diese Positions-Daten verfügt, dann ist es ein Leichtes, die ganze Adressdatenbank in ein Navigationssystem oder in Google-Earth zu übertragen ohne dort alle Adressangaben mühsam händisch eingeben zu müssen. Auch kann man sich im Programm wünschen, welche Daten in dem beschreibenden Text enthalten sein sollen, und ob alle oder nur einige Adressen enthalten sein sollen. Das Ergebnis ist jedenfalls eine sehr anschauliche geografische Darstellung des Adressmaterials sowohl für private aber auch für kommerzielle Anwendungen.

Die nachfolgende Beschreibung dokumentiert die Schritte zum Geokodieren einer Adressdatenbank.

Geokodieren

Mit dem Online-Tool Google-Maps und dem desktop-basierten Google-Earth ist es möglich, mit jeder Adresse der Adressdatenbank eine exakte geografische Position zu verbinden. Es gibt aber auch die Möglichkeit, diesen Vorgang mit Hilfe der Google-APIs zu automatisieren.

Geokodieren mit Google Maps

In Google Maps kann man die Koordinaten eines gefundenen Ortes so feststellen:

Eingabe einer Adresse in der Form `<Straße>, <Ort>, <Land>`, z.B. `Wexstraße 19, Wien, Österreich`.

Es gibt zwei Möglichkeiten, die Geokoordinaten zu erfahren: als Link im Menüpunkt *"Link"* und als gesendeter Text im Punkt *"Senden"*.

Link

Die Linkdaten sind:

```
http://maps.google.com/maps?f=q&source=s_q&hl=de&geocode=&q=Wexstra%C3%9Fe+19,+Wien,+%C3%96sterreich&sll=48.180976,16.371187&sspn=0.006653,0.01442&ie=UTF8&ll=48.236794,16.369672&spn=0.006646,0.01442&z=16&iwloc=A
```

Senden

Positionsdaten aus Google-Maps an verschiedene Ziele übertragen.

- **E-Mail**: funktioniert immer,
- **Handy**: Handy-Providern steht T-Mobile zur Auswahl, leider nicht A1;
- **Fahrzeug**: BMW und Mercedes implementiert,
- **GPS**: Clarion, Garmin, Insignia, Mio/Navman, Pioneer und TomTom.

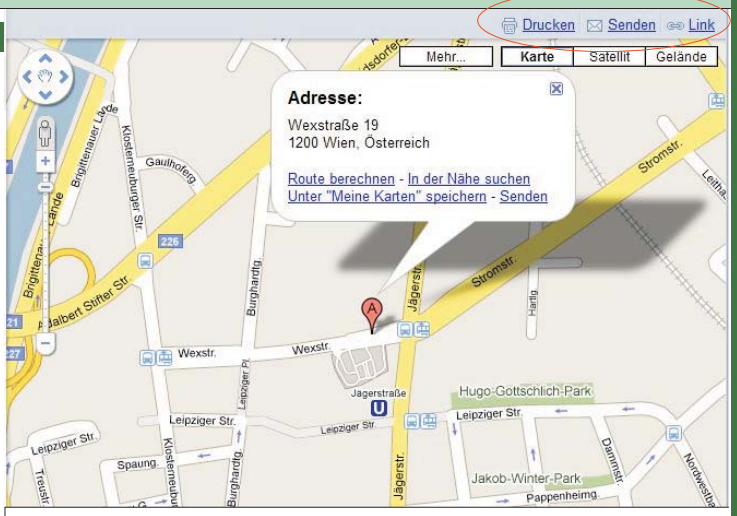
Die über E-Mail gesendete Nachricht enthält den Text, den man auch über den Punkt *"Link"* auf der Seite abrufen kann.

In diesem Link steckt die genaue geografische Position der gesuchten Adresse. Achtung: in den Links sind zwei Adressen angegeben:

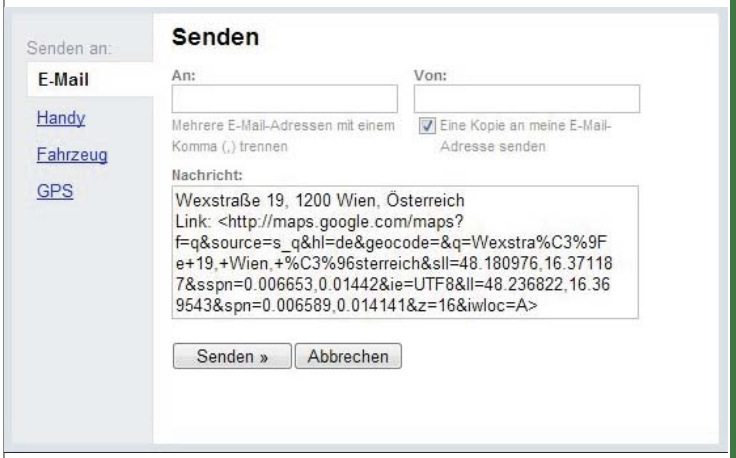
`sll=48.180976,16.371187` und `ll=48.236822,16.369543`

`sll` ist der eigene Standort (oder die unmittelbar vor der Suche gezeigte Position) und `ll` ist das gesuchte Ziel.

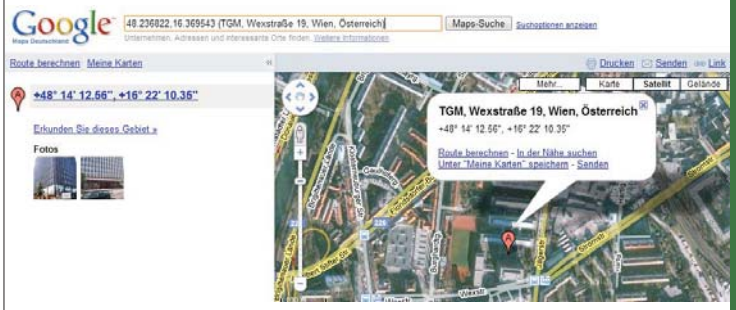
Man kann jederzeit kontrollieren, ob diese Zahlen auch korrekt sind, denn Google-Maps akzeptiert eine geografische Koordinate genau so wie eine Adresse. Kehrt man daher die Suche um und gibt die Koordinate ein, gefolgt von einem beliebigen eingeklammerten Text, kann man eine gegebene geografische Position überprüfen.



Anzeige einer gesuchten Adresse in Google Maps
Optionen: Drucken, Senden, Link



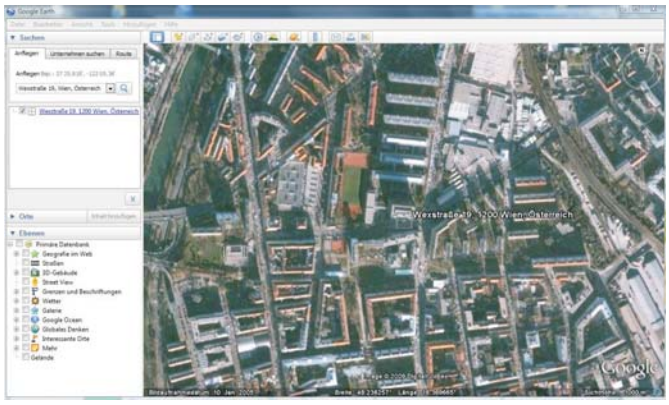
Dialog bei der Option "Senden"

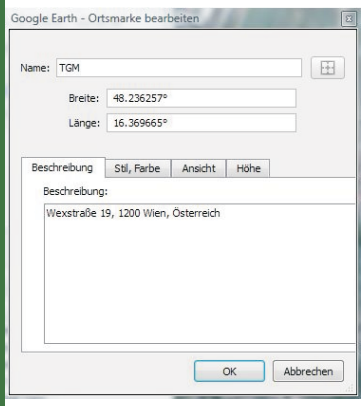


Die Wahl GPS erfordert, dass das Navi mit dem PC verbunden ist. Google leitet dann an den Navi-Erzeuger weiter und dieser speichert die Ortsmarke im Gerät. Beispiel für Garmin und diese Adresse erzeugt eine Datei `gpxfile10941664719.GPX` mit dem Inhalt:

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx creator="http://www.garmin.com" version="1.1"
xmlns="http://www.topografix.com/GPX/1/1"
xmlns:gpxx="http://www.garmin.com/xmlschemas/GpxExtensions/v3"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><wpt
lat="48.236257"
lon="16.369665"><time>2009-05-16T04:46:21Z</time>< name>Wexstraße 19, 1200
Wien,
Österreich</name><sym>Waypoint</sym><extensions><gpxx:WaypointExtension><
gpxx:Address><gpxx:StreetAddress>Wexstraße
19</gpxx:StreetAddress><gpxx:City>Wien</gpxx:City><gpxx:State>Wien</gpxx:
State><gpxx:Country>Österreich</gpxx:Country><gpxx:PostalCode>1200</gpxx:
PostalCode></gpxx:Address><gpxx:PhoneNumber>+436641015070</gpxx:PhoneNumb
er></gpxx:WaypointExtension></extensions></wpt></gpx>
```

Geokodieren mit Google Earth





Google Earth, der große Bruder von Google Maps, macht es ähnlich. Man gibt die Adresse unter *"Anfliegen"* ein und klickt auf die Lupe. Danach landet man bei der gesuchten Adresse. Die aktuelle geografische Position des Mauszeigers wird in der Statusleiste angezeigt. Man muss diese Zahlen aber nicht abtippen, sondern geht in das Kontextmenü des gesuchten Punktes und kann von dort die Koordinaten mit Copy&Paste kopieren. Ebenso kann man dem Punkt eine Beschreibung unterlegen.

Möchte man sehr viele Adressen anzeigen, dann ist Google Earth viel schneller in der Anzeige als Google Maps, weil alle Punkte am lokalen Rechner gespeichert sind und nicht über das Internet geladen werden müssen.

Geokodieren durch Direkte Abfrage des Google-Servers

Man kann die geografischen Koordinaten auch ohne einen besonderen Client vom Google-Server erfragen und zwar mit einem Request an die Adresse *http://maps.google.com/maps/geo* gefolgt von drei Parametern:

q Die Adresse, die man geokodieren möchte,

output Ausgabeformat mit den Optionen **xml, kml, csv** oder **json** und

key ein Schlüssel, den man für die eigene Seite anfordern muss, wenn man eine Karte im eigenen Web einbetten möchte.

Die Abfrage unserer Adresse schaut so aus:

```
http://maps.google.com/maps/geo?output=csv&q=Wexstraße 19,Wien,Österreich
```

die Antwort

```
200,8,48.2362441,16.3696704
```

wobei **200** der Statuskode **OK** ist und **8** eine Genauigkeitsangabe, danach die geografische Breite und Länge.

Die Option **key=** braucht man bei dieser Abfrage nicht.

Etwas mehr erfährt man mit der Option **output=kml**:

```
<?xml version="1.0" encoding="UTF-8" ?>
<kml xmlns="http://earth.google.com/kml/2.0">
  <Response>
    <name>Wexstraße 19,Wien,Österreich</name>
    <Status>
      <code>200</code>
      <request>geocode</request>
    </Status>
    <Placemark id="p1">
      <address>Wexstraße 19, 1200 Wien, Österreich</address>
      <AddressDetails Accuracy="8"
          xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">
        <Country>
          <CountryNameCode>AT</CountryNameCode>
          <CountryName>Österreich</CountryName>
          <AdministrativeArea>
            <AdministrativeAreaName>Wien</AdministrativeAreaName>
            <SubAdministrativeArea>
              <SubAdministrativeAreaName>Wien</SubAdministrativeAreaName>
              <Locality>
                <LocalityName>Wien</LocalityName>
                <DependentLocality>
                  <DependentLocalityName>Brigittenau</DependentLocalityName>
                  <Thoroughfare>
                    <ThoroughfareName>Wexstraße 19</ThoroughfareName>
                  </Thoroughfare>
                  <PostalCode>
                    <PostalCodeNumber>1200</PostalCodeNumber>
                  </PostalCode>
                </DependentLocality>
              </Locality>
            </SubAdministrativeArea>
          </AdministrativeArea>
        </Country>
      </AddressDetails>
      <ExtendedData>
        <LatLonBox north="48.2393917" south="48.2330965"
                   east="16.3728180" west="16.3665228" />
      </ExtendedData>
      <Point>
        <coordinates>16.3696704,48.2362441,0</coordinates>
      </Point>
    </Placemark>
  </Response>
</kml>
```

Hier werden alle Angaben zu Gemeinde, Bezirk und zum Bundesland sowie zum angezeigten Bereich eingeschlossen.

Die Option **output=xml** liefert exakt dasselbe Ergebnis. Der Unterschied ist nur der, dass die Struktur der Xml-Datei im Internet-Explorer angezeigt wird und die Kml-Datei als Download gespeichert wird.

Wenn daher nur die Koordinaten allein gefragt sind, genügt das Format **csv**. Wenn man auch alle postalischen Angaben möchte (Gemeinde, Bezirk, Bundesland, Postleitzahl) empfiehlt sich das Ausgaben-Format **kml** oder **xml**.

Dokumentation des Kml-Formats

http://code.google.com/intl/de-DE/apis/kml/documentation/kmlreference.html

Der nächste Schritt ist die Eintragung der Koordinaten aller Adressen. Und wie man das erledigt, hängt davon ab, um wie viele Adressen es sich handelt. Sind es nur wenige, dann kann man das sicher mit Google Earth in der vorhin beschriebenen Art bequem erledigen.

Wenn es aber viele Adressen sind, ist das mühsame Handarbeit und dabei kann uns ein kleines Programm, Google Earth und einige Kml-Kenntnisse weiter helfen.

Geokodieren mit Google Earth und Kml

Eine nicht unwichtige Kleinigkeit für das Geokodieren ist die folgende Eigenschaft einer Kml-Datei: Wie man an dem Beispiel in der linken Spalte sieht, sind in der Datei die Position sowohl indirekt in der Adresse im Tag **address** als auch direkt im Tag **Point** enthalten. Wenn beide vorhanden sind, dann hat **Point** die Präferenz. Wenn aber nur **address** vorhanden ist, dann versuchen Google Maps oder Google Earth daraus die Position zu ermitteln. Diesen Umstand kann man sich beim Geokodieren einer großen Zahl von Adressen zunutze machen und zwar so: man erzeugt aus gegebenen Adressen der Felder **STRASSE, ORT, LAND** einen Csv-Text (**Wexstraße 19, Wien, Österreich**) und fügt ihn in das Tag **address** einer durch ein Programm generierten Kml-Datei ein. Wenn man diese Datei mit einem Doppelklick mit Google Earth startet und zu dem betreffenden geografischen Bereich hinein zoomt, erlebt man so etwas wie die Abenddämmerung wenn am Nachthimmel nach und nach mehr Sterne sichtbar werden. Die "Sterne" sind in diesem Fall die Icons, jener Orte, die bereits durch eine Abfrage des Google-Servers berechnet worden sind. Bei diesem Vorgang muss daher eine Internetverbindung bestehen, sonst würde ja Google Earth nicht funktionieren, und Google Earth übernimmt die Aufgabe, die in der Kml-Datei enthalten textuellen Adressangaben durch geografische Koordinaten zu ergänzen, weil diese ja zur Anzeige der Symbole benötigt werden. Wenn man jetzt das Kontextmenü der gerade importierten Kml-Datei anklickt, und *"Ort speichern unter..."* wählt, erhält man eine um die geografischen Koordinaten ergänzte Kml-Datei. Damit hat man sich erspart, jede einzelne Adresse über Google Maps oder Google Earth anzufragen. Man hat sie alle in einer Kml-Datei gemeinsam mit den Adressen gespeichert und kann sie bequem in die Datenbank übertragen.

Das Resultat sollte eine Datenbank sein, die die Felder **LAT** und **LNG** ausgefüllt hat, etwa wie im folgenden Beispiel:

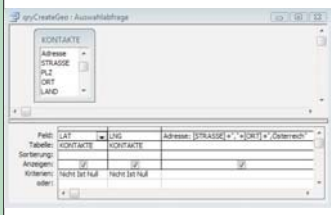
Microsoft Access - [RAPID_TOPOGRAFIE : Tabelle]

Datei Bearbeiten Ansicht Einfügen Format Datensätze Extras Fenster ?

ID	STRASSE	PLZ	ORT	LAN	LAT	LNG
100	Johnstraße 05	1150	Wien	A	48,192947388	16,316644669
101	Johnstraße 05	1150	Wien	A	48,192947388	16,316644669
24	Felberstraße 110	1150	Wien	A	48,193103790	16,319190979
102	Johnstraße 07	1150	Wien	A	48,193183899	16,316766739
21	Felberstraße 104	1150	Wien	A	48,193435669	16,320341110

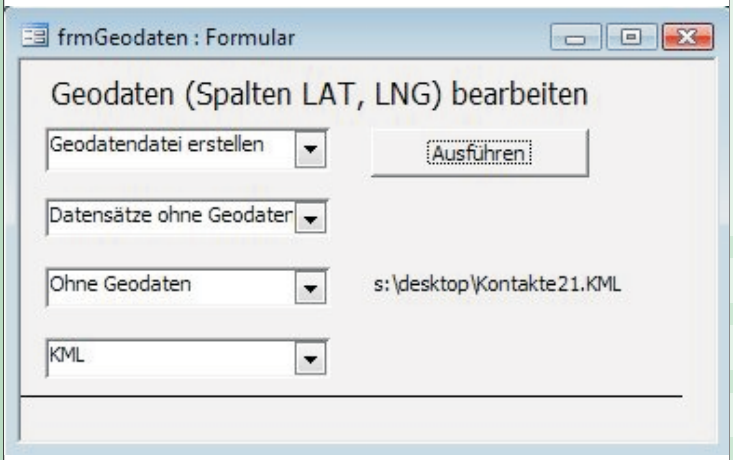
Datenexport mit Visual Basic

Jede Access-Datenbank verfügt über die eingebaute Sprache *Visual Basic for Applications*, mit der Erweiterungen der Grundfunktionalität von Access programmiert werden können. Die Beispieldatenbank `Adressen.mdb` enthält die (eventuell importierte) Tabelle `KONTAKTE`.



Damit man die Namen der exportierten Datensätze verändern kann, ohne das Programm dabei ändern zu müssen, wird die Abfrage `qryCreateGeo` formuliert. Die Namen der Felder für Länge, Breite, Ort, Straße, Land werden durch die Abfrage in die Ausgabenamen `LAT`, `LNG` und `ADRESSE` überführt. Diese Abfrage ist auch die Quelle für das verarbeitende Programm.

Das Formular `frmGeodaten` erleichtert den Aufruf des Programms und zeigt, welche Datei generiert wird:



Die Grundfunktion ist

- das Löschen der Geodaten in der Tabelle `KONTAKTE` und
- das Erstellen eine Geodatendatei

Die entstehende Geodatendatei kann das Format KML oder CSV haben (Dateiendung). Der Dateiname wird mit zwei Indizes versehen, sodass jede Einstellvariante eine anders benannte Ausgabedatei erzeugt. Es können alle Datensätze, die noch nicht kodierten Datensätze oder die bereits kodierten Datensätze (erster Index 0,1,2) in diese Datei exportiert werden. Die Datensätze können mit oder ohne Geodaten (zweiter Index 0,1) generiert werden. Es können daher insgesamt 12 verschiedene Dateien entstehen.

Den Dateipfad und den Dateinamen stellt man in Form von Konstanten im Programmkode ein.

Wenn noch keine Adressen geokodiert wurden, stellt man ein:

- Geodatendatei erstellen
- Alle Datensätze
- Ohne Geodaten
- Format KML

(Die anderen Einstellvarianten benötigt man, wenn zum Beispiel neue Adressen dazukommen und man eben nur diesen Adressen ohne Koordinatenangaben geokodieren will.)

Man erhält eine Datei `Kontakte01.KML`. Diese Datei doppelklicken.

Google Earth öffnet sich und kodiert durch alle Adressen automatisch durch Abfrage des Google Servers.

Kontextmenü den Ordner *"Temporäre Orte"* -> *"Ort speichern unter..."* -> `KontakteGeokodiert.kml`

Diese Datei ist die Grundlage zur Übernahme der Koordinatendaten aller Adressen in die Tabelle `KONTAKTE`, denn jetzt hat Google Earth die Tags `Point` und `coordinates` eingefügt, die man händisch in die Datenbank überträgt.

Warum händisch?

Access 2003 kann noch nicht mit XML-Dateien umgehen, daher kann die geokodierte Kml-Datei nicht ohne Weiteres in Access zurückgelesen werden. Wer daher Access 2007 installiert hat, kann versuchen, diese Kml-Datei zu importieren und danach mit einer Aktualisierungsabfrage die Felder `LAT` und `LNG` automatisch zu füllen. Schwierigkeiten wird dabei

VBA-Programm zum Export von Geodaten in eine KML- und eine CSV-Datei

```
Option Explicit
Option Compare Database
Dim iIndentLevel As Integer
Const iIndentValue = 2
Dim strOutput As String
Const strIcon = "http://www.google.com/mapfiles/marker_yellow.png"
Const CstrOutputPath = "s:\desktop\"
Const CstrOutputFile = "Kontakte"
Private Function GetOutputFile() As String
    Dim strFile As String
    strFile = CstrOutputFile
    Select Case Kombinationsfeld_Datensatzauswahl.Value
    Case 0 'Alle Datensätze
        strFile = strFile + "0"
    Case 1 'Datensätze mit Geodaten
        strFile = strFile + "1"
    Case 2 'Datensätze ohne Geodaten
        strFile = strFile + "2"
    End Select
    Select Case Kombinationsfeld_Geodaten.Value
    Case 0 'Mit Geodaten
        strFile = strFile + "0"
    Case 1 'Ohne Geodaten
        strFile = strFile + "1"
    End Select
    strFile = CstrOutputPath + strFile + "." + _
            Kombinationsfeld_Ausgabeformat.Column(0)
    GetOutputFile = strFile
End Function
Private Sub Kombinationsfeld_Aktion_AfterUpdate()
    InitControls
End Sub
Private Sub Kombinationsfeld_Ausgabeformat_AfterUpdate()
    Bezeichnungsfeld_Dateiname.Caption = GetOutputFile()
End Sub
Private Sub Kombinationsfeld_Datensatzauswahl_AfterUpdate()
    Bezeichnungsfeld_Dateiname.Caption = GetOutputFile()
End Sub
Private Sub Kombinationsfeld_Geodaten_AfterUpdate()
    Bezeichnungsfeld_Dateiname.Caption = GetOutputFile()
End Sub
Private Sub Befehl_Ausfuehren_Click()
    Select Case Kombinationsfeld_Aktion.Value
    Case 0 'Löschen
        GeodataClear
    Case 1 'Ausgeben
        GeodataCreate Kombinationsfeld_Ausgabeformat.Value, _
        Kombinationsfeld_Datensatzauswahl.Value, _
        Kombinationsfeld_Geodaten.Value
    End Select
End Sub
Private Sub Form_Load()
    InitControls
End Sub
Private Sub InitControls()
    Select Case Kombinationsfeld_Aktion.Value
    Case 0 'Löschen
        Kombinationsfeld_Ausgabeformat.Visible = False
        Kombinationsfeld_Geodaten.Visible = False
        Kombinationsfeld_Datensatzauswahl.Visible = False
        Bezeichnungsfeld_Dateiname.Visible = False
    Case 1 'Datei anlegen
        Kombinationsfeld_Ausgabeformat.Visible = True
        Kombinationsfeld_Geodaten.Visible = True
        Kombinationsfeld_Datensatzauswahl.Visible = True
        Bezeichnungsfeld_Dateiname.Visible = True
        Bezeichnungsfeld_Dateiname.Caption = GetOutputFile()
    End Select
End Sub
Private Sub GeodataClear()
    Dim Db As Database
    Dim Ds As Recordset
    Dim strSQLabfrage As String
    Set Db = CurrentDb()
    strSQLabfrage = _
        "UPDATE Kontakte SET Kontakte.LAT = 0, Kontakte.LNG = 0 "
    DoCmd.RunSQL (strSQLabfrage)
End Sub
Private Sub GeodataCreate(Format As Integer,
                Selection As Integer, WithGeodata As Integer)
    iIndentLevel = 0
    strOutput = ""
    Select Case Format
    Case 0 'Kml
        CreateKmlFile Selection, WithGeodata
    Case 1 'Csv
        CreateCsvFile
    End Select
End Sub
```

```
Private Sub CreateCsvFile()
    Dim Db As Database
    Dim Ds As Recordset
    Dim strSQLabfrage As String
    Set Db = CurrentDb()
    iIndentLevel = 0

    strSQLabfrage = "SELECT * FROM qryCreateGeo "
    strOutput = ""

    Set Ds = Db.OpenRecordset(strSQLabfrage)
    If Ds.EOF Then
        MsgBox "Keine Datensätze", vbInformation
        Exit Sub
    End If
    Do
        strOutput = strOutput + DoubleConvert(Ds("LNG")) + ", " + _
                    DoubleConvert(Ds("LAT")) + ", """ + _
                    Ds("Adresse") + """ + vbCrLf
        Ds.MoveNext
    Loop While Not Ds.EOF()
    Ds.Close
    WriteFile GetOutputFile(), strOutput
End Sub
Private Sub CreateKmlFile(Selection As Integer, WithGeodata As Integer)
    Dim Db As Database
    Dim Ds As Recordset
    Dim strSQLabfrage As String
    Set Db = CurrentDb()
    iIndentLevel = 0
    strSQLabfrage = "SELECT * FROM qryCreateGeo "
    Select Case Selection
    Case 0 'Alle Datensätze
    Case 1 'Datensätze mit Geodaten
        strSQLabfrage = strSQLabfrage + _
          "WHERE (((LAT)<>0) AND ((LNG)<>0))"
    Case 2 'Datensätze ohne Geodaten
        strSQLabfrage = strSQLabfrage + _
          "WHERE (((LAT)=0) OR ((LNG)=0))"
    End Select
    strOutput = "<?xml version=" + """ + "1.0" + """ + _
                " encoding=" + """ + "utf-8" + """ + "?>" + vbCrLf
    OpenTag "kml", "xmlns", "http://earth.google.com/kml/2.0"
    OpenTag "Document"
    Set Ds = Db.OpenRecordset(strSQLabfrage)
    If Ds.EOF Then
        MsgBox "Keine Datensätze", vbInformation
        Exit Sub
    End If
    Do
        OpenTag "Placemark"
            MakeTag "name", strToAscii(Ds("Name"))
            MakeTag "address", Ds("Adresse")
            Select Case WithGeodata
            Case 0 'Mit Geodaten
                OpenTag "Point"
                    MakeTag "coordinates", _
                        DoubleConvert(Ds("LNG")) + "," + _
                        DoubleConvert(Ds("LAT"))
                CloseTag "Point"
            Case 1 'Ohne Geodaten
            End Select
            OpenTag "Style"
                OpenTag "IconStyle"
                    OpenTag "Icon"
                        MakeTag "href", strIcon
                    CloseTag "Icon"
                CloseTag "IconStyle"
            CloseTag "Style"
        CloseTag "Placemark"
        Ds.MoveNext
    Loop While Not Ds.EOF()
    Ds.Close
    CloseTag "Document"
    CloseTag "kml"
    Dim strFile As String
    WriteFile GetOutputFile(), strOutput
End Sub
Private Function DoubleConvert(dDouble As Double) As String
    DoubleConvert = Replace(CStr(dDouble), ",", ".")
End Function
Private Sub OpenTag(TagName As String, _
            Optional Attr As String = "", Optional Value As String = "")
    Dim Tag As String
    Tag = TagName
    If (Attr <> "") Then
        Tag = Tag + " " + Attr
        If Value <> "" Then
            Tag = Tag + "=" + """ + Value + """
        End If
    End If
    Tag = MakeOpeningTag(Tag) + vbCrLf
    Tag = Indent(Tag)
    iIndentLevel = iIndentLevel + 1
    strOutput = strOutput + Tag
End Sub
Function MakeOpeningTag(Tag As String) As String
    MakeOpeningTag = "<" + Tag + ">"
End Function
Function MakeClosingTag(Tag As String) As String
    MakeClosingTag = "</" + Tag + ">"
End Function
Private Sub MakeTag(TagName As String, TagValue As String)
    Dim Tag As String
    Tag = MakeOpeningTag(TagName) + TagValue + MakeClosingTag(TagName)
    Tag = Indent(Tag) + vbCrLf
    strOutput = strOutput + Tag
End Sub
Private Sub CloseTag(TagName As String)
    Dim Tag As String
    Tag = MakeClosingTag(TagName) + vbCrLf
    iIndentLevel = iIndentLevel - 1
    Tag = Indent(Tag)
    strOutput = strOutput + Tag
End Sub
Private Function Indent(Tag As String) As String
    Dim strIndent As String
    Dim i, j As Integer
    strIndent = ""
    If iIndentLevel <> 0 Then
        For i = 0 To iIndentLevel - 1
            For j = 0 To iIndentValue
                strIndent = " " + strIndent
            Next j
        Next i
    End If
    Indent = strIndent + Tag
End Function
Private Function strToAscii(strUtf8 As String) As String
    'Dim strResult As String
    'strResult = ""
    'For i = 1 To Len(strUtf8)
    '    strResult = strResult + chrToAscii(Mid(strUtf8, i, 1))
    'Next i
    'strToAscii = strResult
    strToAscii = strUtf8
End Function
Private Function chrToAscii(strUtf8 As String) As String
    Dim strResult As String
    strResult = ""
    Select Case (strUtf8)
    Case "ä"
        strResult = "ae"
    Case "ö"
        strResult = "oe"
    Case "ü"
        strResult = "ue"
    Case "ß"
        strResult = "ss"
    Case "Ä"
        strResult = "Ae"
    Case "Ö"
        strResult = "Oe"
    Case "Ü"
        strResult = "Ue"
    Case Else
        strResult = strUtf8
    End Select
    chrToAscii = strResult
End Function
Private Sub WriteFile(FileName As String, Text As String)
    Dim fs, a
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set a = fs.CreateTextFile(FileName, True)
    a.Write (Text)
    a.Close
    MsgBox "Datei " + FileName + " angelegt", vbInformation
End Sub
```

bereiten, dass die KmlDatei kein Schlüsselfeld enthält. Das wäre dann eventuell im Programm nachzutragen.

Das angezeigte Symbol ist eine gelbe Ortsmarke vom Google-Server. Man kann sich alle möglichen Symbole vom Google-Server holen, es gilt aber die Regel, dass man sie für die eigene Anwendung am eigenen Server speichern soll.

Verzeichnis der Symbole

http://mapki.com/wiki/Available_Images

Übertragen zum Navi

Ist Google-Earth das Zielsystem, ist man schon fertig, denn ein Doppelklick auf die generierte Kml-Datei zeigt alle Ortsmarken an.

Die meisten Navigationsgeräte erlauben den Download von POIs *(Points Of Interest)* aus vorgefertigten XML-Dateien. Mein Gamin-Navi braucht dazu das Gpx-Format.

Der kostenlose RouteConverter (Version 1.26) von Christian Pesch (`www.routeconverter.de`) ist in seiner Universalität kaum zu übertreffen. Hier die Liste der unterstützten Navis:

- Alan Map 500 Waypoints and Routes (*.wpr)
- Alan Map 500 Tracklog (*.trl)
- Auto Onroute/Promotor Onroute (*.bcr)
- CoPilot 6 to 7 (*.trp)
- Falk Navigator (*.tour)
- Garmin MapSource 5.x (*.mps)
- Garmin MapSource 6.x (*.gdb)
- Garmin PCX5 (*.wpt)
- Garmin POI (*.gpi)
- Garmin POI Database (*.xcsv)
- Geocaching.com/EasyGPS (*.loc)
- Glopus (*.tk)
- Google Earth 3, 4, 4.2 and 5 incl. Network Links (*.kml and *.kmz)
- Google Maps URL
- GoPal Route (*.xml)
- GoPal Track (*.trk)
- GPS Tuner (*.trk)
- GPX XML 1.0 (*.gpx)
- GPX XML 1.1 with Garmin and trekbuddy Extensions (*.gpx)
- National Geographic Topo 3 to 4 (.tpo)
- Navigating POI-Warner (*.asc)
- Navigon Mobile Navigator 4 to 6 (*.rte)
- Navigon Mobile Navigator 6 Favorites (*.storage)
- Navigon Mobile Navigator 7 (*.freshroute)
- Magellan Explorist (*.trk)
- Magellan MapSend (*.wpt)
- MagicMaps Project (*.ikt)
- MagicMaps Tour (*.pth)
- Map&Guide Tourenplaner 2005 bis 2009 (*.bcr)
- Microsoft AutoRoute 2002 to 2006 (*.axe)
- NMEA 0183 Sentences (*.nmea)
- OziExplorer Route (*.rte)
- OziExplorer Track (*.plt)
- OziExplorer Waypoint (*.wpt)
- pilsit.logpos (*.itn)
- Route 66 POI (*.csv)
- TomTom Route 5 to 8 (*.itn)
- TomTom POI (*.ov2)
- Top50 OVL ASC/GeoGrid Viewer (*.ovl)
- Tour Exchange Format (*.tef)
- Training Center Database 1 (*.tcx/*.crs/*.hst)
- Training Center Database 2 (*.tcx)
- Tripmaster 1.4 to 2.4 (*.itn/*.gpx/*.kml)
- TTTracklog (*.itn)
- ViaMichelin (*.xvm)

Spezielle Garmin-Umwandlungen

Für die bei Outdoor-Usern beliebten Garmin-Geräte gibt es eigene Konversionsseiten:

`http://garmin.gps-data-team.com/conversion.php`

`http://www8.garmin.com/products/poiloader/`

`http://www8.garmin.com/products/communicator/`

Zusammenfassung

Statt Handarbeit an vielen einzelnen Adressen in Google-Earth und dann auch am Navigationsgerät trägt man die Geoposition einer Adresse ein einziges Mal in einer Adressdatenbank ein (mit Google-Earth und einer halb ausgefüllten Kml-Datei bekommt man alle Werte in einem Arbeitsschritt) und exportiert diese mit einem Visual-Basic-Programm als Kml-Datei in Google-Earth und danach durch eine Konversion als Gpx-Datei in ein Garmin-Navi.

Optimal wäre aber, dass die Eintragung einer neuen Adresse, etwa in Outlook, die Datenbank selbständig Kontakt mit dem Google-Server aufnimmt und die Geokoordinaten in die Felder `LAT` und `LNG` einträgt, sodass man sich um das Geokodieren nicht mehr kümmern müsste.

Eigentlich schade, dass die Hersteller von Navigationsgeräten zu diesen keine Hilfen zur Erfassung persönlicher Datenbanken anbieten.

Im Zuge der Versuche zu diesem Artikel war der Autor ziemlich überrascht, wie gravierend die Mängel eines so populären Navigationssysteme sind, wie bei dem für die Versuche verwendeten Garmin nüvi 550, dessen Besonderheit es ist, Straßennavigation und Offroad-Navigation in einem Gerät zu vereinen.

RouteConverter: Darstellung der Umwandlung der Kml-Datei (für Google-Earth) in eine Gpx-Datei (für Garmin-Navis)