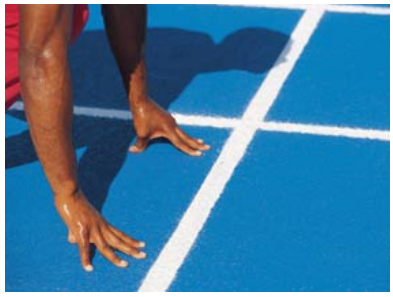
~~mapbox~~



mapbox community

Viele Dienste aus deiner Hand

1.1 Vom Satelliten auf die Tasse

Straßen, Städte, Seen und Satelliten, ja die Welt will vermessen sein. Zumal der Hunger nach Datenauswertung in diesem Bereich laufend wächst und sich auf diversen Medien präsentieren muss. Hier empfehlen wir mapbox.com mit seinem mächtigen API und starten zuerst mit dem Static API der Plattform.

„mapbox.com“ ist eine Programmierumgebung als Plattform mit Onlineservice, die eine frei nutzbare (volumenbasiertes Preismodell) und gut dokumentierte Programmierschnittstelle (API) für Kartendaten, Satellitenbilder, sowie analytischen Geodaten für Entwickler von Events, Webanwendungen und mobilen Geräten bereitstellt. Vor allem das Map Matching als unscharfe Suche via GPS und OpenStreetMap zeigt Interesse bei der GEO location und GEO Referenzierung.

Bei map-box als weiteren Zusatzdienst lassen sich sogar bunte wie auch selbst generierte Stadtpläne auf einer Leinwand, Tasse, Teller, T-Shirt, Visitenkarte, Tasche oder für Landkartenfetischisten auf Tapete oder Bettwäsche kaufen. Warum nicht mal ein T-Shirt wo drauf steht: Hier wohne ich.

Die Community ist groß und mit allen Kanälen und Medien gewaschen, Twitter z.b. wird viel genutzt, vor allem wenn's um Großanlässe geht: @Mapbox mit über 43k followers. Momentan bei der Tour de France 2016 sind dann standardmäßig mapbox datasets mit OpenStreetMap Karten zur Darstellung im Einsatz.

Ok., soweit so map, beginnen muss man mit dem Registrieren und Lösen eines access-tokens. Dieser public key pk.* lässt sich dann in den REST Aufruf wie folgt einbauen.

```
mapboxAPI = 'https://api.mapbox.com/geocoding/v5/mapbox.places/
%s.json?country=%s'+'&access_token=pk.eyJ1.....Beyhow'
```

Wie kann denn so ein Aufruf in seiner einfachsten Form in der Konsole aussehen:

```
Writeln(GetGeoInfo5('cologne','de',mapboxAPI))
```

Die beiden get Parameter `places` und `contry` sind im Aufruf als variabel `%s` codiert.
Bereits dieser Request liefert die wichtigsten Daten und Kennzahlen zur Standortangabe im JSON Format zurück. Somit hat man Zugriff auf freie Kartendaten, Koordinaten, sowie Distanzangaben mit Infos über das Routing einer Strecke. Die meisten Kartendaten sind im JSON, XML oder HTML Format bezugsbereit. Ein WSDL Endpunkt existiert als Web Service API mit low level Zugriff direkt aus einem Skript oder Rich-Client heraus.

https://www.mapbox.com/help/define-rest-api/

Die API ist effizient und flexibel genug, um weitere Parameter wie Distanz Sequenzen als Bsp. hinzuzufügen[1].

So lässt sich bereits ausführlich und Restful mit der URL interagieren, um mal schnell das Ergebnis zu testen:

```
http://api.mapbox.com/v4/mapbox.streets-satellite.html?
access_token=pk.ey.....C3NfLA#14/46.9659/7.4684
```

In Kombination mit Satelliten und Wetterdaten ist mit OpenWeatherMap ein guter Player gefunden. Es sind Datenpunkte von rund 43,000 Wetterstationen oder Sensoren verfügbar und die Prognosen basieren auf bekannten UTM Modellen.

Nun lasst uns eine erste Static Map programmieren. Als erstes hab ich die URL in eine wiederverwendbare Konstante gepackt:

```
Const
  mapboxAPIMap=
                'https://api.mapbox.com/v4/mapbox.satellite/16.95,47.45,5,0/%dx
%d.png128?access_token=pk.eyJ.....fLA#4/46.210/7.45';
```

Gehen wir die Punkte durch. Die Version als v4 ist bereits in der Schnittstelle erkennbar, in seiner einfachsten Form übergebe ich die Koordinaten als longitude 16.95 und latitude sowie einen zoom Parameter. Ein Bearing als 5 (Rotation) und Pitch als 0 (Perspektive) sind optional.

Static Maps haben den Vorteil, dass ich keine Plugins oder JavaScript benötige, vor allem bei Blogs oder mobilen Geräten mit Policies ein Effizienz- und Sicherheitsgewinn. Geliefert wird dann als Memorystream ein waschechtes PNG ohne Interaktivität eben oder bösen hidden Links.

[1] REST for Distance API (travel times) Forecasting

Aufgepasst noch bei den Koordinaten im degree Format in mapbox, hier ist der Längengrad `lon` (-180..+180) offensichtlich mit dem Breitengrad `lat` (-90..+90) vertauscht, Konvention ist eigentlich zuerst `lat` und dann `lon` zu parametrieren:

lat='46.9570806' lon='7.45880536873668' als N und E

Mit dieser REST-API Schnittstelle lassen sich natürlich externe Apps mittels HTTP-Methoden (vor allem GET, POST oder PUT) in jeden Server wie auch node.js integrieren.
Damit kann ich auch Teilinformationen auslesen und Listen von z.B. Livedaten, Distanzdaten, Objekt-Eigenschaften oder von Tags erstellen.
Die Static Map liefert mir nun Europa als Stream auf den Tisch:

mX4mapbox_ent2_europe.png

Weiter geht’s mit der GET Methode als Integration in einem Skript. Diese Methode dient als Funktionsmuster für viele Sprachen oder Frameworks und besitzt als Argumente die URL Parameter und eben den API-String selbst. Ein Android SDK wie Query Parameter sind auch dabei.

https://www.mapbox.com/api-documentation/#retrieve-a-static-map-from-a-style

In der API muss dann nur noch der eigene Standort / ID als Koordinaten verändert werden.
Die meisten Webmaster und Entwickler benötigen vermehrt Kartendaten als Mashup in einem PHP, Phyton oder maXbox Skript, dies hat mich dann eben zu diesem Tutorial bewegt. Meine Intention war einfach ein API zu finden, dass mir eine parametrisierbare Map liefert, die ich beliebig weiter-verarbeiten und platzieren kann und genau das erfüllt mapbox mit dem Static API. Der GET Methode wird in der maXbox ein zuvor brav allozierter Memory Stream übergeben, die mir die JSON Daten in diesem Fall als PNG zurück liefert:

```
function GetGeoInfoMap2(const location, country: string; asize: integer;
                        const mapboxAPIMap: string): string;
var
   pngStream: TMemoryStream;
begin
   pngStream:= TMemoryStream.Create;
   try
     HttpGet(Format(mapboxAPIMap,[asize,asize]), pngStream);
    except
     IHTTP.Get1(Format(UrlGeoLookupInfo2,[location]),pngStream);  // try backup server.
    end;
  with TLinearBitmap.Create do try
     pngStream.Position:= 0;
     LoadFromStream2(pngStream,'PNG');
     paintToCanvas(getForm(700,700).canvas, Rect(5,5,Width,Height),false);
  finally
    Dispose;
    Free;
    pngStream.Free;
 end;
end;
```

Das Bitmap wird dann direkt mit `paintToCanvas()` aufs Form gerendert.
Für eine weitere App die dynamisch zoomen will hab ich einfach die API als Konstante ausgewechselt und kann eben den zoom Parameter laufend nach-führen:

```
mapboxAPIMap4=
'https://api.mapbox.com/v4/mapbox.satellite/16.95,47.45,15,0/%
dx%d.png128?access_token=pk.eyJ.....fLA#4/';
```

Ideologisch ist mapbox wie OpenWeatherMap inspiriert von OpenStreetMap und Wikipedia, die Information und Dienste allgemein und kostenlos zur Verfügung stellen, ein kommerzieller Nutzen kostet dann.

Das Skript zum Testen ist online verfügbar:

http://www.softwareschule.ch/examples/711_geo_satellite_mapbox.txt

Der Aufruf zeigt dann folgendes Bild gezoomt:

mX4mapbox_ent2.png

1.2 Auch in mapbox Studio

Es gibt in der mapbox community auch Entwicklertools die man als basemap für Styles und Tiles einsetzen kann.
Mapbox Studio (online) ist einfach gesagt eine Software zur Erstellung von Kacheln als Vektoren oder CSS Templates, die als Hintergrundkarten oder Weltkarten für Webkarten benutzt werden können. Mögliche Typen sind einfache Kacheln als Tiles sowie Raster- und Vektorkacheln die sich dann als Karten und Tilesets in einer weiteren Leaflet und OpenLayers Karten-anwendung einbinden lassen. Exemplarisch sei hier eine Erweiterung der Satellitenkarte mit Straßen- oder Länderinformation demonstriert, respektive dargestellt:

```
mapboxAPIMap=  'https://api.mapbox.com/v4/mapbox.streets-satellite/7.45,46.95,15,0,5/%dx%d.png128?access_token=pk.eyJ1.....LA#4';
```

Wie man sieht hab ich die URL mapbox.satellite zu mapbox.streets-satellite erweitert, als sogenannte classic map IDs:

- mapbox.streets
- mapbox.light
- mapbox.dark
- mapbox.satellite
- mapbox.streets-satellite
- mapbox.wheatpaste
- mapbox.streets-basic
- mapbox.comic
- mapbox.outdoors
- mapbox.run-bike-hike
- mapbox.pencil
- mapbox.pirates
- mapbox.emerald
- mapbox.high-contrast

Übrigens sind auch Satelliten nicht vor Softwareupdates befreit; nur gewinnt hier der Begriff Upload eine andere Dimension.
In mapbox Studio lassen sich auch Shapefiles (Esri vector) importieren und so mit anderen Größen im Markt interagieren. Es existiert deshalb ein Esri Connect zu ArcMap und ArcView. Mapbox bezeichnet dies auch interessanterweise als missing link.
Ein Grund für die große Zahl an Mobile-Location-Anwendungen, die es mittlerweile im Bereich Freizeit, Tourismus und Outdoor heute am Markt

gibt, sind die für alle drei großen Betriebssysteme verfügbaren Software Development Kits (SDKs) und Maps APIs, mit denen Entwickler und Maker interaktive aber auch statische Kartenfunktionen einfach in eigene mobile Anwendungen integrieren können.
Bei den Angeboten findet man meistens OpenStreetMap als fundierte und globale Datengrundlage für Karten- und Routenplaner-Anwendungen wie eben mapbox, maps with me oder CloudMade zeigt.

Wie siehts aber mit dem Urheberrecht aus?
Geodaten sind durch das Urheberrecht als geistiges Eigentum soweit geschützt. Der Schutz bezieht sich sowohl auf die schöpferische Leistung bei der Datenmodellierung und der Darstellung als auch auf die Investition in die Herstellung einer Geodatenbank.
Der Erzeuger von Geodaten wie mapbox kann demnach bestimmen, welches Lizenzmodell für die Nutzung seiner Daten gelten soll:

https://www.mapbox.com/pricing/

Mit einer GPS-Navigation lässt sich schlussendlich eine Automatisierung nochmals drastisch vereinfachen. Die die UX, neudeutsch User Experience, deutlich bequemer wird, da nun die Koordinaten der mobilen App direkt zum API wandern und Karte wie Position auf dem Schirm erscheinen; die Schnittstelle dazu findet man unter:

http://open.mapquestapi.com/nominatim/

http://www.softwareschule.ch/examples/509_GEOMap3.txt

```
procedure GetMapScript(C_form,apath: string; const Data:string);
var encURL: string;
    mapStream: TMemoryStream;
begin
  encURL:= Format(UrlMapQuestAPICode2,[c_form,HTTPEncode(Data)]);
  mapStream:= TMemoryStream.create;
  try
    HttpGet(EncURL, mapStream);    //WinInet
    mapStream.Position:= 0;
    mapStream.Savetofile(apath)
    OpenDoc(apath);
  finally
    mapStream.Free;
  end;
end;
```

Wenn nur ein Ort „gemappt“ wird, genügt die folgende API:

```
UrlMapQuestAPICode2='http://open.mapquestapi.com/nominatim/v1/search
.php?format=%s&json_callback=renderBasicSearchNarrative&q=%s';
```

1.3 Fazit

Mit der mapbox Open Source Plattform hat man ein Toolset und diverse REST APIs zugleich, die man schlicht zur Next Generation zählen darf. Immer mehr Apps der Geoinformationstechnologie basieren auf Basis von HTML5, CSS3 und JavaScript im Umfeld von sog. Responsive Design.
Mit den modularen APIs, vor allem Static, Geocoding, Distance und Map Matching ist man breit aufgestellt, was auch die plattformunabhängige Entwicklung fördert.
Auch mit dem durchdachten Upload API wird man mit dem Stagen von Daten in eine Cloud wie S3 nicht alleine gelassen.
Das eigene Tool mapbox Studio hilft beim Editieren von neuen Layern oder ausgefeiltem, interaktivem Kartenmaterial. Das Tool vereint Flexibilität, moderner Look und einfache Bedienung in einem, in der Basic Version sogar kostenlos.
Für mich ein Highlight ist der Satellite Layer kombiniert mit dem Static API. Mapbox kann auf unterschiedliche Auflösungen in unterschiedlichen Gebieten auch in Detailkarten mit Zoom-stufen reagieren. Man darf teilweise DigitalGlobe-Bilder nutzen und erweitern, die genaue Rechtslage ergibt sich aber nicht aus der Veröffentlichung.

Mit Mapbox lassen sich also Karten auf einfache Weise in Anwendungen oder Artikel integrieren. Die Verwendung setzt ein Account voraus und der Preis ist volumenbasiert. Mit der App lassen sich nicht nur Orte, sondern auch ganze Strecken und Areale deutlich machen um der Erde ein neues Antlitz zu verschaffen.

Feedback @ max@kleiner.com
Literature: Kleiner et al., Patterns konkret, 2003, Software & Support

https://www.mapbox.com

www.openweathermap.com

http://www.programmableweb.com/api/mapbox

https://github.com/maxkleiner/maXbox3/releases

```
if isInternet then begin
   //geocode:= GetGeoCode('xml','','bern switzerland', false);
   geocode:= GetGeoCode('xml','','werzer poertschach austria', false);
    writeln(GetGeoInfo5('poertschach','',mapboxAPI))
     writeln('geocode: '+geocode)
     latf:= GEOCoord2Point(geocode).x
     longf:= GEOCoord2Point(geocode).y
     PrintF('lat: %.2f - lon: %.2f ',[latf,longf])
     zoomf:= 16.1
   //2 APIs: mapboxAPIMap - no names ; mapboxAPIMapStreet - with names

   GetGeoInfoMap4save(latf,longf,zoomf,650,mapboxAPIMapStreet,
                                    ExePath+'mX4mapbox_poertschach.png')

end else writeln('No Internet Connection found');
```

mX4mapbox_sat.png

http://bleiwüsten.de/work/mapbox-interaktive-karten-erstellen/