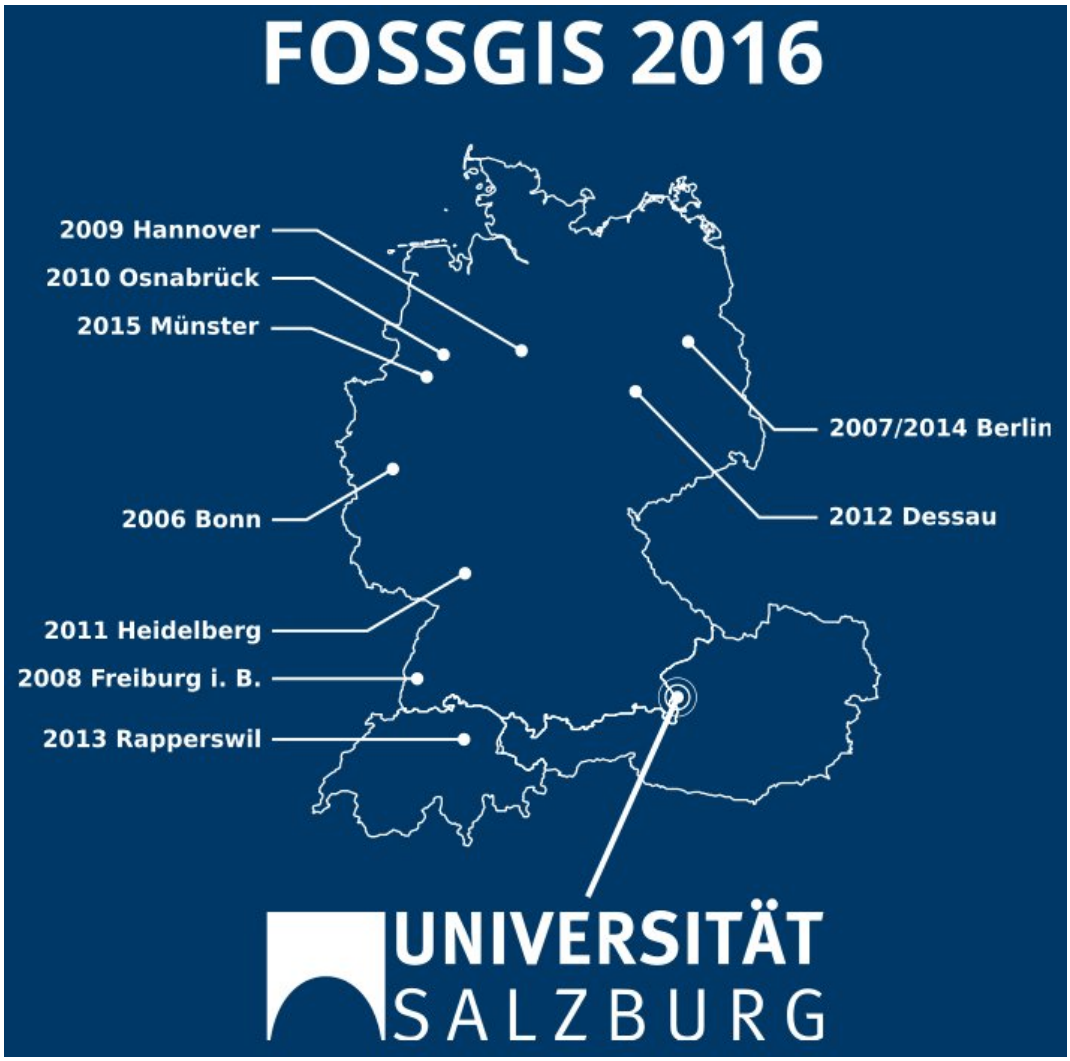
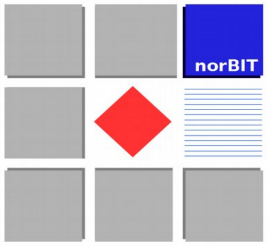
Anwenderkonferenz für Freie und Open Source Software für Geoinformationssysteme

Salzburg 04. - 06. Juli 2016



Silbersponsoren:



Bronzesponsoren:

ISBN 978-3-00-053437-9

Herausgeber: FOSSGIS e.V.
http://www.fossgis.de

Inhaltsverzeichnis

Was sind "Open" Source, Data und Standards - und wie funktioniert das?

Arnulf Christl, Frederik Ramm, Dominik Helle

Open Source hat viele Facetten - und es ranken sich inzwischen ebenso viele Mythen darum. Was davon richtig ist und was nicht stellen wir in einer kurzen Einführung zusammen. Was sind Open Data und Open Standards, welche Gemeinsamkeiten gibt es und wo unterscheiden sie sich. Der Vortrag richtet sich an alle, die mit Open Source, Open Data oder Open Standards bisher noch wenig Kontakt hatten und die Grundlagen verstehen möchten.

Open Source ist auf der einen Seite ein Entwicklungsmodell und auf der anderen ein Lizenzmodell. Zusammen bilden sie eine Kultur offener Entwicklungsgmemeinschaften, die höchst effektiv arbeiten. Diese Kultur ist um ein Vielfaches effektiver, als proprietäre Modelle es je sein können. Ein einfaches Beispiel: Das Betriebssystem des Herstellers Apple basiert auf dem Open Source Unix FreeBSD. Es gibt halt einfach nichts besseres, und es selbst herzustellen wäre unendlich teuer, das hat sogar der hyper-proprietäre Hersteller Apple eingesehen.

Der Vortrag stellt die Geschichte der Entwicklung von Open Source vor und geht auf wichtige Grundlagen ein. Ziel des FOSSGIS e.V. und der OSGeo ist die Förderung und Verbreitung freier Geographischer Informationssysteme (GIS) im Sinne Freier Software und Freier Geodaten. Dazu zählen auch Erstinformation und Klarstellung von typischen Fehlinformationen über Open Source und Freie Software, die sich über die Jahre festgesetzt haben.

GIS in der Cloud - Schönwetterwolke, Gewitter oder reiner Dunst?

Till Adams

Das WebGIS-Systeme Desktop-basierten an Funktionalitäten zunehmend ebenbürtig werden, ist fast schon ein alter Hut. Doch wie sieht es auf der Serverseite aus?

Ist die Verlagerung von GIS-Architekturen in die Cloud wirklich die allseits Schönwetter-Machende, problemlos skalierbare Alternative zur klassischen Anmietung eines Servers, wo sind Fallstricke, wie sieht es mit den Kosten aus und überhaupt, wo landen meine Daten eigentlich?

Der Vortrag beleuchtet vor dem Hintergrund technischer und rechtlicher Aspekte praktische Vor- und Nachteile des Cloud-Hostings und zeigt anhand von konkreten Beispielen auf, wann sich die Cloud als Hostingadresse für Kartendienste und WebGIS-Plattformen wirklich lohnt.

Als Beispiele werden auf OSM- und anderen Freien Daten basierende weltweite Kartendienste sowie die auf SHOGun2 basierende WebGIS-Plattform „MapMavin“ vorgestellt. Für beide wurde der Betrieb auf einem Cloud-Server geprüft, doch nur für MapMavin dann auch tatsächlich realisiert.

Freie (Geo-)Daten mit Freier (Geo-)Software - oder: wie kommen Geodaten zum Nutzer?

Dr. Markus Neteler, Till Adams

Auch wenn sich der Titel des Talks wie ein vereinfachter Werbeslogan anhört, entpuppt sich der dahinterstehende Inhalt bei näherer Betrachtung als tatsächlich spannende Fragestellung. Im Jahr 2014 wurde der erste Erdbeobachtungs-Satellit "Sentinel-1A" des EU Erdbeobachtungsprogramms "Copernicus" gestartet und sendet seit einiger Zeit kontinuierlich Fernerkungungsdaten zur Erde, gefolgt von "Sentinel-2A" in 2015. Dass über das Copernicus Programm mehr und mehr Daten verschiedenster Spektralbereiche zur Verfügung stehen, wird in der Fernerkundungs-Community natürlich wahrgenommen, verschiedenste Fachbereiche arbeiten bereits an der Nutzung der Daten. An potentiellen Nutzern und Anbietern der Geoinformations-Community fährt dieser Zug allerdings nahezu unbeachtet vorbei. Dabei sind sämtliche durch Copernicus bereit gestellte Daten Freie (Geo-)Daten und dürfen daher auch kommerziell aufbereitet angeboten und verarbeitet werden.

Dass man mit Hilfe von Prozessierung aus Fernerkundungsdaten neue räumliche Erkenntnisse mit Nutzen für verschiedenste Bereiche der Geo-Information, sei es Landnutzung, Qualitätssicherung von Geodaten, Bodenfeuchte oder der Oberflächenstruktur des Landes, um nur einige Beispiele zu nennen, gewinnen kann, ist auch allgemein in der GIS-Welt bekannt. Was derzeit fehlt, oft auch inzwischen aufgrund der schieren Datenmenge, ist ein einfacher Zugang zu diesen Daten und vor allem zu den daraus gewonnenen Informationen.

Der Vortrag stellt einen Ansatz vor, wie man diese neuen Freien Geodaten zeitnah mit der Freien Software GRASS GIS prozessieren und automatisiert als einen standardisierten OGC-Web-Service mittels GeoServer und MapProxy bereit stellen kann. Erweitert man die Architektur um Komponenten, die es dem Nutzer erlauben, auf Basis von bestimmten Algorithmen automatisiert und ohne technische Kenntnis der Software selber aktuelle Informationen zu generieren, so erweitert man den Nutzerkreis des Copernicus-Programms und verbindet gleichzeitig zwei Welten, die sich bisher in vielen Bereichen in friedlicher Ko-Existenz nebeneinander entwickelt haben: Die GIS- und die Fernerkundungs-Community. Auch die hierfür erforderlichen Schnittstellen existieren seit Manifestierung der WPS-Spezifikation des OGC längst. Damit entfällt die Hürde des mühseligen Datensammelns, langer Download-Zeiten, die Notwendigkeit des Vorhaltens von Rechen- und Speicherkapazitäten sowie der erforderlichen Kenntnis und Verfügbarkeit von Expertenwerkzeugen, denn das Laden eines standardkonformen WMS in eigene Werkzeuge ist für viele GIS-Nutzer seit Jahren gelebte Realität. Der Vortrag rundet sich durch konkrete Nutzungsbeispiele sowie einem praktischen Beispiel, die Vorstellung des Workflows und der Architektur ab.

Koautoren: Till Adams, Carmen Tawalika, Hinrich Paulsen

Schadstoffeinleitungen in Kanäle und Gewässer verfolgen

Gerhard Genuit

1 Einführung

Wenn die Wasserqualität in einem Kanal oder Fließgewässer überprüft werden soll, wird in der Regel eine Stichprobe genommen. Häufig finden Einleitungen aber nicht kontinuierlich statt, sodass es dem Zufall überlassen bleibt, ob man diese mit der Probe erwischt oder nicht. Klarheit kann letztendlich nur der Einsatz eines automatischen Probenehmers mit einem entsprechend hohen apparativen, personellen und finanziellen Aufwand liefern.

Die Lücke zwischen Stichprobe und einer großen Anzahl automatisch gezogener Proben kann der Einsatz von Passivsammlern teilweise schließen. Diese werden in den Kanal bzw. das Gewässer eingehängt und sammeln die zu untersuchenden Substanzen über einen bestimmten Zeitraum. Danach werden sie entnommen und ausgewertet. Wegen der komplexen Matrix bietet sich hierfür im Abwasserkanal die Untersuchung von Biofilm (norddeutsch: Sielhaut) ganz besonders an.

2 Methode

2.1 Funktionsweise

Die Sielhaut ist ein grau-brauner Biofilm aus Mikroorganismen (Bakterien und Pilze), der sich auf der Innenseite von Abwasserleitungen (Sielen) bildet und je nach mikrobieller Zusammensetzung oftmals eine “fettige bis seifige” Konsistenz hat. Sie ist in der Lage, Schadstoffe aus dem Abwasser aufzunehmen und anzureichern (Memory-Effekt).

Sielhautuntersuchungen werden zur Ermittlung von schwermetallhaltigen Einleitungen seit der Veröffentlichung der Dissertation von Gutekunst vor über 20 Jahren vielfach eingesetzt [1]. Auch in Bielefeld konnte schon Anfang der 90er Jahre eine massive Zinkeinleitung regional eingegrenzt und dann durch intensive Suche einer Firma konkret zugeordnet werden. Die illegale Abwassereinleitung konnte durch direkte Beprobung des Abwassers weiter belegt und unterbunden werden. Durch das schnelle Auffinden der Schadstoffquelle konnte seinerzeit der Kläranlagenbetrieb sichergestellt werden. Auch die Eignung für organische Schadstoffe ist bereits mehrfach nachgewiesen [2].

2.2 Datenhaltung

In unserem „Anlagen- und Indirekteinleiterkataster“ (AUIK) werden sowohl die Daten zur Sachbearbeitung, als auch die Dokumentation der Messpunkte und Messwerte des Sielhaut Kontrollprogramms gespeichert. Dieses Kataster ist eine Eigenprogrammierung und in Java realisiert. Datenhaltung erfolgt in einer PostgreSQL/PostGIS. AUIK ist so flexibel aufgebaut, dass es leicht um beliebige weitere Objekte, die mindestens über einen Standort und einen Betreiber definiert sind, erweitert werden kann. In 2015 wurde eine Representational State Transfer (REST)-Schnittstelle zu der NRW Landesdatenbank ELKA programmiert und für 2016 ist die Erweiterung um Daten der Direkteinleitung in Oberflächengewässer und Sonderbauwerke geplant. Das Einleiterkataster ELKA umfasst sechs vormals separat bestehende DV-Anwendungen aus den Bereichen Niederschlagswasser, Industrieabwasser und Kommunalabwasser.

3 Auswertung

3.1 Hintergrundwerte

Um Messwerte besser bewerten zu können, wurden erstmals im Jahre 2001 Hintergrundwerte für eine Vielzahl von Elementen von Kintrup und Wünsch in der Korrespondenz Abwasser veröffentlicht [3]. Die Hintergrundwerte wurden durch Untersuchung von Biofilmproben aus überwiegend häuslichen aber auch industriellen Entwässerungsgebieten ermittelt. Diese Werte wurden in Bielefeld viele Jahre zur Beurteilung der Analysen verwendet, indem die gefundenen Messwerte als Verhältniszahl zu diesen Hintergrundwerten angegeben wurden. So lassen sich Elemente, welche in völlig unterschiedlichen Konzentrationsbereichen vorkommen, wie z. B. Zink und Cadmium, in einem Diagramm darstellen. Dabei hat sich gezeigt, dass für ein dauerhaftes Überschreiten des dreifachen Hintergrundwertes in der Regel eine Ursache gefunden werden kann.

Mittlerweile ist unsere Datenbasis aber deutlich umfangreicher als die, die 2001 Kintrup und Wünsch zur Verfügung stand, so dass wir nun die eigenen Daten statistisch ausgewertet haben und unsere selbst ermittelten Werte für die Auswertung der Analysenergebnisse benutzen. Für die Analyse der Werte haben wir zunächst die beschreibenden Größen: Anzahl der Messwerte, Minimum, erstes Quartil, Median, Mittelwert, drittes Quartil und Maximum ermittelt.

Tabelle 1: Verschiedene Hintergrundwerte

	Cd	**Cr**	**Cu**	**Hg**	**Ni**	**Pb**	**Zn**
Kintrup und Wünsch	1,2	27	180	2,2	23	42	880
Bielefeld Firmenkontrolle	0,7	30	280	0,2	22	34	800
Bielefeld Routineüberwachung	0,6	24	240	0,7	17	34	800

Der Vergleich unserer in 2013 ermittelten Hintergrundwerte mit denen von Kintrup und Wünsch aus dem Jahre 2001 zeigt, dass unsere Werte mit Ausnahme von Kupfer niedriger, aber durchaus in der gleichen Größenordnung sind. Auch sind die Unterschiede zwischen Routine- und Firmenkontrollpunkten nicht erheblich. Hier muss aber berücksichtigt werden, dass es z. T. deutlich höhere Werte gibt, die sich bei der Auswertung über den Medianwert und die Dichteverteilung nicht adäquat niederschlagen.

3.2 Raumbezug

Über die Standorte der Anlagen und Lage der Messpunkte haben die Sachdaten einen Raumbezug und können sowohl stadtweit in unserem „Online Kartendienst“ auf Basis von MapBender als auch in lokalen QGIS Projekten dargestellt und mit Karten anderer Dienststellen wie z. B. dem Kanalplan der Stadtentwässerung verschnitten werden. Die Daten hierfür liegen entweder direkt in unserer PostgreSQL/PostGIS Datenbank oder werden von unserem Katasteramt über standardisierte Schnittstellen wie WMS und WFS zur Verfügung gestellt.

Mithilfe eines modifizierten QGIS Plug-Ins (zoom-to-point) ist die Navigation zu Standorten per Auswahl einer Straße und Hausnummer möglich. QGIS lässt sich aber auch direkt aus dem Kataster heraus starten und bekommt dann die Koordinaten des selektierten Standortes als Umgebungsvariable

übergeben. Außerdem gibt es in dem Kataster die Möglichkeit, im QGIS aufgenommene Koordinaten über die Zwischenablage in einen Datensatz einzufügen.

3.3 Fließwegverfolgung mit QGEP

Für die Auswertung der Messwerte ist die Verfolgung der Fließwege von einem Messpunkt im Kanal- und Gewässernetz stromauf- und -abwärts von ganz besonderem Interesse. Diese Funktion stellt das QGEP Plug-In über eine entsprechende Abfrage unserer Kanaldaten zur Verfügung.

Das QGEP-Projekt hat zum Ziel auf Basis von QGIS eine Fachschale für die Bereiche Abwasser und GEP (Genereller Entwässerungsplan) zu erstellen. Da QGEP vermutlich eine der ersten größeren Fachschalen auf Basis von QGIS ist, müssen einige Probleme im QGIS-Kern gelöst werden. Diese Entwicklungen kommen dann der generellen Fachschalenentwicklung unter QGIS zugute.

Umgesetzt wurde auch bereits eine modellbasierte Netzverfolgung (im Gegensatz zu vielfach üblichen grafischen Netzverfolgungen) auf Basis von NetworkX (Python) und eine dynamische Profilgenerierung (interaktiv) um Gefälle und Dimensionen der Abwasserbauwerke im Kontext der Geländeoberfläche darzustellen. Für die Darstellung des interaktiven Kanalprofils kommt d3.js innerhalb eines Browser-Bereichs (Webkit) zum Einsatz [4].

4 Fazit und weiteres Vorgehen

Die Methode der Biofilmuntersuchung stellt für den Abwasserkanal eine einfache und effiziente Möglichkeit dar, unerlaubte Einleitungen zu lokalisieren. Eine Übertragung auf Fließgewässer ist nicht ohne eine Modifikation möglich, da hier nicht genug Substrat für die Bildung eines ausreichenden Biofilms vorhanden ist. Der Einsatz von Filtervliesmatten, die mit einem Standard-Nähragar beladen sind, scheint aber durchaus verwertbare Ergebnisse auch für eine substratarme Matrix zu liefern. Wenn in den Kanaldaten die Schächte und zu jeder Haltung ein Anfangs- und ein Endschacht gespeichert sind, kann die Funktion der Fließwegverfolgung des QGEP-Projektes genutzt werden.

Kontakt zum Autor:

Dipl.-Ing. Gerhard Genuit

Stadt Bielefeld

August-Bebel-Straße 75-77

33602 Bielefeld

0521/51-2832

Gerd.Genuit@bielefeld.de

Literatur

[1] *GUTEKUNST, B.:* Sielhautuntersuchungen zur Einkreisung schwermetallhaltiger Einleitungen. Doktorarbeit, Institut für Siedlungswasserwirtschaft, Universität Karlsruhe, 1988.

[2] *KINTRUP, J., WÜNSCH, G.:* Multielementanalytik der Sielhaut: Optimierte Identifizierung von Schwermetalleinleitern. Korrespondenz Abwasser, 48, 1068-1073, Hennef 2001.

[3] *GENUIT, G., BLOCK, M.:* Ermittlung von Einleitungen PFT-haltigen Abwassers durch Untersuchung der Sielhaut, Gewässerschutz – Wasser – Abwasser, Band 217, 2009

[4] *NEUMANN, A., KUHN, M.:* Das QGEP Abwasser Projekt, FOSSGIS-Konferenz, Rapperswil, 2013

Neues in Metador 2.1

Axel Schaefer

Metador2 ist eine OpenSource Lösung zum einfachen Erstellen und Bearbeiten von Metadaten. Dabei unterstützt Metador2 die Aufnahme von Metadaten gemäß der INSPIRE Technical Guidelines und der Richtlinien der GDI-DE, kann aber auch einfach an völlig unterschiedliche und beliebige Metadatenprofile angepasst werden. Die Webanwendung bietet Importfunktionen, wie z.B. aus WMS-Capabilities an und kann die Metadaten als XML, PDF oder HTML exportieren. Metador2 bringt selbst keine CSW-Schnittstelle mit, sondern wird dafür durch CSW-Software wie Geonetwork oder deegree erweitert.

Metador 2.1 bringt ein neues Plugin-System mit, das die Erweiterung der Software um neue Funktionen einfacher und übersichtlicher macht. Plugins können dabei u.a. unterschiedliche Metadatenprofile sein. Während im INSPIRE und GDI-DE-Kontext die Metadatenprofile recht ähnlich sind, können z.B. für interne Metadaten unterschiedliche Profile für unterschiedliche Datentypen genutzt werden. Ist ein Metadatenprofil in einem Plugin umgesetzt, können auch weitere, von diesem abhängige Plugins entwickelt werden, beispielsweise für den Import von Metadaten in dieses Profil. Eine weitere Einsatzmöglichkeit von Plugins kann die Anpassung des Themas sein, d.h. der Farben, des Logos, etc.

Der Vortrag stellt die Neuerungen in der kommenden Version 2.1 vor, mit Live-Beispielen.

Metador ist verfügbar unter: https://github.com/WhereGroup/metador2/

Metador2 ist eine OpenSource Lösung zum einfachen Erstellen und Bearbeiten von Metadaten. Metador 2.1 enthält ein neues Plugin-System, mit dem beispielsweise unterschiedliche Metadatenprofile einfacher und übersichtlicher erstellt und mit Import- und Exportfunktionen unterstützt werden können. Der Vortrag stellt die Neuerungen in der kommenden Version 2.1 vor, mit Live-Beispielen. Metador ist verfügbar unter: https://github.com/WhereGroup/metador2/

Neue Funktionen in QGIS 2.10 – 2.16

Dr. Marco Hugentobler

Releases

QGIS hat einen Releasezyklus mit drei Versionen pro Jahr, wobei eine davon ein sogennanter long term release (LTR) ist. Ein LTR wird während einem Jahr mit Bugfixes versorgt. Seit der letzten FOSS-GIS Konferenz in Münster sind drei neue Versionen herausgekommen:

- 2.10 am 26.06.2015
- 2.12 am 23.10.2015
- 2.14 am 26.02.2016

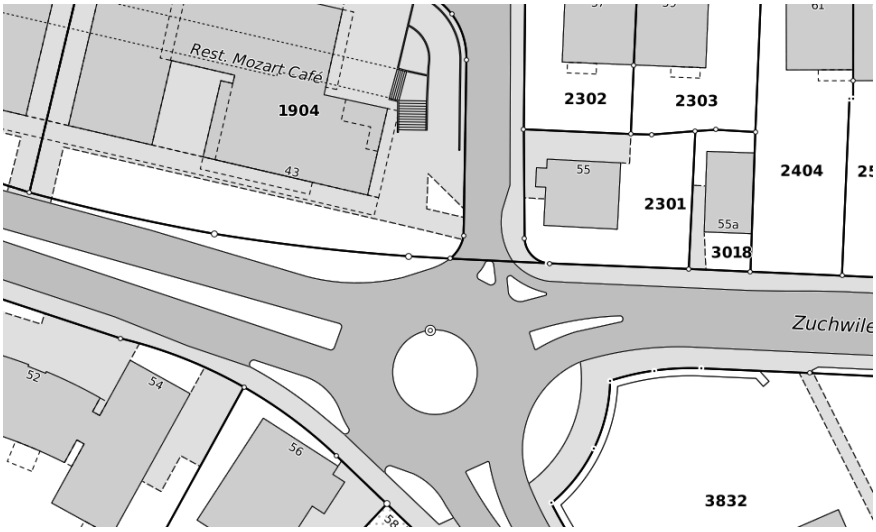
Die Version 2.16 wird in wenigen Tagen herauskommen. Die Anzahl neuer Funktionen, welche in diesen vier Versionen dazugekommen ist, ist sehr gross. Nachfolgend wird eine Auswahl vorgestellt, die vollständige Beschreibung aller neuer Funktionen ist in den Changelogs des QGIS Projekts zu finden ([2], [3], [4]).

Neues Geometriemodell (2.10)

Für die Version 2.10 wurde der Geometriecode in QGIS komplett erneuert. Neu werden alle Geometrietypen aus dem ISO-Standard SQL-MM-Part3 unterstützt, insbesondere auch diejenigen, welche Kreisbögen enthalten können (Circular String, Compound Curve, Curve Polygon, Multi Curve, Multi Surface, Geometry Collection). Ausserdem werden Z- und M-Koordinaten konsequent unterstützt.

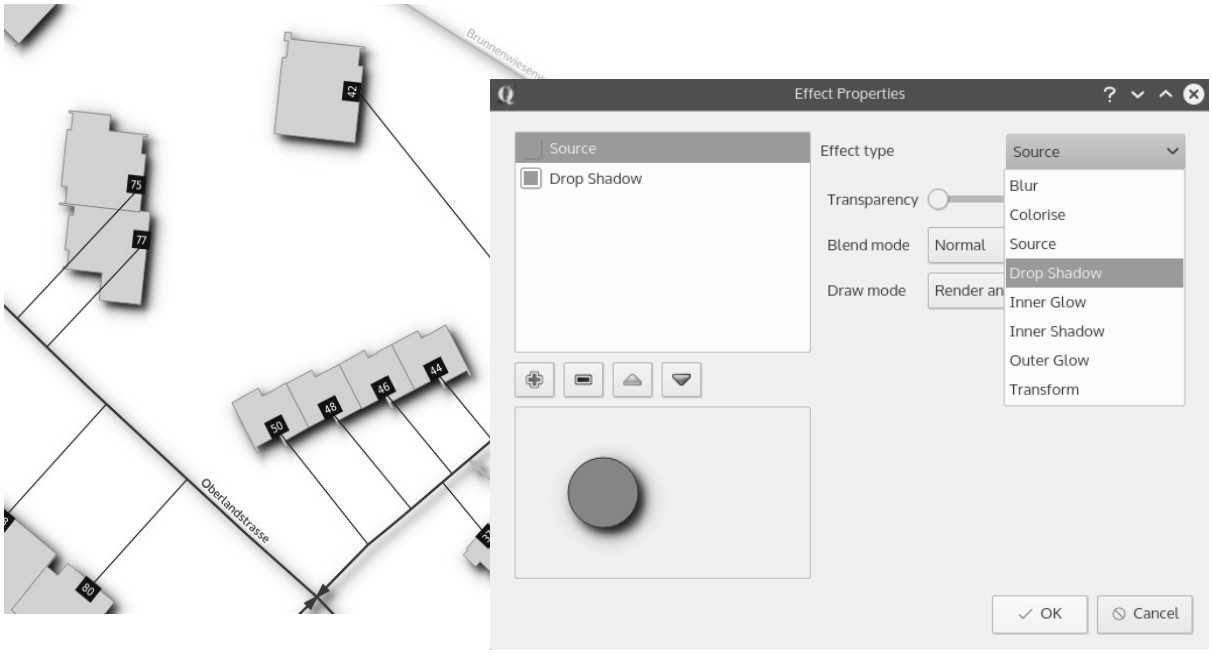
Zur Erfassung von kreisbogenbasierten Geometrien gibt es zwei neue Editierwerkzeuge. Mit dem einen können Kreisbögen durch Anfangspunkt, Punkt auf dem Kreisbogen und Endpunkt erfasst werden. Mit dem anderen kann ein Kreisbogen durch Anfangspunkt, Endpunkt und Radius definiert werden. Aus den vier möglichen Kombinationen kann mit der Maus ausgewählt werden.

Da die Geometriebibliothek GEOS ([1]) keine Kreisbögen unterstützt, werden die Kurven bei komplexen Geometrieoperationen, wie z.B. Intersect, Union, Difference, etc. intern segmentiert. Längen und Flächen werden aber direkt aus den Kreisbögen berechnet.



Live Layereffekte (2.12)

Schöne kartographische Produkte lassen sich mit den Layereffekten machen. Dabei wird, ähnlich einem Bildbearbeitungsprogramm, beim Rendern der Karte ein pixelbasierter Filter angewendet.



Innerer / äusserer Schattenwurf sowie innere / äussere Beleuchtung können beliebig kombiniert werden. Die Effekte sind sowohl auf die einzelne Symbolebene sowie auf ein ganzes Symbol anwendbar.

Authentifizierung (2.12)

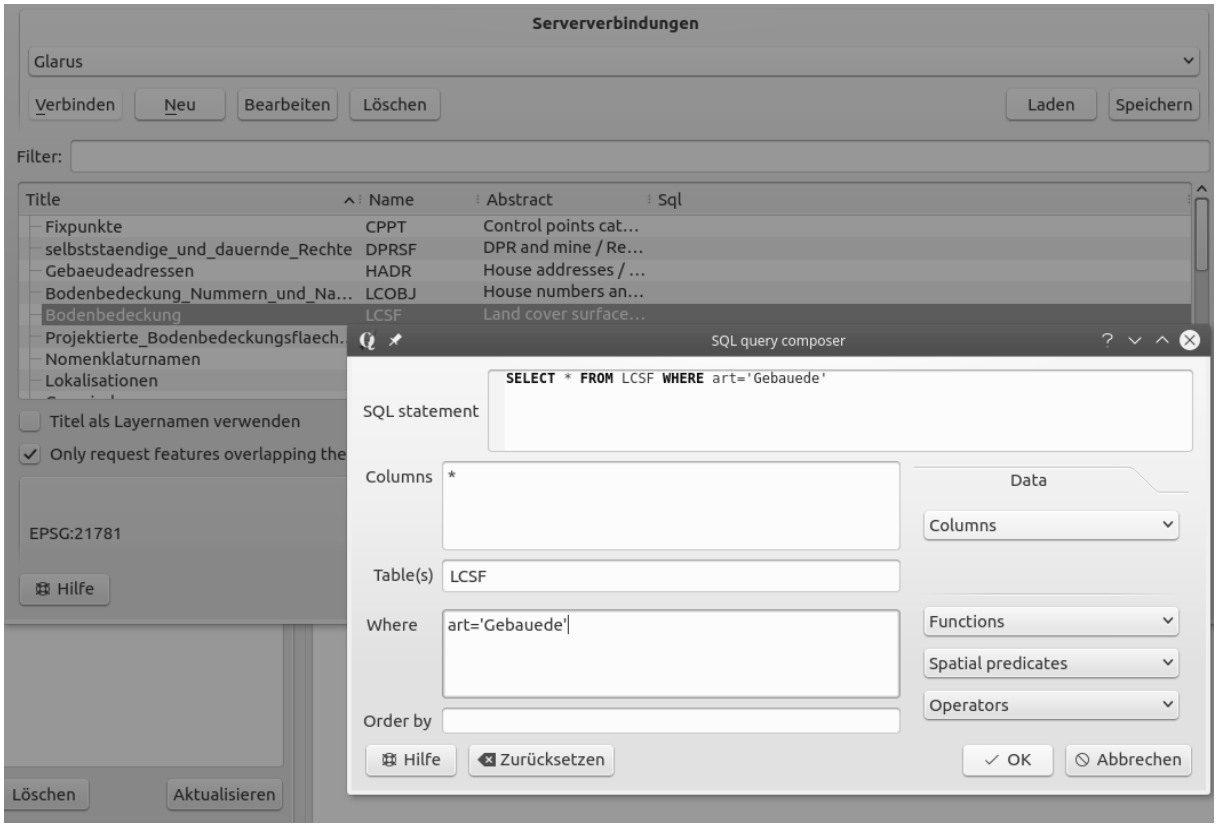
Bis und mit 2.10 konnte man Passwörter für Datenbank und Webservices entweder vor dem Verbinden eingeben oder unverschlüsselt in den Einstellungen bzw. dem Projektfile speichern. Ersteres ist nicht sehr komfortabel und letzteres nicht sehr sicher. Daher gibt es jetzt einen neuen Authentifizierungsmechanismus. Der Benutzer gibt einmal ein Masterpasswort ein, und QGIS holt sich dann die Passwörter für die einzelnen Verbindung aus einer lokalen Datenbank. Da die Passwörter in der Datenbank verschlüsselt sind, erhöht dies auch die Sicherheit.

Aggregatfunktionen (2.16)

Die sogenannten Expressions sind Rechenausdrücke, welche dynamisch ausgewertet werden und an mehreren Orten in QGIS vorkommen, z.B. im Feldrechner bei der Berechnung neuer Attribute oder bei der regelbasierten Symbolisierung. Da es viele vordefinierte Expressionfunktionen gibt und es recht einfach ist, eigene Funktionen zu machen, sind Expressions sehr mächtig. Bisher wurden die Expressions ausschliesslich pro Objekt ausgewertet. Es war daher nicht möglich, den Durchschnitt eines Attributes im Rechenausdruck zu verwenden. In 2.16 wird es neu Aggregationsfunktionen geben, mit denen unter anderem Mittelwert, Minimum, Maximum, häufigster Wert, seltenster Wert, etc. einer Spalte berechnet werden können.

Unterstützung für WFS 2.0 (2.16)

Der Code des WFS-clients wurde grundlegend überarbeitet. Es werden nun auch die Versionen 1.1 und 2.0 unterstützt. Zudem gibt es für die Erstellung des WFS-Filters einen komfortablen SQL-Editor.



Kontakt zum Autor:

Dr. Marco Hugentobler
Sourcepole
Weberstrasse 5
CH-8004 Zürich
marco.hugentobler@sourcepole.ch

Literatur

[1] GEOS(2016): https://trac.osgeo.org/geos/

[2] QGIS(2015)a: Changelog for QGIS 2.10, http://qgis.org/en/site/forusers/visualchangelog210/

[3] QGIS (2015)b: Changelog for QGIS 2.12, http://qgis.org/en/site/forusers/visualchangelog212/

[4] QGIS (2016)c: Changelog for QGIS 2.14, http://qgis.org/en/site/forusers/visualchangelog214/

Neues von MapProxy

Dominik Helle und Oliver Tonnhofer, Omniscale GmbH & Co. KG

Ende 2008 wurde der Grundstein für die Software MapProxy gelegt. Im März 2010 wurde MapProxy von den Entwicklern, der Firma Omniscale, im Rahmen der FOSSGIS-Konferenz in Osnabrück als Open-Source-Software veröffentlicht.

Ausgangspunkt für die Entwicklung des MapProxy war die Idee bestehende Kartenserver, bei voller Unterstützung des WMS-Standards, zu beschleunigen. MapProxy sollte in der Lage sein, bestehende WMS einer aktiven Geschwindigkeitsverbesserung zu unterziehen, ohne dass diese geändert werden müssen. Dies war die erste grundlegende Funktion die durch MapProxy bereitgestellt wurde.

Die Grundfunktionalität des MapProxy ermöglicht es, beschleunigte WMS wie gewohnt in vorhandenen Web- und Desktop-Anwendungen zu nutzen.

Die Beschleunigung der Dienste funktioniert durch Zwischenspeichern der abgerufenen Karten - so genanntes Caching. MapProxy liefert die Karten direkt aus dem Cache, so dass eine deutliche Geschwindigkeitssteigerung möglich ist. Anfragen werden dabei in beliebiger Auflösung beantwortet. Dadurch ist weiterhin ein freies Zoomen möglich, wie es in vom Desktop-GIS bekannt ist. Neben dem Ausliefern von Karten bietet MapProxy die WMS Funktionalitäten GetFeatureInfo und GetLegend.

Durch die zentrale Position eignet sich MapProxy außerdem dazu, weitere Funktionen anzubieten.

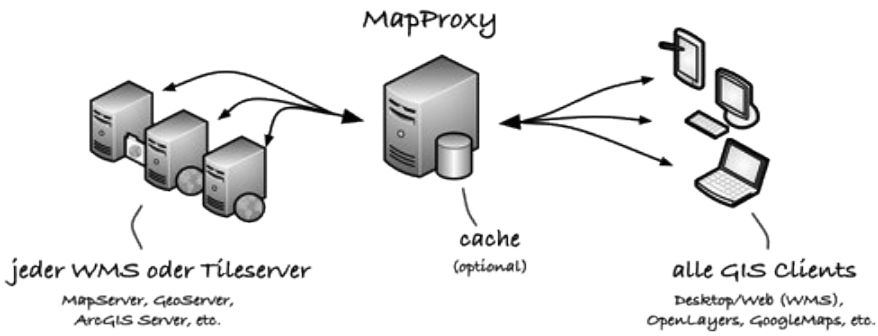


Abb 1: Integration von MapProxy

Mobile Anwendungen

Die Verbreitung von mobilen Endgeräten steigt stetig an. Häufig besitzen die verwendeten Displays eine sehr hohe Punktdichte. Dies führt dazu, dass normale Karten für den Betrachter oft pixelig oder unscharf aussehen.

MapProxy ermöglicht es, Karten in hoher Auflösung abzurufen und auszuliefern. Über einen extra Layer kann der gewünschte DPI-Faktor einfach mit konfiguriert werden.

Nachträgliche Bildbearbeitung

Um die Kartenauslieferung noch weiter zu beschleunigen, bietet MapProxy die Möglichkeit die Kartenbilder zusätzlich zu bearbeiten. Über verschiedene Resampling-Methoden und können Bilder zum Beispiel in geringen Bit-Varianten abgelegt werden.

Eine weitere Besonderheit ist der so genannte „mixed-mode“. Hierbei entscheidet MapProxy selbst, ob die Bilder als PNG oder JPEG gespeichert werden sollen. Besonders gut geeignet ist dieser Modus zum Beispiel für Luftbilder: Wird keine Transparenz benötigt, werden die platzsparenden JPEG-Bilder verwendet; den in Randbereichen wird das Bild als PNG abgespeichert, um die Transparenz zu erhalten.

Zusätzlich verfügt MapProxy über die Möglichkeit, Wasserzeichen in der Karte einzublenden. Diese können transparent über das Bild geelgt werden, um den Gesamteindruck der Karte nicht zu stören.

Zusammenfassen von Kartendiensten

Ein Beispiel für die Nutzung von MapProxy ist die Kombination zweier Luftbild-WMS-Server mit einem Fachdaten-Overlay. Der entstandene Layer kann wie gewohnt in einem Geoinformationssystem eingebunden werden und beinhaltet sowohl die Luftbilder als auch die Fachdaten. Auch wenn MapProxy als kaskadierender Server eingesetzt wird, unterstützt MapProxy die WMS Funktionen GetFeautreInfo und GetLegendGraphic. Wenn nötig, können die verwendeten Kartendienste hierbei auch ohne Zwischenspeichern verwendet werden.

Absicherung von Diensten

MapProxy verfügt über die Möglichkeit vorhandene Kartendienste abzusichern und stellt hierfür eine flexible Schnittstelle bereit, die auch mit bereits bestehenden Benutzer-datenbanken verbunden werden kann.

Bei der Nutzung der Schnittstelle stellt MapProxy eine Vielzahl von Funktionen zur Absicherung der Daten zur Verfügung. Neben den Gesamtdaten können zum Beispiel auch exakte Gebiete oder einzelne Layer abgesichert werden. Über ein Polygon kann definiert werden, auf welche Gebiete ein einzelner Benutzer zugreifen darf und auf welche nicht.

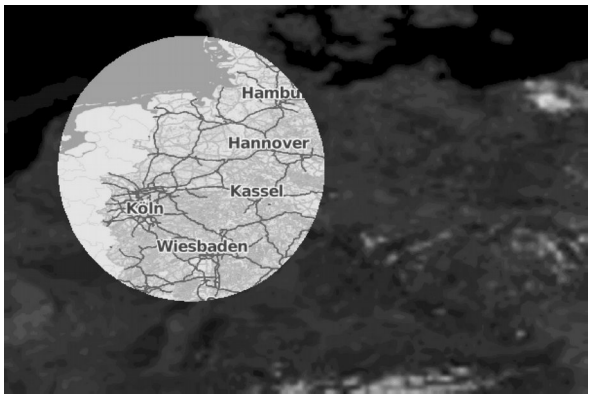


Abb. 2: Absicherung mit Puffer um einen Punkt

Abb. 3: Absicherung mit einem Polygon

Effizientes Seeding und Co.

MapProxy erstellt die Karten sobald diese abgerufen werden. Um die Geschwindigkeit für häufige Anfragen zu beschleunigen, ist es möglich die Karten entsprechend vorzugenerieren. Mithilfe des MapProxy-Seeding-Tool kann dies automatisiert durchgeführt werden.

Hiermit können Karten in festgelegten Ausschnitten oder Bereichen erzeugt werden. Von der einfachen Angabe einer BoudingBox bis hin zu einer komplexen Abfragen einer Geometrie aus einer Da-

tenbank kann eine Angabe erfolgen. Auf diese Weise können Karten nach Änderungen an den Daten gezielt aktualisiert werden.

Zusätzlich verfügt MapProxy über weitere Boardwerkzeuge wie zum Beispiel ein Kommandozeilenwerkzeug, welches bei der Konfiguration von MapProxy behilflich ist. Neben dem Analysieren von bestehenden WMS-Capabilities, ist auch die Berechnung von Kacheln und Gittern möglich.

Kontakt zu den Autoren:

Dominik Helle
Omniscale GmbH & Co. KG
Nadorster Straße 60
26123 Oldenburg
E-Mail: helle@omniscale.de

Oliver Tonnhofer
Omniscale GmbH & Co. KG
Nadorster Straße 60
26123 Oldenburg
E-Mail: tonnhofer@omniscale.de

Literatur

[1] MapProxy Homepage: http://mapproxy.org,

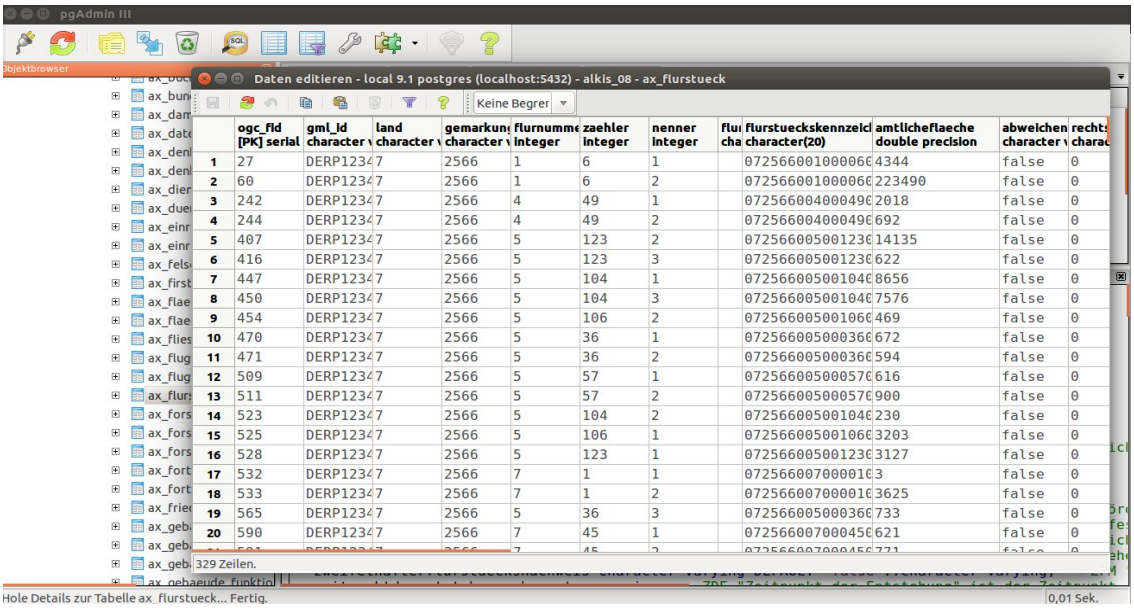
[2] MapProxy API Dokumentation: http://mapproxy.org/docs/nightly/auth.html

PostNAS-Suite - Lösungen für den ALKIS Datenimport, die Darstellung, Informationsausgabe und Suche

Astrid Emde

Die PostNAS Suite bietet Lösungen für den Import, die Weiterverarbeitung und die Inwertsetzung von ALKIS- und ATKIS-Daten in OGC-konforme Geodateninfrastrukturen.

Von Datenhaltungskomponenten können Liegenschaftsdaten über die Normbasierte Austauschschnittstelle (NAS) oder als Fortführungsdatensatz (Nutzerbezogene Bestandsdatenaktualisierung - NBA) bereitgestellt werden. Diese XML basierten Austauschformate können über die eigens für PostNAS erweiterte Software GDAL/OGR eingelesen und in unterschiedliche Formate übertragen werden. Unterstützt wird PostgreSQL, aber auch dateibasierte Formate wie Shape oder GML.



	ogc_fid [PK] serial	gml_id character	land character	gemarkung character	flurnumme integer	zaehler integer	nenner integer	flurstueckskennzeich character(20)	amtlicheflaeche double precision	abweichen character
1	27	DERP1234	7	2566	1	6	1	072566001000066	4344	false
2	60	DERP1234	7	2566	1	6	2	072566001000066	223490	false
3	242	DERP1234	7	2566	4	49	1	072566004000496	2018	false
4	244	DERP1234	7	2566	4	49	2	072566004000496	692	false
5	407	DERP1234	7	2566	5	123	2	072566005001236	14135	false
6	416	DERP1234	7	2566	5	123	3	072566005001236	622	false
7	447	DERP1234	7	2566	5	104	1	072566005001046	8656	false
8	450	DERP1234	7	2566	5	104	3	072566005001046	7576	false
9	454	DERP1234	7	2566	5	106	2	072566005001066	469	false
10	470	DERP1234	7	2566	5	36	1	072566005000366	672	false
11	471	DERP1234	7	2566	5	36	2	072566005000366	594	false
12	509	DERP1234	7	2566	5	57	1	072566005000576	616	false
13	511	DERP1234	7	2566	5	57	2	072566005000576	900	false
14	523	DERP1234	7	2566	5	104	2	072566005001046	230	false
15	525	DERP1234	7	2566	5	106	1	072566005001066	3203	false
16	528	DERP1234	7	2566	5	123	1	072566005001236	3127	false
17	532	DERP1234	7	2566	7	1	1	072566007000016	3	false
18	533	DERP1234	7	2566	7	1	2	072566007000016	3625	false
19	565	DERP1234	7	2566	5	36	3	072566005000366	733	false
20	590	DERP1234	7	2566	7	45	1	072566007000456	621	false

Das Projekt wurde 2008 von der WhereGroup und diversen Unterstützern ins Leben gerufen, um GDAL/OGR um die Unterstützung des NAS-Formats zu erweitern.

In den letzten Jahren hat sich PostNAS in vielerlei Hinsicht entwickelt und ist zur PostNAS Suite geworden. Mit einer Unmenge neuer Funktionen und Komponenten und vor allem vielen am Projekt beteiligten Anwendern, Entwicklern und Firmen wurde die Anwendung zu einem leistungsstarken Paket rund um die Nutzung von ALKIS und ATKIS.

So ist PostNAS mittlerweile in allen Bundesländern in öffentlichen Verwaltungen und bei Dienstleistern im Einsatz.

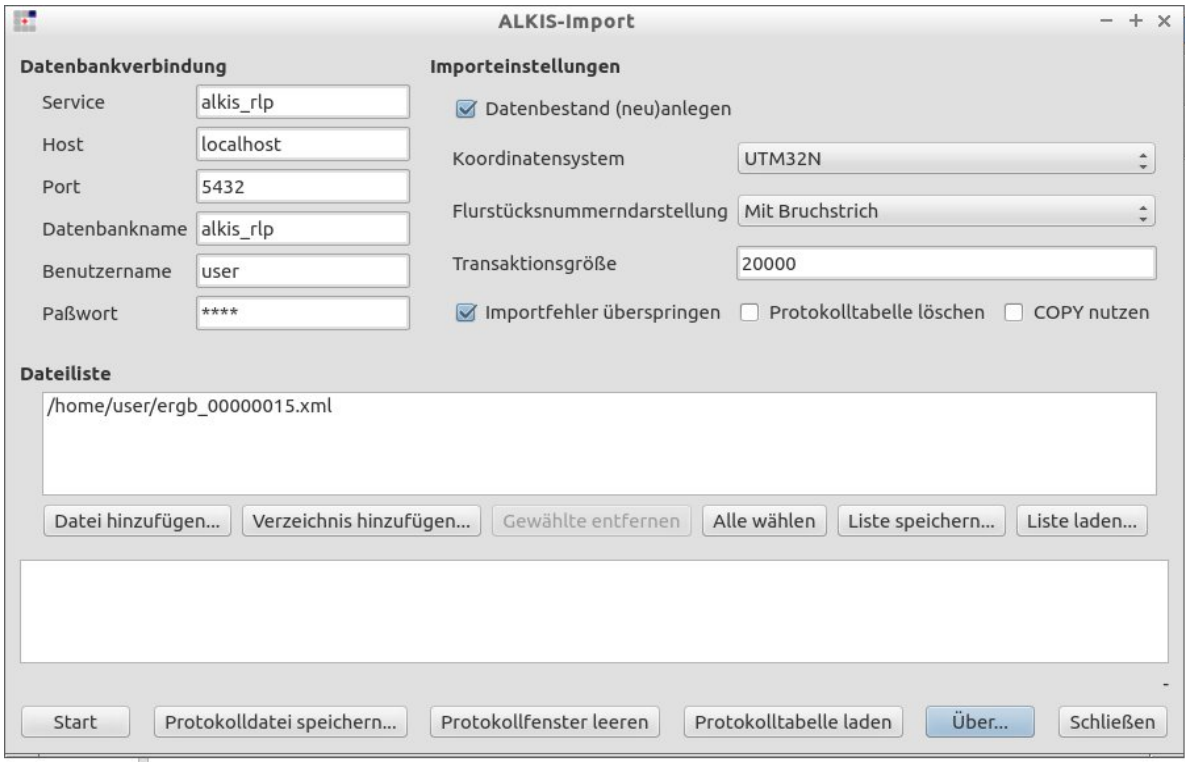
Die PostNAS Suite umfasst diverse Komponenten:

- Die eigentliche in GDAL/OGR (ogr2ogr) implementierte NAS-Schnittstelle
- MapServer-Mapdateien zur standardnahen Darstellung von ALKIS- und ATKIS-Daten
- norGIS ALKIS Import – grafische Oberfläche oder Shellskript zum Import [2]
- Skripte zur Suche nach Flurstücken, Adressen, Eigentümern und im Grundbuch (von Frank Jäger Kreis Lemgo) zur Integration in WebGIS-Clients z.B. Mapbender2
- Skripte zur nicht standardkonformen Beauskunftung und Informationsausgabe (von Frank Jäger Kreis Lemgo) in WebGIS-Clients z.B. Mapbender2 oder Mapbender3
- Plugins für den Import und die Visualisierung in QGIS, MapServer und AutoCAD [2]
- Ein Plugin zur Flurstückssuche in QGIS (Marvin Brandt Kreis Unna) [3]

norgis ALKIS Import

Werkzeug zum Import von NAS Dateien in die PostgreSQL Datenbank. Das Werkzeug kann über die grafische Oberfläche oder als Shellskript verwendet werden und kann eine Liste an NAS Dateien einlesen. Der Import führt dabei für jede NAS Datei den ogr2ogr Befehl aus.

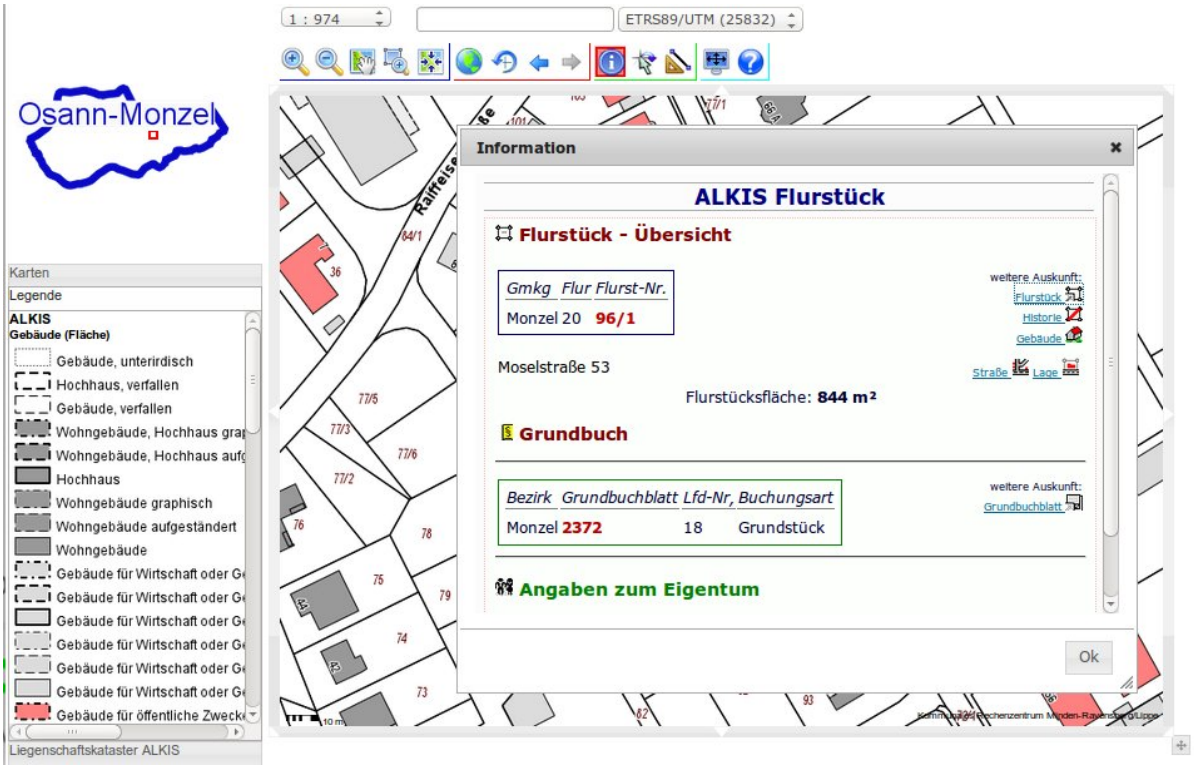
Über das Werkzeug kann das Datenmodell angelegt werden. Nach dem Import erfolgt ein Postprocessing, das zu einer Aufbereitung der eingelesenen Daten für die Datrstellung führt. Es werden dabei die GeoInfoDok Ableitungsregeln aus dem ALKIS-Signaturkatalog weitgehend umgesetzt.



Skripte zur Suche und Navigation

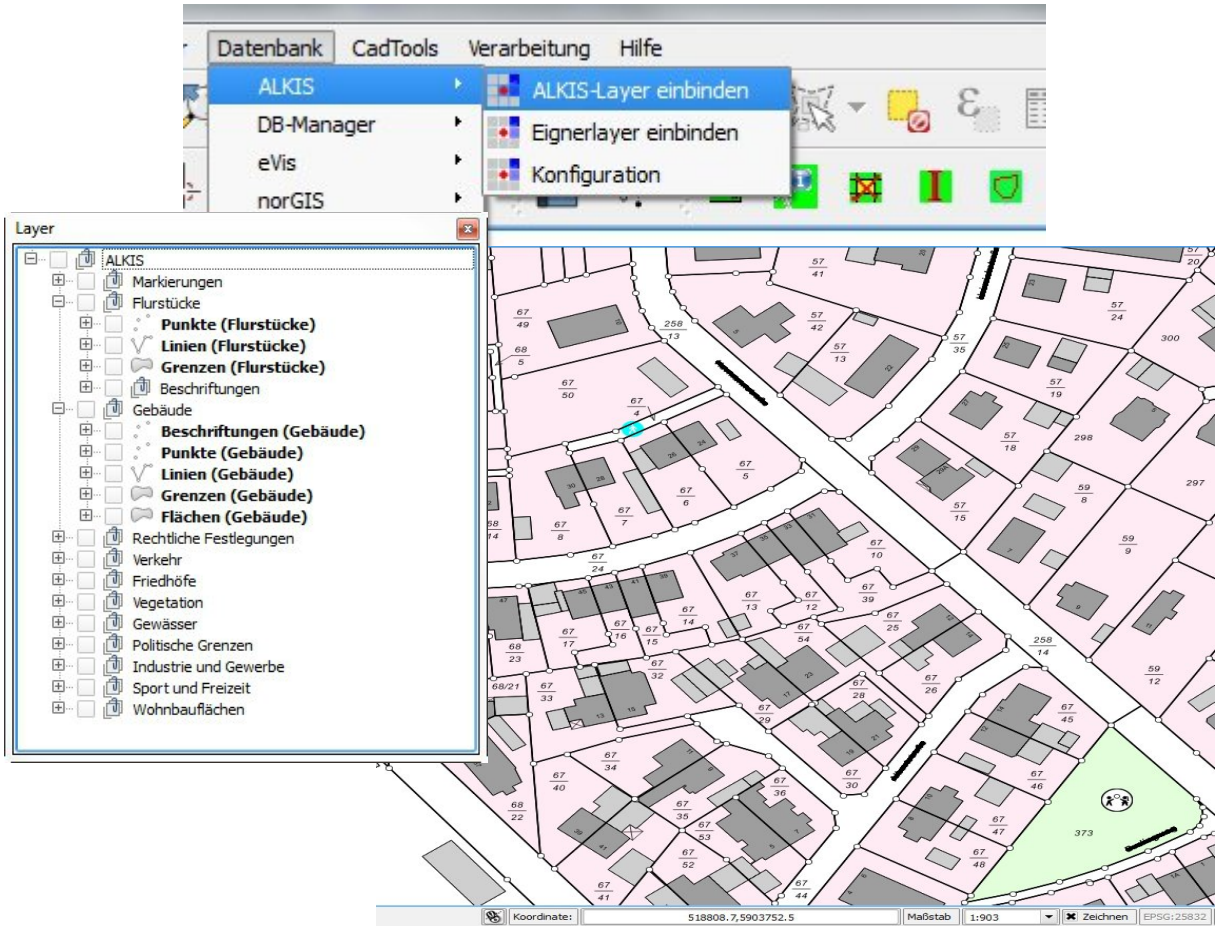
Zwei wichtige Komponenten sind die auf die PHP basierte Suche und Navigation. Diverse Skripte dienen hierbei zur Suche nach Flurstücken, Adressen, Eigentümern und Grundbuch zur Integration in

WebGIS-Clients wie beispielsweise Mapbender. Dazu kommen Skripte zur nicht standardkonformen Beauskunftung und Informationsausgabe.



QGIS ALKIS Plugin

Nach der Einbindung des Plugins norGIS ALKIS können zuvor mit norGIS-ALKIS-Import geladene Daten in QGIS gemäß der GeoInfoDok visualisiert werden. Es kann eine Eignerabfrage (Einzelabfragen) erfolgen. Die Darstellung kann als UMN-Mapfile exportiert werden (optional, erfordert python-mapscript) oder über QGIS Server als WMS bereitgestellt werden.



Der Sourcecode der verschiedenen PostNAS-Komponenten lässt sich auf der Projekt-Webseite [1] und im Github [2][3] herunterladen oder alternativ über das QGIS Plugin-Repository [4] einbinden.

Anwendertreffen

Das große Interesse an der PostNAS Suite wurde auch beim letzten Anwendertreffen deutlich, das am 25. Mai 2016 beim Kreis Unna stattfand. Etwa 20 Anwender und Entwickler trafen sich im Kreishaus und diskutierten über diverse Themen. Ziel war es, die zukünftige Ausrichtung und die nächsten Meilensteine des Projektes abzustimmen, Wünsche der Anwender aufzunehmen und konkrete Arbeitspakete für die weitere Umsetzung zu definieren. Es war ein sehr produktives Treffen, dessen Ergebnisse sich auf der Projekt-Webseite einsehen lassen [5].

Das nächste Anwendertreffen findet auf der FOSSGIS 2016 in Salzburg statt [6]. Wir würden uns über viele neue Teilnehmer freuen.

Kontakt zur Autorin:

Astrid Emde
WhereGroup GmbH & Co. KG
Eifelstraße 7
53119 Bonn
+49 (0)228 90 90 38 19
astrid.emde@wheregroup.com

Literatur und weiterführende Links:

[1] http://postnas.org

[2] norGIS ALKIS Import https://github.com/norBIT/alkisimport

[3] https://github.com/Kreis-Unna/PostNAS_Search

[4] QGIS Plugins: https://plugins.qgis.org/search/?q=ALKIS

[5] https://trac.wheregroup.com/PostNAS/wiki/PostNASAnwendertreffen2016-05-25

[6] http://trac.wheregroup.com/PostNAS/wiki/PostNASAnwendertreffen2016-07-04

OpenStreetMap und Wikidata

Michael Meyer

Was Tim-Berners Lee mit dem www begonnen hat (Texte zu verlinken), wird nun mit dem Öl des 21. Jahrhunderts, den Daten weitergeführt. Lasst OpenStreetMap ein Teil der Linked Open Data Cloud werden!

Zwischen OpenStreetMap und Wikipedia wird bereits rege verlinkt - Nehmen wir die nächste Stufe und verlinken auch nach Wikidata - und zurück!

Im ersten Teil wird kurz Wikidata und sein Datenmodell erklärt.

Danach werden die Einsatzgebiete von Wikidata zur Bereicherung von OSM aufgezeigt:

- Wikidata als Lösung des permanent ID-Problems
- Wikidata als nachhaltige Lösung der Sprachproblematik (name:*) bei Orten
- Übersetzung von OSM-Tags für Anwendungsprogramme und Editoren
- Neue Arten von Analysemöglichkeiten, z.B. zur Wortherkunft (name:etymology:wikidata)

Wie kann man nun in der Praxis OSM und Wikidata verlinken:

- JOSM als Tool, um Wikidata-Tags in OSM einfach zu setzen
- Wie wird von Wikidata nach OSM gelinkt

Zum Schluss wird als Beispielsanwendung eine Web-Karte zur Aggregierung aus verlinkten Datenquellen vorgestellt.

Was Tim-Berners Lee mit dem www begonnen hat (Texte zu verlinken), wird nun mit dem Öl des 21. Jahrhunderts, den Daten weitergeführt. Lasst OSM ein Teil der Linked Open Data Cloud werden!

Zwischen OpenStreetMap und Wikipedia wird bereits rege verlinkt - Nehmen wir die nächste Stufe in Richtung Maschinenlesbarkeit und verlinken auch nach Wikidata - und zurück!

Die Karte verändert sich: Der Standardstil openstreetmap-carto

Michael Glanznig

Alle paar Wochen ist es wieder soweit: „Mapnik-Update“ (openstreetmap-carto/osm-carto), die Karte verändert sich. Nicht immer sind so große Änderungen wie die Darstellung der Straßen dabei und nicht immer sind alle von den Änderungen begeistert. Dieser Vortrag versucht ein wenig hinter die Kulissen der osm-carto Entwicklung zu blicken und darzustellen in welchem Spannungsfeld sich der Standardstil bewegt.

Welche Ziele hat osm-carto eigentlich? Wie finden Änderungen den Weg in den Kartenstil? Wer entscheidet das und wie? Wie erfolgt der Test einer Änderung? Kann ich dazu beitragen? Das sind Fragen, die man sich als Mapper möglicherweise gelegentlich stellt. OSM ist zwar eine Datenbank, trotzdem kommt man auf openstreetmap.org gleich mit einer bestimmten Darstellung der Datenbank in Kontakt.

Die Ziele von osm-carto widersprechen sich teilweise. Einerseits soll der Mapper Feedback-Loop unterstützt werden (Wie sieht meine Änderung aus?) und die Datenvielfalt von OSM möglichst gut abgebildet werden und andererseits soll der Stil gut lesbar sein und gut aussehen. Das ist nicht gerade einfach und die verschiedenen Ziele kommen sich gerne in die Quere. Zusätzlich gibt es auch technische und organisatorische Limitationen, wie das bei Freiwilligen-Projekten eben vorkommt. Mit Beispielen versuche ich zu veranschaulichen wie der Entwicklungsprozess von osm-carto funktioniert - von der Idee, über die Umsetzung, den Test, die Diskussion, bis zur Aufnahme in den Stil. Ziel ist es mit dem Blick hinter die Kulissen einige Entscheidungen und Gegebenheiten anschaulicher und nachvollziehbarer zu machen.

Natürlich sind nicht alle mit der Darstellung auf der Standardkarte zufrieden. Das müssen sie auch nicht sein, denn Vielfalt ist eine der großen Stärken von OSM. Das gilt auch für Darstellungen der Datenbank. Je mehr verschiedene Karten, desto besser. Ich möchte kurz ansprechen warum es eine gewisse Anzahl von Karten auf openstreetmap.org gibt und wie sich die Zahl in Zukunft vielleicht erhöhen könnte. Ausgehend von osm-carto zeige ich im Überblick wie man mit Hilfe von Kosmtik und CartoCSS den Kartenstil verändern kann und was man für ein Entwickler-Setup alles braucht. Damit ist man theoretisch auch in der Lage seine eigene Karte zu gestalten.

Kontakt zum Autor:

Michael Glanznig
nebulon42@yandex.com
https://openstreetmap.org/user/nebulon42
https://github.com/nebulon42

SciGRID: ein offenes Referenzmodell europäischer Übertragungsnetze für wissenschaftliche Untersuchungen

Wided Medjroubi, Carsten Matke, David Kleinhans

Kurzfassung:

Die Verfügbarkeit belastbarer Daten der Europäischen Übertragungsnetze ist derzeit nicht gegeben, was Forschungsaktivitäten im Bereich des Ausbaus Erneuerbarer Energien und der Netzplanung erschwert. Zentraler Gegenstand des SciGRID Projektes [1] , das in Oktober 2014 gestartet ist, ist die Entwicklung eines frei verfügbaren und gut dokumentierten Netzmodells der europäischen Übertragungsnetze, das als erstes Übertragungsnetzmodell unter der ODbL Lizenz [2] kostenlos zur Verfügung steht.

Der Schwerpunkt von SciGRID liegt in der Nutzung der ebenfalls unter der ODbL stehenden Geodaten von OpenStreetMap (OSM) [3] . Die OSM Relationen werden dafür nach dem Schlüssel „route=power“ gefiltert und in eine relationale Datenbank überführt. Eine automatische Verarbeitung der relevanten Daten, das Extrahieren und Abstrahieren des Netzmodells erfolgt durch ein SQL Skript. Ein Beispiel der Ausgangsdatenbanken für die Knoten und Wege des Netzes ist in Abb.1 dargestellt.

Aufbauend auf diesen Daten konnte im Juni 2015 zunächst ein qualitativ hochwertiges Modell der Deutschen Höchstspannungsnetze (220/380 kV) veröffentlicht werden (Abb. 2). Leitungen und Leitungsverläufe sind dabei abstrahiert und auf in technischer Sicht relevante Kenndaten wie Leitungslängen und Impedanzen reduziert worden. Perspektivisch soll die Methodik auf das Europäische Netz ausgeweitet werden.

In diesen Beitrag werden das Vorgehen und die Nutzung offener Geodaten in SciGRID vorgestellt und erste Ergebnisse der Charakterisierung des deutschen Übertragungsnetzes präsentiert.

Automatische Edits und Importe in OpenStreetMap

Frederik Ramm

Mit zunehmender Bekanntheit von OpenStreetMap steigt auch die Anzahl von Menschen, die es gut mit dem Projekt meinen und mal eben weltweit etwas "reparieren". OpenStreetMap ist voll von Tippfehlern und logischen Problemchen, von Flüssen, die aufwärts fließen und Straßen, die durch Häuser führen. Was liegt für den cleveren Hacker da näher, als Probleme wie diese kurzerhand mit einem Python-Skript zu reparieren - Handarbeit würde ja viel zu lange dauern und ist ausserdem was für die Dummen!

Wer nicht programmieren kann, der reiht wenigstens ein paar frei verfügbare Tools geschickt aneinander, um ein weltweites Suchen und Ersetzen zu realisieren, oder man beglückt OSM mit einem freien Adressdatensatz, der Web herumschwirrte und den OSM aus völlig unverständlichen Gründen noch nicht vereinnahmt hatte - und solche Vorhaben vorher mit der Community zu diskutieren, ist eh was für Anfänger, oder? It's a Wiki, Feuer frei!

Dieser Vortrag erklärt die Risiken und Nebenwirkungen solcher gut gemeinten Beiträge und versucht, den Schaffensdrang hilfsbereiter Hacker in community-verträglichere Bahnen zu lenken.

Dieser Vortrag zeigt die Probleme auf, die mit großflächigen automatischen und halb-automatischen Edits in OpenStreetMap einhergehen, und diskutiert Alternativen hierzu.

Braucht OpenStreetMap Flächen und Kanten?

Roland Olbricht

Modellbildung findet bei OpenStreetMap nicht nur durch die Wahl der Tags statt. Neben der noch einigermaßen prominenten Frage, ob POIs in Gebäuden als Node oder Fläche erfasst werden ist eine weniger beachtete, aber deutlich weitreichendere Frage, welche Elemente als Linie oder Fläche erfasst werden. Offensichtliche Zweifelsfälle sind Fußgängerzonen in Breiten zwischen Straße und Platz. Aber auch generell nehmen die meisten Objekte im realen Raum Fläche ein. Gleichzeitig haben dennoch einige Objekte entweder sehr strikten Liniencharakter wie z.B. Gleise. Und für andere Objekte trägt die Linie die wesentliche Information, z.B. bei Treppen die Richtung der Steigung.

Ein ausreichend rigide definites Mischmodell muss theoretisch die Möglichkeit zulassen, aus seinen Daten sowohl ein reines Kantenmodell als auch ein reines Flächenmodell zu berechnen. In diesem Vortrag werden Werkzeuge vorgestellt, um ein Kartenmodell als auch ein reines Flächenmodell zu berechnen und um die dabei zutage tretenden Unstimmigkeiten zu finden. Modellbildung findet bei OpenStreetMap nicht nur durch die Wahl der Tags statt. Werden Elemente als Linie oder Fläche erfasst? Offensichtliche Zweifelsfälle sind Fußgängerzonen in Breiten zwischen Straße und Platz. Aber auch generell nehmen die meisten Objekte im realen Raum Fläche ein. Gleichzeitig haben dennoch einige Objekte entweder sehr strikten Liniencharakter wie z.B. Gleise.

GeoPackage, das Shapefile der Zukunft

Pirmin Kalberer

Im Februar 2014 hat das Open Geospatial Consortium (OGC) den GeoPackage Encoding Standard offiziell freigegeben.

Dieses noch junge Format hat sich bereits gut etabliert und wird in vielen GIS-Produkten unterstützt.
In GeoPackage-Dateien können sowohl Vektor- als auch Rasterdaten mit zugehörigen Metainformation gespeichert werden.

Damit können Geodaten einfach ausgetauscht und auch auf mobilen Geräten effizient genutzt werden.
Als Fileformat wird SQLite verwendet, welches einen effizienten Zugriff mit einem SQL-API bietet. Das Format ist erweiterbar und wird unter anderem bereits für die Speicherung von Point Cloud Daten verwendet. Auch Vektor-Tiles können darin für den Offline-Gebrauch abgelegt werden.

Im Februar 2014 hat das Open Geospatial Consortium (OGC) den GeoPackage Encoding Standard offiziell freigegeben.

Dieses noch junge Format hat sich bereits gut etabliert und wird in zahlreichen GIS-Produkten unterstützt.
In GeoPackage-Dateien können sowohl Vektor- als auch Rasterdaten mit zugehörigen Metainformation gespeichert werden. Damit können Geodaten einfach ausgetauscht und auch auf mobilen Geräten effizient genutzt werden.

Umstellung von Übungen auf QGIS

Michael Paulmann

An der Hochschule für Technik (HfT) in Stuttgart wird von proprietärer Geoinformationssoftware auf die Open Source Lösung QGIS umgestellt. Im Modul Geografische Informationsysteme an der HfT Stuttgart werden Studenten im 1. und 2. Semester mit einem Geographischen Informationsystem vertraut gemacht. Bisher war dies die propriätere Software Geomedia von Intergraph. Nun wird dieses Modul auf QGIS umgestellt. Im Vortrag erhalten Sie Einblick über den bisherigen Fortschritt der Umstellung und können sehen welche Probleme auftraten und mit welchen Problemen zu rechnen sind bei der Umstellung auf QGIS.

Umstellung des Hochschulmoduls Geographische Informationssysteme von proprietärer Software auf die Open Source Software QGIS. Es gibt einen Einblick in die Probleme die bisher auftraten und den bisherigen Fortschritt.

Automatische Erkennung der Projektion von Geodaten. Unter Verwendung freier Geodaten und offener GIS-Services im Internet.

Manfred Egger

Entstehungsgeschichte dieses Projektes:

Im Rahmen meiner beruflichen Tätigkeit beim Forsttechnischen Dienst für Wildbach- und Lawinenverbauung Sektion Tirol, war ich häufig mit Supportanfragen von GIS-Anwendern konfrontiert, die große Schwierigkeiten mit der Integration von Geodatenlieferungen in bestehenden GIS-Projekten hatten. Der Grund war meist, dass die Projektion des Datensatzes unbekannt oder falsch zugewiesen war. Für meinen Arbeitgeber entwickelte ich auf Basis von Bezirksnamen/-grenzen in Österreich, ein kleines Tool zur automatischen Projektionserkennung. Im Rahmen der AGIT 2016 wurde das Tool als kleine Länderlösung von mir präsentiert (siehe Literaturverzeichnis).
Was mich enorm überraschte war, dass die Supportanfragen sofort zurückgingen und auch neue Mitarbeiter oder fachfremde Praktikanten, fast ohne Schulungsaufwand Geodaten zumindest für zehn gängige Österreichische Projektionen mit Knopfdruck in GIS-Projekten integrieren konnten. Auch für fachkundige wie mich, war praktisch mit Knopfdruck die Projektion einem Datensatz zuzuweisen. Seitens der GIS-Anwender wurde das Tool als deutliche Arbeitserleichterung bezeichnet.
Was mich auch wunderte war, dass mir derartige Lösungen bis jetzt nicht bekannt waren, obwohl diese programmiertechnisch sehr einfach umsetzbar sind.
Aufgrund dieser Beobachtungen und Eindrücke begann ich mich in meiner Freizeit mit einer allgemeinen globalen Lösung, die alle weltweit bekannten Projektionen berücksichtigt, zu befassen und fand ein Open Source Projekt von Aaron Racicot (2013). Auf dessen Basis entstand von mir ein Erstversuch einer globalen Lösung zur automatischen Projektionserkennung/-zuweisung, die Inhalt dieses Artikels ist.

Zielgruppe:
Zielgruppe dieses Programms sind vor allem Anwender, die mit geringen Fachkenntnissen zu Koordinatensystemen, ohne Aufwand, gelieferte Daten in GIS-Projekte lagerichtig integrieren möchten.

Beschreibung:
Das Programm mit dem Namen SHAPEFILE PROJECTIONFINDER wertet ein ausgewähltes Shapefile mit unbekannter Projektion (ohne PRJ-Datei), in Kombination mit einer geographischen Referenzkoordinate aus und erhält automatisch nach Anfrage beim bestehenden GIS-Testservice von Aaron Racicot eine Liste möglicher zutreffender Projektionen. Nach händischer Auswahl wird für jede ausgewählte Projektion eine Kopie des Shapefiles mit entsprechender PRJ-Datei erstellt. Der GIS-Anwender muss nun die erstellten Kopien in sein GIS-Projekt laden und entscheidet nach visueller Prüfung über die richtige Projektion.

Grundregel der Bedienung:
Je näher die geographische Koordinate bei der projizierten Koordinate des Shapefiles liegt, desto eindeutiger kann die Projektion durch einen niedrigen Distanzwert erkannt werden (siehe Abbildung 1).

Automatische Erkennung der Projektion von Geodaten. Unter Verwendung freier Geodaten und offener GIS-Services im Internet.

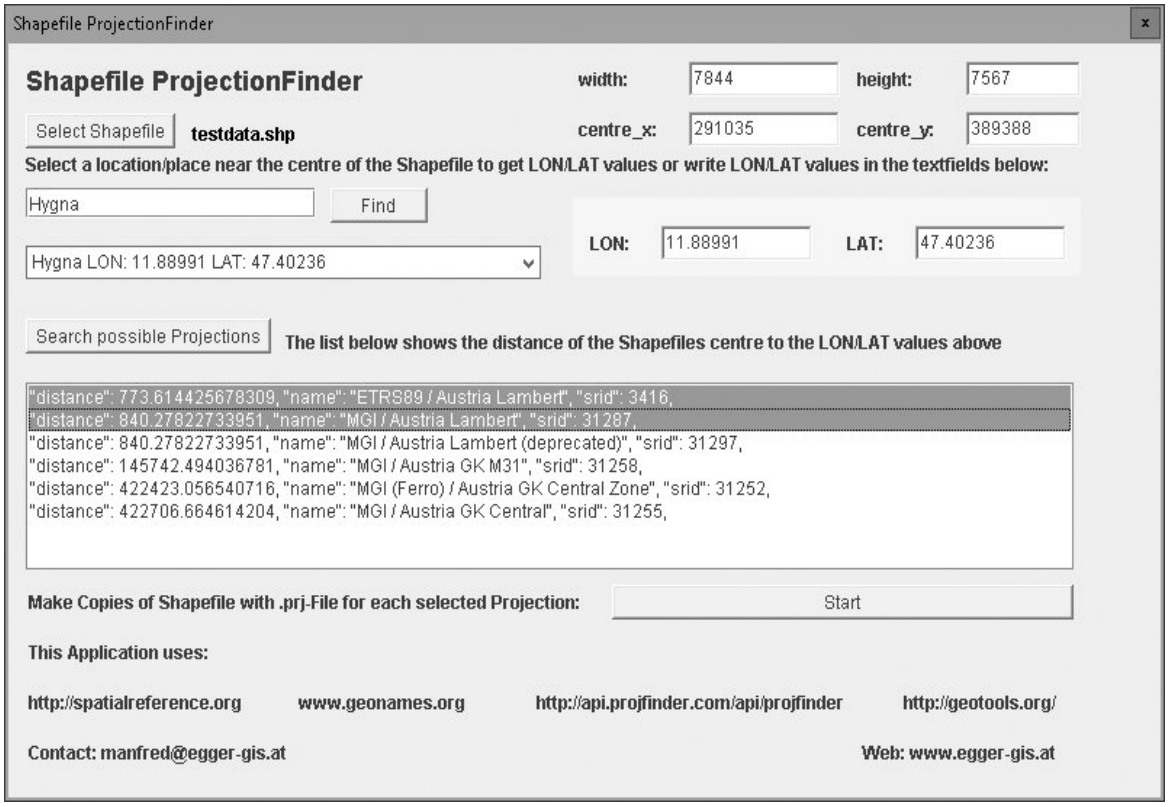


Abb. 1: GUI mit Auswahlliste möglicher Projektionen.

Um diese Grundregel besser umsetzen zu können, plane ich sowohl für das Shapefile, als auch für die Auswahl der geographischen Referenzkoordinate zwei kleine Kartenfenster in das Programm zu integrieren, um eine bessere Flexibilität der Punktauswahl zu erreichen. Nur den Mittelpunkt des Shapefiles zu verwenden, ist vor allem bei Datensätzen über sehr große Gebiete nicht optimal.

Technischer Hintergrund:
Bei diesem GIS-Tool handelt es sich um eine Java-Netbeans Desktopanwendung unter Nutzung der freien Java-Bibliothek GeoTools. Der Vorteil der Desktopanwendung liegt aus meiner Sicht darin, dass ich beim Anlegen von Kopien des Ausgangsshapefiles am lokalen Rechner deutlich mehr Freiheiten habe, als mit Javascript basierten Technologien. Der Anwender benötigt keine besonderen Plugins, oder muss Vertrauensfragen beantworten.
Im Hintergrund werden für dieses Tool jedoch drei freie WebGIS-Services genutzt:

- http://projfinder.com: Dieses GIS-Testservice von Aaron Racicot, ermittelt mögliche zutreffende Projektionen.
- http://geonames.org: Hilfe für die Ermittlung einer geographischen Koordinate möglichst nahe dem Mittelpunkt des Shapefiles.
- http://spatialreference.org: Abfrage WKT für PRJ-Dateien der Shapefilekopien.

Problembereiche des Programms:

Für Shapefiles in geographischen Koordinaten, ist es sehr schwierig einen aussagekräftigen Distanzwert zu erhalten, da Schwankungen des Distanzwertes nur durch die Parameter des Referenzellipsoides entstehen. Daher werden auch viel zu viele mögliche Projektionen mit sehr ähnlichen Distanzwerten vorgeschlagen.

Ähnliche Probleme entstehen bei Projektionen, welche unterschiedliche Referenzellipsoide (zum Beispiel: UTM 32 N : WGS84 und ETRS98) verwenden. Hier listet aber das Programm nur sehr wenige

Vorschläge mit einem ähnlichen Distanzwert auf, wie man in Abbildung 1 in den ersten beiden hervorgehoben Zeilen sehen kann. Hier kann man jedoch Kopien erstellen und im GIS-Projekt im großmaßstäbigen Bereich, über einen pragmatischen Weg visuell entscheiden, welche Projektion besser passt. Problematisch sind CUSTOM-Projektionen, beziehungsweise dem Online-Service unbekannte Projektionen, da diese einfach nicht erkannt werden.

Wie geht es weiter?

Bevor ich mein Programm über meine Webseite zum Download zur Verfügung stelle, benötige ich einen eigenen Webserver, der die Aufgabe des bestehenden Testservice von Aaron Racicot übernimmt. Ich hoffe, dass ich bis Herbst 2016 einen Server habe und dann das Programm auch zum Download freigeben kann.

Es kann aber jeder meinen Quellcode einsehen (siehe Literaturverzeichnis) und selber hier eine Weiterentwicklung vorantreiben, wobei mich Aaron Racicot gebeten hat, unbedingt darauf hinzuweisen beim Produktivgang einen eigenen Server aufzusetzen. Im Literaturverzeichnis findet man auch einen Link zu seinem Quellcode.

Aus visionärer Sicht wäre wohl eine Integration des Testservice von Aaron Racicot in die Webseite http://spatialreference.org für mich persönlich am idealsten, da für alle Entwickler so immer alle registrierten Projektionen im Service inkludiert sind.

Kontakt zum Autor:

Manfred Egger
Alois-Schrott-Str. 34
6020 Innsbruck
Österreich
004369981372857
manfred@egger-gis.at

Literatur

[1] Egger, Manfred: Automatische Projektionserkennung mit geographischen Namen, AGIT - Journal für Angewandte Geoinformatik, 2-2016, Seitenangabe bei Abgabetermin unbekannt, 2016.

[2] Egger, Manfred: Free GIS Services von Manfred Egger. http://egger-gis.at, 2016.

[3] Egger, Manfred: SHPProjFinder. https://github.com/maegger/SHPProjFinder, 2016.

[4] Racicot, Aaron: projfinder.com. https://github.com/aaronr/projfinder.com, 2013.

Betrieb von QGIS in einer heterogenen Client-Server-Umgebung

Andreas Schmid

Das Amt für Geoinformation des Kantons Solothurn (Schweiz) betreibt ein Geographisches Informationssystem für die Kantonsverwaltung. Als Desktop-Anwendung für die Bearbeitung und Visualisierung von Geodaten stellt es den Benutzern QGIS auf einem zentralen Linux-Anwendungsserver zur Verfügung.
Der Standard-Arbeitsplatz in der Kantonsverwaltung besteht aus einem Thin Client, der mit Citrix-Technologie zu einem Windows-Desktop auf einem Windows-Anwendungsserver verbindet. Um die Linux-Anwendung QGIS in den Windows-Desktop zu integrieren, setzt das Amt für Geoinformation die Open-Source-Lösung X2Go ein: X2Go Server auf dem Linux-Anwendungsserver und X2Go Client auf dem Windows-Desktop.

Trotz der zweifachen Weiterleitung der Grafikausgabe ist diese Lösung in der Bedienung sehr performant.
Damit die Benutzer sich nicht separat am Linux-Anwendungsserver anmelden müssen, wird ihnen bei der Einrichtung des Linux-Benutzerkontos ein RSA-Schlüsselpaar erzeugt und dieses für die Anmeldung des X2Go-Clients am Linux-Anwendungsserver bereitgestellt.

Dadurch kann QGIS mit einem einzigen Klick auf eine Verknüpfung im Windows-Startmenu gestartet werden.
Mit dieser Lösung ist eine erstaunlich gute Integration einer Linux-Anwendung in die Windows-Umgebung gelungen. Dies nicht zuletzt dank der Flexibilität des X2Go-Clients, der mit seinen umfangreichen Startoptionen sehr fein gesteuert werden kann.

Das Amt für Geoinformation des Kantons Solothurn (Schweiz) betreibt für die Benutzer der Kantonsverwaltung QGIS auf einem Linux-Applikationsserver.

Für die Integration der Linux-Anwendung in den via Citrix bereitgestellten Windows-Desktop, der den Standard-Arbeitsplatz in der Kantonsverwaltung darstellt, kommt die Open-Source-Lösung X2Go zum Einsatz.
Der Vortrag präsentiert diese Lösung im Detail und geht auf deren Vor- und Nachteile ein. Zudem vergleicht er sie mit anderen Lösungen, die früher im Einsatz waren und nun abgelöst wurden.

QGIS meets MapProxy

Sara Biesel

Im Rahmen des Projekts TopDeutschland entwickelt das Bundesamt für Kartographie und Geodäsie ein System, um Kartenausschnitte zu speichern und die Karten Offline im Geoinformatiomssystem QGIS zu laden. Dabei wird ein Proxyserver „MapProxy“ des Unternehmens Omiscale GmbH & Co. KG verwendet, der bei bestehender Internetverbindung die Karten als Kacheln lokal ablegt. Diese Funktion ist besonders hilfreich für unsere Kunden wie zum Beispiel dem Technischen Hilfswerk, bei Einsätzen im Gelände ohne Internetverbindung. Damit die Daten flexibel genutzt werden können, liegt das ganze System auf einer externen Festplatte. Auch QGIS steht als portable Version zur Verfügung.

Durch das Zwischenspeichern der Kartenkacheln wird das Laden der Dienste beschleunigt. Über ein selbst geschriebenes Plugin in Python kann in QGIS entweder ein Rechteck, der Kartenextent oder Deutschland mit einem Umkreis von 100 Kilometern als ausgewählter Bereich zum Speichern angegeben werden. Zusätzlich können aber auch bisher gespeicherte Shapefiles als Ausschnitt verwendet werden. Da der komplette Kartendatensatz sehr groß werden kann, können nur die für den Nutzer wichtigen Gebiete auf der Festplatte gespeichert werden. Durch das Laden der Kartenkacheln über einen Web Map Service, können die Kartendaten auch aktualisiert werden.

Nach dem Aktivieren des Plugins können Punkte in die Karte gesetzt werden, die den zu speichernden Ausschnitt markieren. Die untere Abbildung zeigt den Auswahlvorgang.

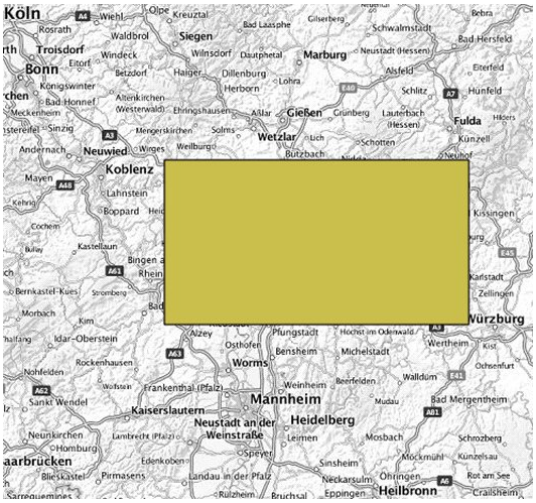


Abbildung 1: Auswahl Kartenausschnitt

Sobald das Rechteck abgeschlossen wurde, besteht die Möglichkeit den Bereich erneut zu zeichnen oder den bereits Gezeichneten zu verwenden. Die nächste Abbildung zeigt das Auswahlfenster des Plugins, welches sich nach dem Zeichnen automatisch öffnet.

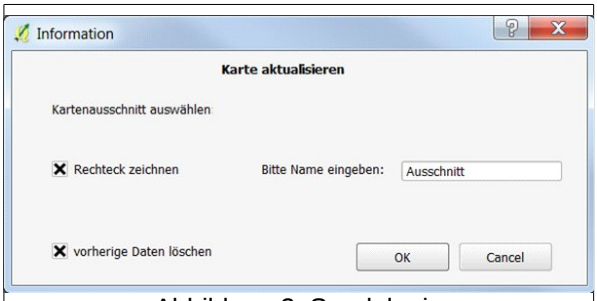


Abbildung 2: Seedplugin

Hier kann ein Name für den Ausschnitt angegeben und der Seedvorgang gestartet werden. Abbildung 3 zeigt den abgeschlossenen Seedprozess. Je nach Größe des ausgesuchten Bereichs und Zoomstufe, kann der Seedvorgang mehrere Stunden dauern.

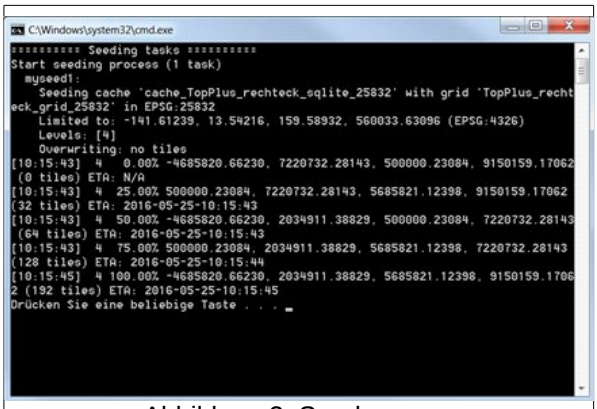


Abbildung 3: Seedprozess

Die heruntergeladenen Kartenkacheln stehen nun auch ohne Internetverbindung zur Verfügung und können über die Anwendung geladen werden. Abbildung 4 veranschaulicht dieses.

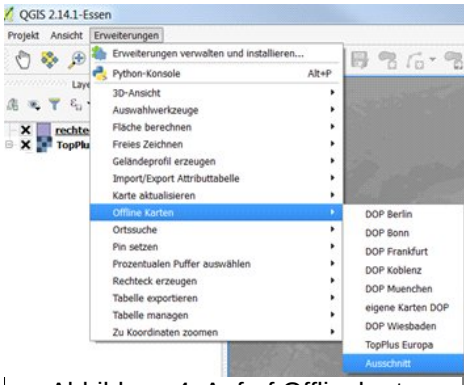


Abbildung 4: Aufruf Offlinekarte

Nach dem Aufruf des heruntergeladenen Kartenausschnitts wird direkt auf den Ausschnitt gezoomt wie Abbildung 5 zeigt.

Abbildung 5: gespeicherter Kartenausschnitt

Es besteht die Möglichkeit unterschiedliche WMS-Dienste zu speichern und diese anzeigen zu lassen. Die Kartendaten können zudem auf einen lokalen Rechner ausgelagert werden, da die Bilddaten in einer SQLite Datenbank vorliegen. Das Projekt befindet sich momentan noch in der Entwicklung.

Kontakt zum Autor:

Sara Biesel
Bundesamt für Kartographie und Geodäsie (BKG)
Richard-Strauss-Allee 11, 60598 Frankfurt am Main
+49 (0)69 6333-260
TopDeutschland@bkg.bund.de

Die Zugriffszeit auf den QGis-Mapserver um Faktor 100 beschleunigen

Jörg Habenicht

Was machen wir:

- mWerk erstellt Planungen für Breitbandinternet mit regionalen Grenzen für Kommunen und Anbieter.
- Wir erstellen Adressgenaue Planungen mit mehreren Millionen Adressdaten.
- Wir versorgen 300 User in 20+ Unternehmen und Kommunen.
- Wir bieten unseren Mandanten individualisierte Kartendaten mit unterschiedlichen QGIS Dateien.

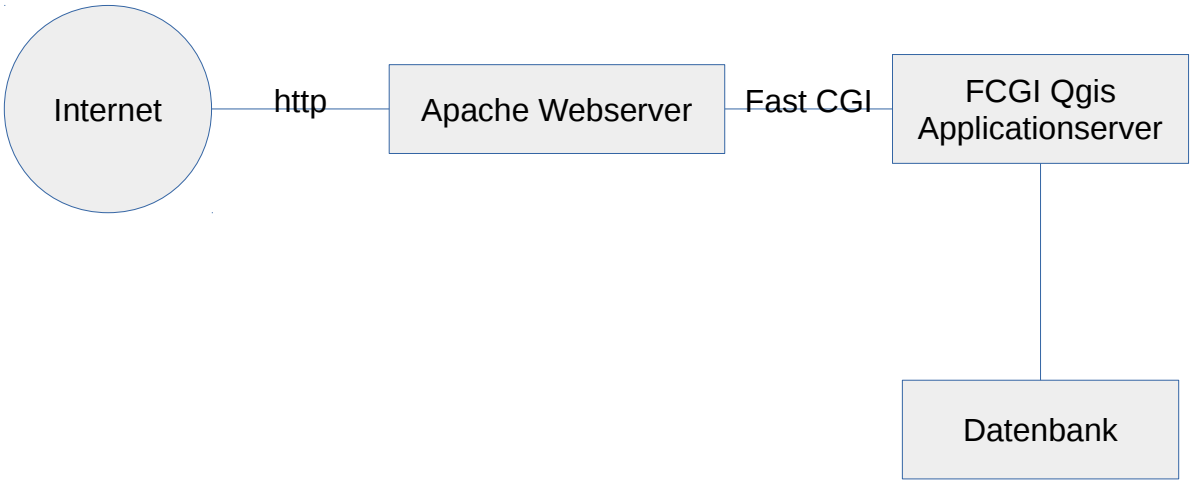
Unsere Herausforderung:

Wir haben für unsere User 20 verschiedene Karten im Portfolio mit ebenso vielen Qgis-Konfigurationsdateien. In Stoßzeiten beträgt die Klickrate auf dem Webserver ca. 10/Sekunde. Das ist alles nicht viel, wurde allerdings in der Vergangenheit schon einmal schwierig.

Im Herbst 2015 hatten wir mehrere Fälle von Thrashing auf dem Server, verursacht durch zu viele Qgis-Prozesse mit zu viel Speicherverbrauch. Eine Analyse ergab, dass die Prozesse bis zu 50 Sekunden benötigen, um die Konfiguration aus der Qgis-Datei zu parsen. Jeder Dreh mit dem Mausrad verändert den Zoomfaktor der Karte und löst eine Anfrage nach einem neuen Kartenbild aus. Was wiederum den Stress auf dem Webserver erhöhte.

Analyse und prinzipielle Problemlösung:

In unserem Setup wird der QGis-Mapserver verwendet, um die Session von QGis über ein Webinterface anzuzeigen. Die Architektur sieht wie folgt aus:



Zeichnung 1: Architektur des Kartenservers mit Qgis-Server eingebunden

Die QGis Konfigurationsdateien haben eine Speichergröße von bis zu 19MB. Der erste Zugriff auf eine Karte erfordert das Parsen der Konfigurationsdatei, was in der QGis-Version 2.10 den Zugriff auf das Webinterface auf 30 Sekunden oder mehr verlängert. 30 Sekunden ist auch der Timeout des Webbrowsers, so dass es in unregelmäßigen Abständen Sessionabbrüche gibt.

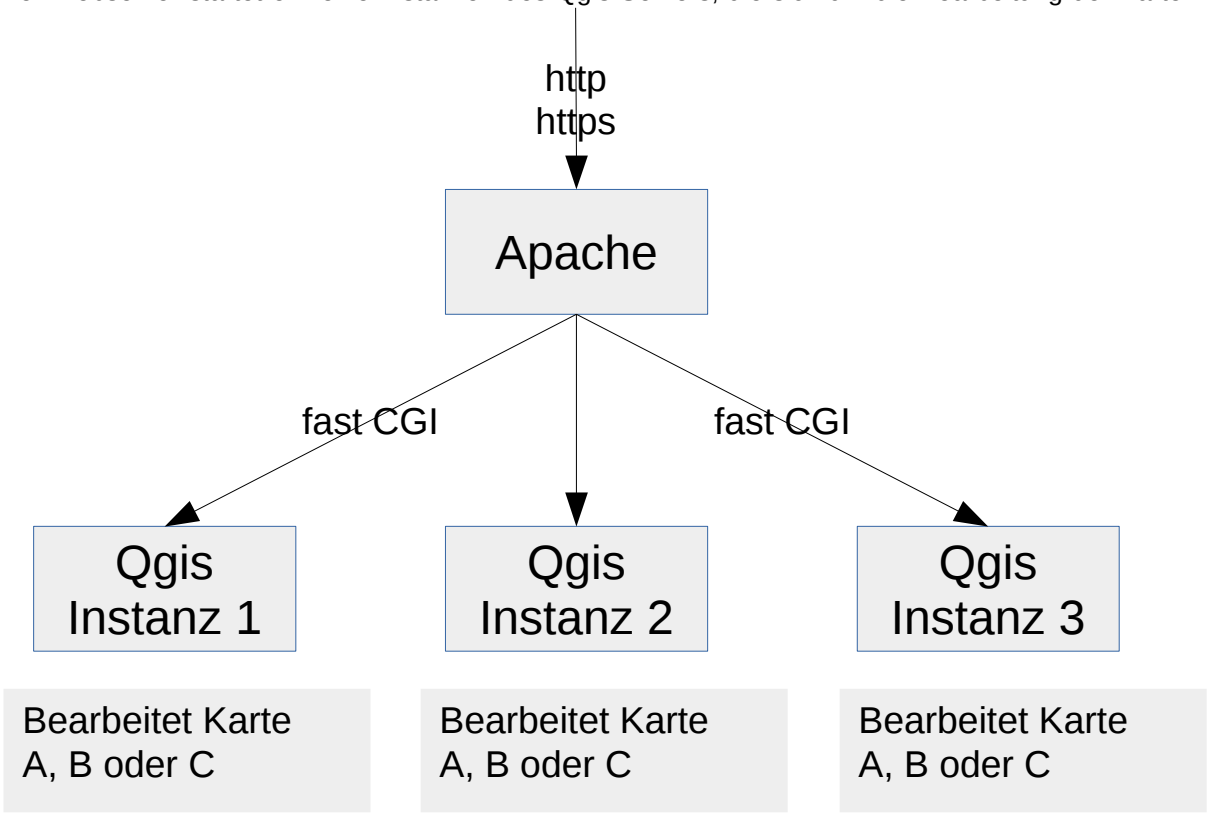
Die Zugriffszeit auf den QGis-Mapserver um Faktor 100 beschleunigen

Eine Analyse der Problematik ergab eine sehr lange Zeit beim Parsen der XML Struktur der Qgis Konfigurationsdatei. Der zweite Zugriff auf dieselbe Karte benötigte nur noch 200 Millisekunden, folgende Zugriffe sogar nur noch 20-50 Millisekunden. Der Qgis-Server speichert also das Ergebnis des letzten Parsedurchgangs der XML Datei als Cache und kann in viel kürzerer Zeit darauf zurück greifen.

Wir können dadurch Sessionabbrüche verhindern, indem wir den ersten Zugriff nicht von unseren Kunden vollziehen lassen, sondern in den Start des Programms vorziehen. Zweitens müssen wir erreichen, dass eine Instanz vom Qgis-Server immer Anfragen für dieselbe Qgis-Konfiguration bekommt, damit die XML-Datei kein zweites Mal geparst wird. Wir können diese zwei Ziele erreichen, indem wir ein spezielles Programm zwischen den Webserver und das Qgis FCGI Programm ziehen.

Eine Standardstruktur zum Starten von Fast-CGI im Webserver sieht so aus, wie auf der Zeichnung 2:

Der Webserver startet einzelne Instanzen des Qgis-Servers, die sich um die Bearbeitung der Karten-



Zeichnung 2: Typische Anordnung vom Apache Webserver mit Fast-CGI Programmen

daten kümmern. Welche Anfrage mit welchem Programm bearbeitet wird, ist nicht vorhersagbar. Wir haben in einem typischen Setup 10 verschiedene Karten für unsere Kunden vorliegen. Ein Webserver, der nicht stark belastet ist, startet ca. 3-4 Instanzen. Dabei kann es passieren, dass eine Qgis Instanz nacheinander verschiedene Karten bearbeiten soll. In dem Fall wird die letzte Information verworfen und die Karte für die zweite Anfrage neu aus der XML-Datei geparst. Dabei kann wieder eine Verzögerung für den Kunden mit einem Sessionabbruch entstehen.

Die Zugriffszeit auf den QGis-Mapserver um Faktor 100 beschleunigen

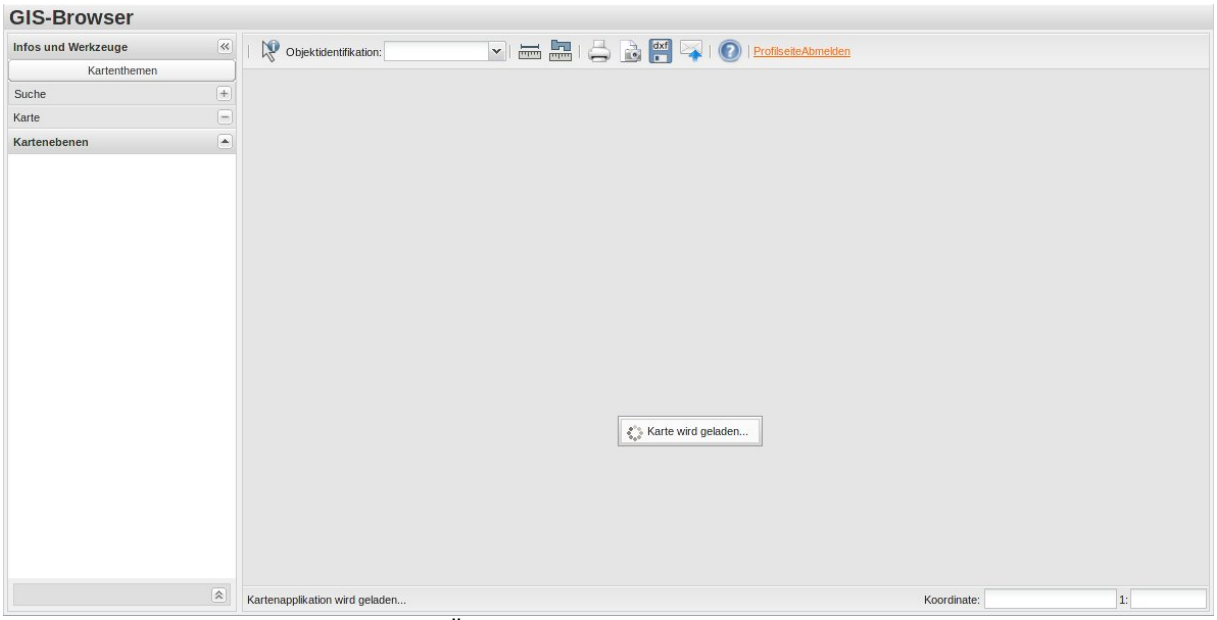
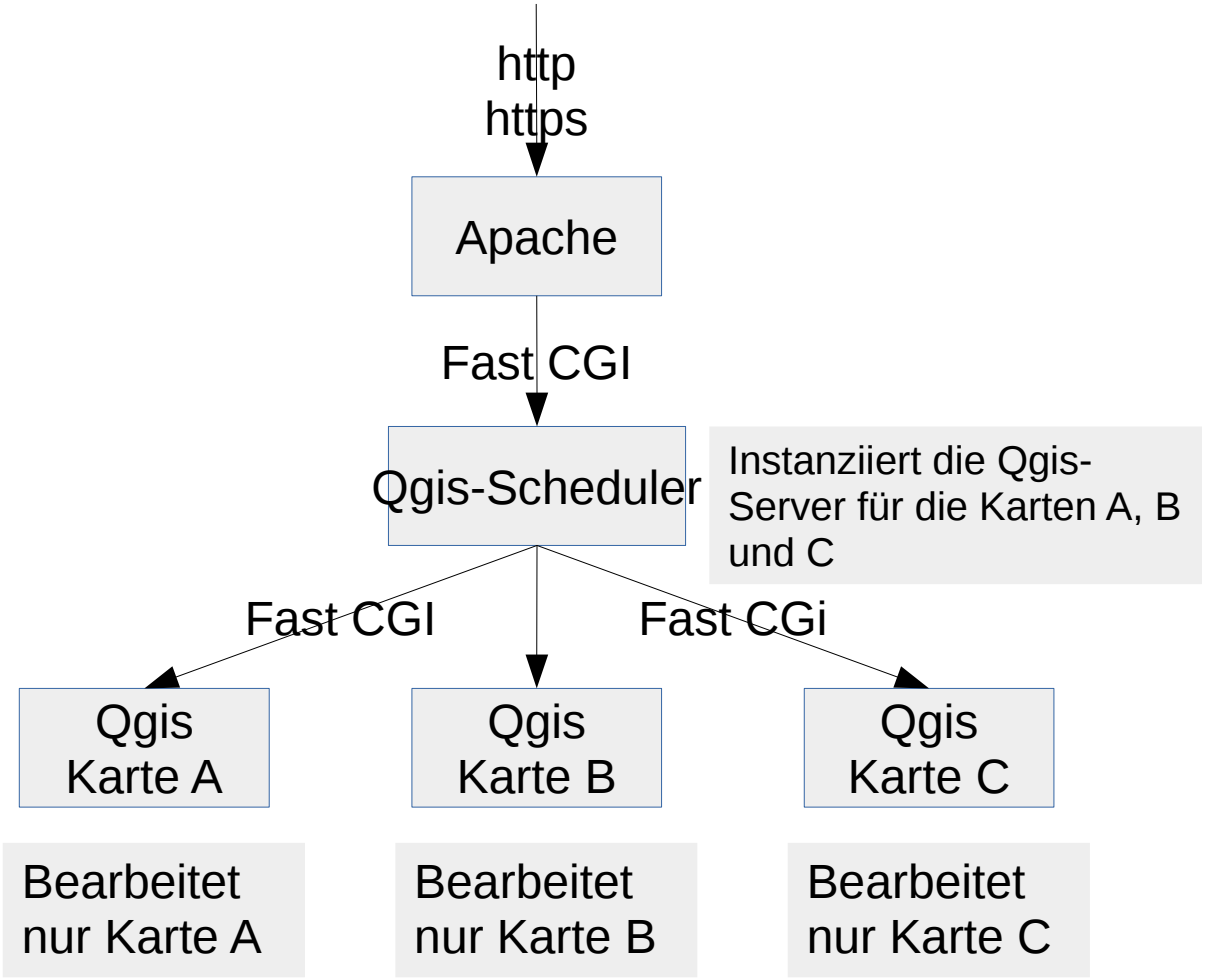


Abbildung 1: Sessionabbruch bei der Übertragung der QGis Karteninformationen

Der Sessionabbruch zeigt sich für den Kunden nur mit einer animierten Warteanzeige („Karte wird geladen..“), die nach dem Abbruch der Datenübertragung unendlich lange läuft. Der Kunde erhält also keine direkte Rückkopplung, ob der Webserver noch Daten überträgt oder die Übertragung schon abgebrochen wurde.

Das Problem verschlimmert sich sogar, wenn der Kunde die Karte erneut anfordert und der vorherige Prozess noch nicht fertig ist. In dem Fall wird ein neuer Prozess gestartet, welcher ebenfalls eine längere Zeit zur Initialisierung benötigt. Im Extremfall kann es sogar dazu führen, dass viele Prozesse unnötigerweise neu gestartet werden, bis der erste Prozess fertig initialisiert ist und die Anfragen vom Webclient korrekt beantworten kann.

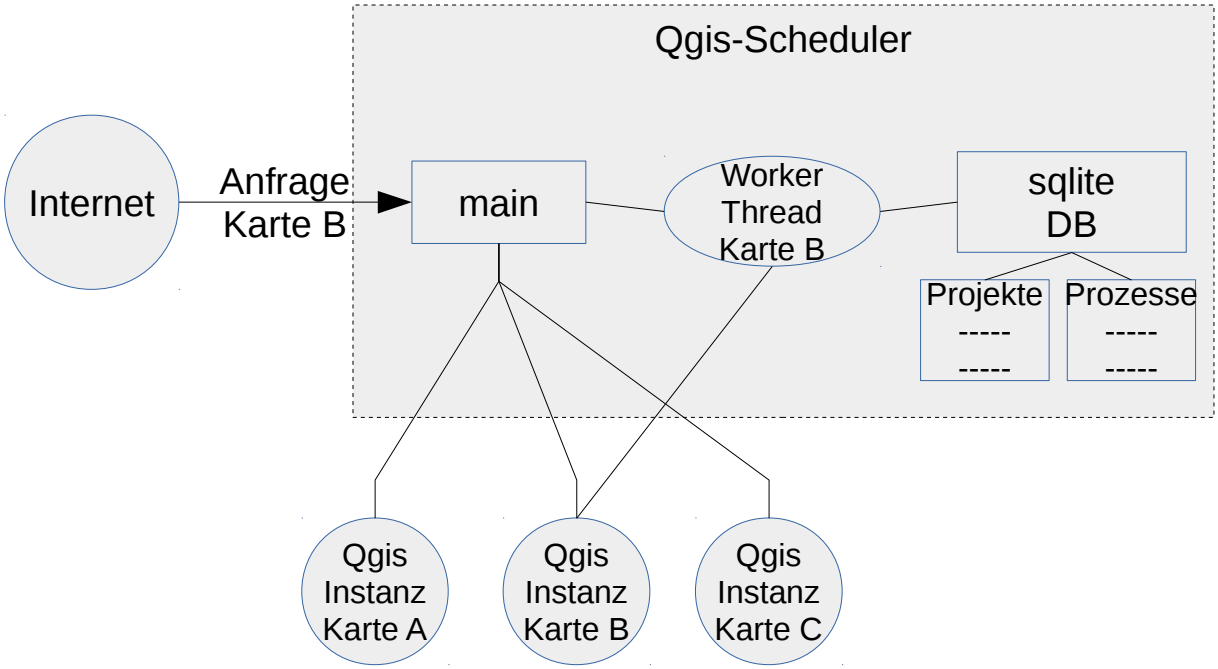


Zeichnung 3: Fast-CGI Webserver mit Scheduler

Wir können jetzt einen Scheduler zwischenschalten, der auf beiden Seiten das Fast-CGI Protokoll bearbeiten kann. Der Scheduler ist für die Instanziierung und Initiierung der Qgis-Server und für die Verteilung der Anfragen auf die Qgis-Server zuständig. Direkt nachdem eine neue Instanz eines Qgis-Server gestartet wurde, wird eine Anfrage an den Qgis-Server gesendet, mit der der Qgis-Server die Konfigurationsdatei parst. Danach ist der Qgis-Server nur für Anfragen mit dieser Konfiguration zuständig. Wenn von einem Webbrowser eine Anfrage kommt, wird sie vom Scheduler geparst und einem Qgis-Server zugeordnet.

Aufbau und Funktionsweise des Tools:

Das Programm ist in ANSI-C geschrieben und benutzt die Bibliotheken fastcgi, iniparser und sqlite. Für die Bearbeitung der Fast-CGI Anfragen werden Threads generiert, die sich um das Verbindungsmanagement und die Auswahl des richtigen Qgis-Prozesses kümmern. Der main-Thread kümmert sich um die Initialisierung der Programmmodule, die Verbindungsanfragen aus dem Internet, das Handling der Signale an den Scheduler und den Modulabbau beim Programmende.



Zeichnung 4: Programmaufbau

In Zeichnung 4 ist das Handling einer Fast-CGI Anfrage schematisch dargestellt. Der main-Thread erhält eine Verbindungsanfrage und startet einen Worker-Thread. Der Worker-Thread liest Teile der Anfrage bis feststeht, welche Karte angefordert wird. Daraufhin wird der entsprechende Qgis-Prozess kontaktiert und die Anfrage an den Prozess weitergeleitet. Die Antwort vom Qgis-Prozess wird durch den Worker-Thread zurück zum Webclient gesendet.

Der Quellcode des Tools steht auf GitHub unter https://github.com/geocalc/qgis-scheduler zur Verfügung.

Setup in Serverumgebungen:

Die Zeichnung 1 zeigt den möglichen Aufbau. Wir haben festgestellt, dass die Instanzen des Qgis-Servers bis zu 1,5GB Speicher verbrauchen können (Qgis Version 2.10), durchschnittlich 0,6GB. Daher ist es in unserem Setup mit vielen verschiedenen Karten sinnvoll, die Qgis-Prozesse auf einen separaten Server auszulagern. Das Fast-CGI Protokoll unterstützt diesen Ansatz eines Applicationservers, weil das Programm ein Socket als Kommunikationseinheit erhält, und dieser Socket auf Basis des Dateisystems oder eines Netzwerks sein darf. Um den Qgis-Scheduler mit netzwerkbasiertem Fast-CGI Protokoll anzusteuern, muss der Webserver das Protokoll netzwerkfähig bereitstellen. Für den Apache Server gibt es das Modul proxy_fcgi, welches für die Kommunikation mit dem Fast-CGI Programm Netzwerksockets anstelle von Dateisystemsockets benutzt. Der Server lighttpd beschreibt ebenfalls ein Setup auf Netzwerkbasis, dieses wurde aber von uns noch nicht erprobt.

In dem vorgeschlagenen Setup sind der Webserver, der Applicationserver und der Datenbankserver getrennte Einheiten, so dass sie für ihre jeweilige Aufgabe spezialisiert werden können. Der Webserver kann schon mit 1 CPU und 4GB RAM ausreichend betrieben werden, da der Webserver die Daten nur weiterleitet an den Applicationserver. Der Datenbankserver benötigt bis zu 4 CPUs und 8-16GB RAM entsprechend des Datenumfangs und sonstiger Verwendung. Der Applicationserver sollte besser 32GB RAM und 8-16 CPUs erhalten. Wobei sich der Speicherverbrauch von Qgis Version 2.6 auf Version 2.14 verbessert hat. In Version 2.14 benötigt ein initialisierter Prozess ca. 350MB und ein uninitialisierter ca. 60MB.

Die Zugriffszeit auf den QGis-Mapserver um Faktor 100 beschleunigen

Durch das Design des Schedulers werden viele Dateizeiger eingesetzt (in Version 0.11.1). Jeder Qgis-Prozess wird mit einem eigenen Socket an den Scheduler angebunden, außerdem benutzt jede Verbindung zum Qgis-Prozess einen Verbindungssocket. Und jeder Worker-Thread erbt alle Dateizeiger im laufenden Betrieb. Es zeigte sich, dass ein Setup mit 20 Karten und 2 Prozessen pro Karte mehr als 1024 Dateizeiger verbraucht. Für einen erfolgreichen Betrieb müssen die Grenzwerte mit dem Tool „ulimit“ heraufgesetzt werden.

Vorteile, Nachteile:

Der Scheduler lässt sich gut in Setups einsetzen, die viele verschiedene Karten enthalten. Oder bei einer höheren Zugriffszahl, um den Start von nicht benötigten Prozessen zu verhindern. Selbst bei einer hohen Zugriffszahl startet der Scheduler neue Prozesse in langsamer Folge, so dass die Gefahr der Überlastung oder Threashing im Applicationserver minimiert wird.

Leider ist der Verbrauch an Dateizeigern recht hoch. Dafür muss im Startskript (init-Skript) der Grenzwert für den Verbrauch an Dateizeigern hoch gesetzt werden. Außerdem kann das Tool den Qgis-Prozess nicht beschleunigen. Als wir in einer Karte alle Layer aktiviert haben, benötigte der Prozess etwas mehr als 5 Sekunden, um das Bild der Karte zu zeichnen. In dem Fall versprechen Verbesserungen am Design vom Qgis-Prozess mehr Erfolg.

Kontakt zum Autor:

Dipl.-Ing. Jörg Habenicht
mWerk GmbH
jh@mwerk.net

Web-basierte Geoprozessierung mit Python und PyWPS

Jonas Eberle

Neben der standard-konformen Bereitstellung von Geodaten wird zunehmend auch die Verarbeitung von Geodaten im Internet immer bedeutender. Mit der Standardisierung der Geodatenverarbeitung durch den „Web Processing Service“ (WPS, [1]) hat das Open Geospatial Consortium (OGC) eine Spezifikation erarbeitet, die es erlaubt, neben den bisher bekannten Spezifikationen zur Datensuche, Datenvisualisierung und Datendownload auch eine Web Service-basierte Verarbeitung den Nutzern anzubieten. So können einerseits grundlegende Funktionen, wie zum Beispiel das Aus- oder Verschneiden verschiedener Vektordaten, realisiert, andererseits aber auch komplexe Algorithmen auf Geodaten angewendet werden. Web-basierte Prozessierungsdienste können daher für eine Vielzahl an Aufgaben verwendet werden und bieten enorme Möglichkeiten, nutzer- und entwicklerfreundliche Geodatendienste und auf Geodaten basierte Web- und Mobil-Anwendungen aufzubauen.

Die WPS-Spezifikation legt grundlegend fest, wie ein Prozess mit ihren Input- und Outputdaten beschrieben wird und wie dieser als Webdienst ausgeführt werden kann. Mittlerweile wurde die Spezifikation schon in unterschiedlichen Softwarepaketen implementiert, wie zum Beispiel dem 52° North WPS, ZOO WPS, deegree WPS, GeoServer WPS oder PyWPS. PyWPS (http://pywps.org, [2]) existiert seit 2006 und basiert auf der Programmiersprache Python. Aktuell ist derzeit noch die Version 3, die die WPS 1.0-Spezifikation unterstützt. Zur Unterstützung der WPS 2.0-Spezifikation wird derzeit an der komplett neu entwickelten Version 4 gearbeitet. Python als Programmiersprache wird oft für die Geodatenverarbeitung verwendet und ist auch gut geeignet, um einen web-basierten Geoprozessierungsdienst aufzusetzen. Es existieren viele Bibliotheken zur Verarbeitung und Analyse von Raster- und Vektordaten sowie Geometrien. Ebenso besitzen viele GIS-Programme Schnittstellen zu Python, wie zum Beispiel QGIS, GRASS GIS oder ArcGIS. Jegliche Python-Scripte können dabei im Rahmen der PyWPS-Software als WPS implementiert und bereitgestellt werden.

Anfrage-Methoden

Die WPS-Schnittstelle besteht in der Version 1.0.0 aus den Methoden „GetCapabilities”, „DescribeProcess” und „Execute”, die im Folgenden kurz beschrieben sind. Ausführliche Beschreibungen befinden sich in der Spezifikation [1] selbst.

GetCapabilities:

> Wie jeder OGC-konforme Dienst muss auch die WPS-Spezifikation eine „GetCapabilities”-Methode implementieren. Diese Methode liefert neben allgemeinen Metadaten spezifische Daten über die implementierten Prozesse, die zur weiteren Ausführung und Beschreibung der jeweiligen Prozesse notwendig sind.

DescribeProcess:

> Die „DescribeProcess”-Methode liefert eine Beschreibung für einen oder mehrere angefragte Prozesse. Die Beschreibung enthält Angaben zu sämtlichen Input- und Outputdaten sowie deren Datenformaten. Mit diesen Angaben ist es möglich, die Anfrage zum Ausführen des Prozesses zu generieren.

Execute:

> Basierend auf den anzugebenden Parametern aus dem „DescribeProcess”-Dokument kann ein bestimmter Prozess über die Methode „Execute” ausgeführt werden. Inputdaten können in der Anfrage direkt angegeben oder als URL übergeben werden. Das Ergebnis des Prozesses kann entweder direkt in das XML-Antwortdokument eingebettet (ResponseDocument) oder als

Rohdaten zurückgeliefert werden (RawDataOutput). Bei den Outputdaten im XML-Antwortdokument kann bei der Ausführung angegeben werden, ob diese auf dem WPS-Server gespeichert werden sollen (storeExecuteResponse) oder nicht. Um einen Prozess asynchron auszuführen, muss noch der Parameter „status“ gesetzt werden. Im Falle einer asynchronen Ausführung wird sofort ein XML-Antwortdokument mit Angabe einer URL zurückgeliefert, unter der das Ergebnis-XML zu erwarten ist sowie der aktuelle Status des Prozesses (failed, completed, paused, accepted, started) abgefragt werden kann.

Prozessbeschreibung

Ein Prozess besteht aus Eigenschaften, Input- und Outputdaten sowie dem Modell bzw. der Berechnung. Ein „Identifier” sorgt für die eindeutige Kennzeichnung, ein Titel und ein „Abstract” für eine Beschreibung des Prozesses. Jeder Prozess besitzt eine Versionsnummer und kann mit weiteren Metadaten im Rahmen eines Links näher beschrieben werden. Zusätzlich kann beim Prozess angegeben werden, ob asynchrone Anfragen inklusive Status-Abfrage und das Speichern der Outputdaten auf dem Server erlaubt sind.

Die Input- und Outputdaten besitzen jeweils eine eindeutige Kennzeichnung sowie einen Titel und eine Kurzbeschreibung. Über bestimmte Einstellungen lässt sich festlegen, ob Input- oder Outputdaten mehrmals vorkommen dürfen. Input- und Outputdaten können mit einer der folgenden Datentypen beschrieben werden:

ComplexData:

Unter ComplexData sind Daten mit einer komplexen Datenstruktur zu verstehen, wie zum Beispiel ein XML-Dokument (z.B. Vektordaten als GML) oder Binärdaten. Diese Daten können entweder direkt in der Anfrage bzw. im Outputdokument enthalten sein, oder als Referenz (URL) übergeben werden. Zusätzlich kann ein Metadaten-Tripel aus Format (obligatorisch, z.B. „text/xml” oder „application/x-hdf”), Kodierung (optional, z.B. „UTF-8”) und XML-Schema (optional, z.B. „http://schemas.opengis.net/gml/3.2.1/gml.xsd”) angegeben werden. Die angegebenen Inputdaten werden im PyWPS beim Ausführen des Prozesses heruntergeladen und auf dem Server zur weiteren Verarbeitung gespeichert.

LiteralData:

LiteralData sind primitive Datentypen wie zum Beispiel Ganzzahlen, Fließkommazahlen oder Zeichenketten. Dieser Input kann auf bestimmte Werte oder einen Wertebereich eingeschränkt werden. Zusätzlich kann das Datenformat und die Kodierung angegeben werden.

BoundingBoxData:

Die BoundingBox beinhaltet vier Koordinaten (Min-X, Min-Y, Max-X, Max-Y) mit Angabe des räumlichen Bezugssystems zur Kennzeichnung eines zu betrachtenden Gebietes.

Beispielanfragen

Die folgenden Beispielanfragen können in den Webbrowser eingegeben werden. Hinweis: Beim Kopieren müssen sämtliche Leerzeichen vor dem Ausführen entfernt werden.

```
http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=1.0.0&
request=GetCapabilities
```

```
http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=1.0.0&
request=DescribeProcess&identifier=1013_single_ts_plot_point
```

```
http://artemis.geogr.uni-jena.de/cgi-bin/testbox.cgi?service=WPS&version=1.0.0&
request=Execute&identifier=1013_single_ts_plot_point&datainputs=
[datasetName=mod13q1_evi;pointX=13.54;pointY=52.31]
&status=true&storeExecuteResponse=true     (optional)
```

Python-basierter Prozess mit PyWPS

Ein PyWPS-Prozess basiert auf den Eigenschaften (Metadaten) des Prozesses, der Beschreibungen der Prozessinputs und -outputs sowie der Funktion zur Ausführung des Prozesses (siehe Abbildung 1). Die Hauptmethode „def execute(self):“ in Abbildung 2 projiziert eine Geometrie von der Quell- zur Zielprojektion, angegeben durch die jeweiligen EPSG-Nummern. Es sind dabei drei Inputdaten vorhanden: 1) Die Geometrie als Zeichenkette im Well-Known-Text (WKT)-Format sowie 2) die Quell- und und 3) die Zielprojektion als Integer-Werte. Als Output wird die Geometrie erneut im WKT-Format als Zeichenkette zurückgegeben. In der Hauptmethode wird die Python-Bibliothek GDAL/OGR verwendet, um die Projektion der Geometrie durchzuführen. Möglich wären hier auch die Verwendung von weiteren Bibliotheken sowie der Aufruf von Kommandozeilenprogrammen zur Verarbeitung der Daten.

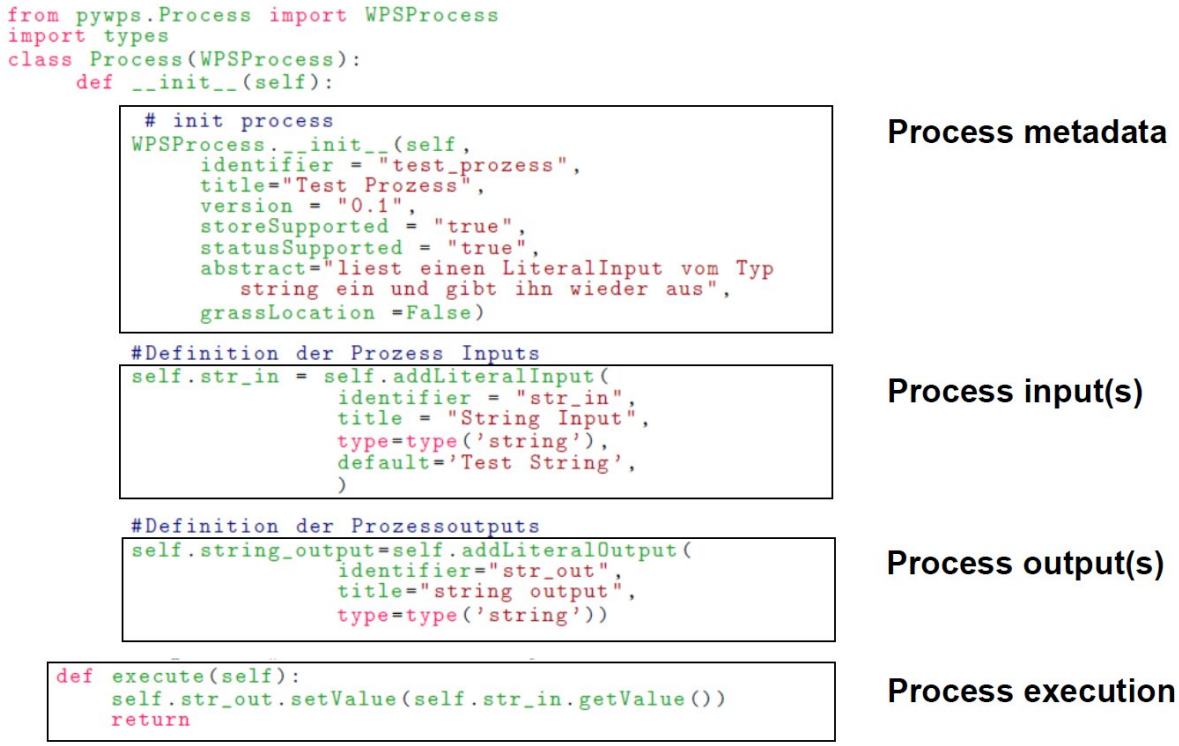


Abbildung 1: Beispielhafter Aufbau eines Prozesses mit der PyWPS-Software (Version 3).

```
    def execute(self):

        # get input values
        wkt = self.wkt.getValue()
        epsg_source = int(self.epsg_source.getValue())
        epsg_target = int(self.epsg_target.getValue())

        osr.UseExceptions()

        # Achtung: Zur Uebersicht werden in diesem Beispiel keine Exceptions aufgefangen!
        source = osr.SpatialReference()
        source.ImportFromEPSG(epsg_source)

        target = osr.SpatialReference()
        target.ImportFromEPSG(epsg_target)

        geom = ogr.CreateGeometryFromWkt(wkt)
        transform = osr.CoordinateTransformation(source, target)
        geom.Transform(transform)

        output_wkt = geom.ExportToWkt()

        # return output
        self.output.setValue(output_wkt)
```

Abbildung 2: Execute-Methode zur Projektion von Geometrien.

Weiterführende Links

- PyWPS Webseite: http://pywps.org
- PyWPS 3 Dokumentation: http://geopython.github.io/pywps/doc/build/html/
- Tutorial für PyWPS 3: http://jachym.github.io/pywps-tutorial/build/html/index.html
- PyPI: https://pypi.python.org/pypi/pywps
- GitHub (PyWPS 3): https://github.com/geopython/pywps/tree/pywps-3.2

Kontakt zum Autor:

Jonas Eberle

Friedrich-Schiller-Universität Jena, Lehrstuhl für Fernerkundung

Loebdergraben 32, 07743 Jena

+49 3641 94 88 89

jonas.eberle@uni-jena.de

http://www.eo.uni-jena.de

Literatur

[1] Schut, P., OpenGIS Web Processing Service; Open Geospatial Consortium Inc., 2007.

[2] Cepicky, J.; Becchi, L., Geospatial processing via Internet on remote servers-PyWPS, OSGeo Journal, 1(5), 11-17, 2007.

Hybride mobile App-Entwicklung mit Angular

Arne Schubert

Die hybride App-Entwicklung bietet viele Vorteile, da Apps plattformübergreifend (Android, iOS usw.) erstellt werden können und somit der Entwicklungsaufwand minimiert wird.

Ein sehr modernen Ansatz bietet Angular. Durch seine gradlinige Implementierung als MVC Framework (Model, View, Controller) ist die Anwendungslogik klar strukturiert und durch das so genannte „two way data binding“, über einfache JavaScript Objekte, werden Model, View und Controller durch Angular verknüpft, ohne dass dies durch den Entwickler geschehen muss. Dies ermöglicht ein sehr hohes Maß an Modularität bei wenig Programmieraufwand.

Gleichzeitig ist es möglich weitere Frameworks, durch Dependency Injection, in die Logik von Angular zu integrieren. Für viele dieser Frameworks existieren sogar bereits Implementierungen, wie z.B. für Bootstrap, Leaflet oder auch pouchDB, wodurch die Entwicklung mit Angular äußerst flexibel und einfach wird und auch im Geo Bereich bereits viel zu bieten hat.

Durch eine große Auswahl an Modulen mit mobile first UI-Elementen, die sowohl die Grundsätze von User Experience als auch Responsive Design erfüllen, ist Angular zum Mittel der Wahl bei hybriden Applikationen geworden.

Der Vortrag gibt zunächst eine kurze Einführung in die Grundbegriffe der hybriden App-Entwicklung und Angular. Es wird ein Grundsystem für mobile App-Entwicklung erstellt und um Webmapping Elemente, durch die Angular Leaflet Directive, erweitert. Das Ergebnis kann dann in einer Vorführanwendung direkt betrachtet werden.

Alle verwendeten Frameworks sind natürlich Open-Source.

Da das Thema hybride App-Entwicklung sehr Umfangreich ist, ist der Vortrag als eine Einführung in die Thematik gedacht, die die Vorzüge von hybriden Webmapping Apps mit Angular aufzeigen soll.

mapmap.js – Ein kartographisches API für interaktive thematische Karten

Florian Ledermann, TU Wien

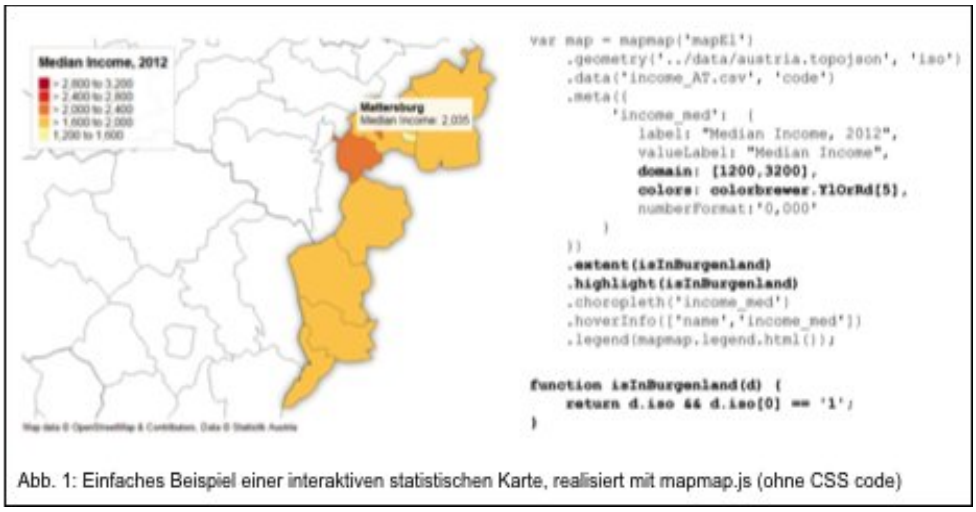


Abb. 1: Einfaches Beispiel einer interaktiven statistischen Karte, realisiert mit mapmap.js (ohne CSS code)

Hintergrund

Beim Programmieren von Web-Karten bewegt man sich derzeit zwischen zwei Polen: entweder mittels "Slippy Map" APIs [1] wie z.B. Leaflet [2] eine Hintergrundkarte aus Bildkacheln laden, oder mit Hilfe von low-level Visualisierungswerkzeugen wie D3.js [3] komplett dynamische, vektorbasierte Karten und Diagrammdarstellungen im Browser erstellen. Im ersten Fall sind der Interaktion mit den Karteninhalten Grenzen gesetzt, da nicht alle Karteninhalte als geometrische Objekte verfügbar sind und lediglich „Overlays“ über eine am Server erstellte statische Hintergrundkarte gelegt werden können, im zweiten Fall ist selbst für vergleichsweise einfache Kartendarstellungen umfangreicher und komplexerer Code notwendig. Einen möglichen Mittelweg weist die jüngere Entwicklung von “Vector Tiles” [4], die ein Verfahren zur kachelbasierten Übertragung von Geometriedaten (anstatt Kartenbildern) vorstellen – zunächst einmal bieten Vector Tiles aber lediglich ein Verfahren zur Übertragung von Geometriedaten und lassen die Frage, wie diese am Client zu visualisieren sind, unbeantwortet.

Anforderungen an ein browserbasiertes kartographisches API

Das an der TU Wien entwickelte API mapmap.js [5] versucht den kartographischen Visualisierungsprozess in seiner Gesamtheit in einem high-level JavaScript API abzubilden, wobei jeder Teilaspekt zunächst „einfach funktioniert“, aber bei Bedarf im Detail an die Notwendigkeiten der jeweiligen Anwendung angepasst werden kann. Die kartographische Visualisierungspipeline – vom Laden und Zusammenführen von Daten, Metadaten und Geometrie, über Projektion, Geometriemodifikation und Generalisierung, dem Zuweisen graphischer Repräsentationen und visueller Attribute bis zur Spezifikation von User-Interaktion – ist dabei im API abgebildet. Das API erlaubt die Erstellung von interaktiven Karten mit etablierten Techniken in wenigen Zeilen, aber auch die Anpassung jeden Schrittes der Visualisierungspipeline im Detail und somit die Entwicklung völlig neuer kartographischer und hybrider Visualisierungstechniken.

Beim Design von mapmap.js wurden folgende Anforderungen definiert:

- Es soll den *kartographischen* Prozess in seiner Gesamtheit abbilden – das Ziel ist ein *horizontales* API das vom Datenzugriff bis zu Rendering und Interaktion alle Details abbildet.
- Es soll ein „*high level*“ API sein, das irrelevante Komplexität abstrahiert und ausblendet (von Tulach [6] „selective cluelessness“ genannt)

- Gleichzeitig soll es *transparent* sein, d.h. das jeder Teilaspekt bei Bedarf im Detail kontrolliert und angepasst werden kann.
- Das API soll *Daten- und Formatagnostisch* sein, d.h. es soll nicht die Verwendung von bestimmten Datenformaten oder –strukturen erzwingen. Adaptermodule für neue Formate sollen einfach einzubinden sein.

Als Leitsätze stehen für diese Anforderungen: „Simple things should be simple, complex things should be possible“ (Alan Kay) und „Batteries Included“ (Python Motto). Weitere Details zur Anforderungsanalyse finden sich in [5].

Implementierungsdetails

Die Gestaltung des API in JavaScript ist inspiriert durch an moderne Konzepte der funktionalen und datengetriebenen Programmierung [3] sowie populärer JavaScript-Konstrukte wie method chaining, closures oder promises. Eine triviale statistische Karte des Medianeinkommens in Österreich pro Bezirk lässt sich in mapmap.js per *method chaining* pattern so implementieren:

```
mapmap("#mapEl").geometry('AT.topojson','iso').data('income.csv','nuts')
  .select('bezirke').choropleth('income_med');
```

Dabei sind die notwendigen Schritte nur an der Oberfläche trivial – um die Karte zu realisieren, müssen das Laden von mehreren Ressourcen koordiniert, die Daten mit der Geometrie zusammengeführt sowie eine geeigente Skala für die Choroplethendarstellung aus den Daten abgeleitet werden. Alle Details können jedoch bei Bedarf auch im Detail angepasst werden, und Standard-Symbolisierungen wie Choroplethenkarten können durch eigene Algorithmen ersetzt werden. (Für ein etwas erweitertes Beispiel einer interaktiven statistischen Karte, siehe Abb. 1.)

Komplexe technische Details werden vom API abstrahiert, um im Code den kartographischen Prozess und kartographische Entscheidungen klarer werden zu lassen – Beispielsweise wird das asynchrone Laden von mehreren Ressourcen mittels Promises so implementiert, das ein sequentieller Programmierstil ohne Callbacks angewendet werden kann (siehe oben). Das Filtern und Zusammenführen von Geometrie und Daten wird per MapReduce-Modell [7] in modularer Weise unterstützt.

Ergebnisse und Ausblick

Im Projekt „genderATlas“ [8] wurde mapmap.js erfolgreich eingesetzt, um eine Vielzahl unterschiedlicher thematischer Karten zu realisieren. Dabei reicht die Bandbreite von einfachen animierten Choroplethenkarten über eigens erstellte Symbolisierungstechniken (z.B. mittels parametrisierbarer Füllmuster) bis hin zur Darstellung von Pendlerverflechtungen mittels Flow Maps. Der Atlas konnte so als durchaus heterogene Sammlung von thematischen Karten konzeptioniert werden, in Kontrast zu manchen anderen Projekten, in denen einer Sammlung von Daten oft ein einheitliche Interface- und Visualisierungskonzept „übergestülpt“ wird.

Außerhalb unseres eigenen Wirkungsbereichs haben einige Institutionen und Forschungsgruppen Interesse an einem Einsatz von mapmap.js gezeigt. Besondere Hoffnungen setzen wir auf den Einsatz von mapmap.js in der Lehre, da auch Nicht-ProgrammiererInnen durch den modularen Aufbau ermöglicht wird, zunächst aus vorgefertigten Bausteinen funktionierende Karten zusammenzustellen und dann Schritt für Schritt in die Anpassung durch Programmierung einzusteigen (z.B. zunächst durch die Implementierung einer einfachen Filter- oder Konvertierungsfunktion sowie später über die Implementierung von Generalisierungs- oder Symbolisierungstechniken). Wir planen den Einsatz in der Lehre and der TU Wien und eine begleitende Evaluierung mit der Fertigstellung der nächsten major Release.

mapmap.js steht derzeit in der Version 0.2.7 – für die nächste „major Release“ 0.3 (also gemäß den Prinzipien von „semantic versioning“ immer noch als „unstable“ bzw. experimentell anzusehen) sind einige fundamentale Erweiterungen und auch Änderungen am API geplant. Unter anderem soll das Modell des Daten- und Geometriefluss überarbeitet werden um noch flexiblere Visualisierungstechniken

zu unterstützen, sowie neue Datenquellen wie Vector Tiles oder Rasterdaten implementiert werden. Die Veröffentlichung von Release 0.3 ist für Herbst 2016 geplant.

Kontakt zum Autor:

DI Florian Ledermann

TU Wien, Department für Geodäsie und Geoinformation, FG Kartographie

Erzherzog-Johann-Platz 1/120-6

florian.ledermann@tuwien.ac.at

http://cartography.tuwien.ac.at/

Literatur

[1] J. T. Sample und E. Ioup, „Introduction“, in *Tile-Based Geospatial Information Systems*, Springer US, 2010, S. 1–3.

[2] V. Agafonkin, „Leaflet — an open-source JavaScript library for interactive maps“, 2010. [Online]. Verfügbar unter: http://leafletjs.com/. [Zugegriffen: 09-März-2015].

[3] M. Bostock, V. Ogievetsky, und J. Heer, „D3: Data-Driven Documents“, *IEEE Transactions on Visualization and Computer Graphics*, Bd. 17, Nr. 12, S. 2301–2309, Dez. 2011.

[4] J. Gaffuri, „Toward Web Mapping with Vector Data“, in *Geographic Information Science*, N. Xiao, M.-P. Kwan, M. F. Goodchild, und S. Shekhar, Hrsg. Springer Berlin Heidelberg, 2012, S. 87–101.

[5] F. Ledermann und G. Gartner, „mapmap.js: A Data-Driven Web Mapping API for Thematic Cartography“, in *Proceedings of the 27th International Cartographic Conference (ICC2015)*, Rio de Janeiro, Brasil, 2015.

[6] J. Tulach, *Practical API Design: Confessions of a Java Framework Architect*. New York: Apress, 2008.

[7] J. Dean und S. Ghemawat, „MapReduce: Simplified Data Processing on Large Clusters“, *Communications of the ACM*, Bd. 51, Nr. 1, S. 107–113, Jän. 2008.

[8] M. Riegler, M. Wenk, E. Aufhauser, F. Ledermann, M. Schmidt, und G. Gartner, „genderATlas Österreich – Entwicklung eines zielgruppenorientierten Online-Tools“, *Mitteilungen der Österreichischen Geographischen Gesellschaft*, Bd. 157, S. 323–339, 2016.

OpenLayers 3: Stand, Neues und Ausblick

Marc Jansen

OpenLayers [1], als hoch performante und vielfältige JavaScript Bibliothek für moderne Kartenanwendungen im Web bekannt, hat sich seit der letzten FOSSGIS Konferenz im März 2015 [2] -- OpenLayers v3.3.0 -- stetig weiterentwickelt. Zur FOSSGIS 2016 wird die aktuelle Version aller Voraussicht nach v3.18.0 sein.

Zunächst wird der Vortrag OpenLayers kurz vorstellen, um dann einen Schwerpunkt auf die Änderungen seit März 2013 zu legen. Abschließend soll kurz dargestellt werden, welche zukünftigen Entwicklungen derzeit bearbeitet werden und wie der Stand von häufig nachgefragten Features ist. Wann ist etwa mit der Unterstützung von LineStrings und Polygonen im WebGL-Renderer zu rechnen? Wann (falls überhaupt) wird OpenLayers einen vollfunktionalen SLD-Parser bereitstellen? Und was bleibt noch an der Bibliothek zu entwickeln, nun da sogar Rasterdaten im Browser zur Laufzeit reprojiziert werden?

Faster, smaller, better: Compiling your application together with OpenLayers 3

Tobias Sauerwein

OpenLayers 3 setzt den Closure Compiler ein, um JavaScript in besseres JavaScript zu kompilieren. Der von Google entwickelte Closure Compiler macht weit mehr als normale Code-Minifier: Es werden nicht nur Variablen- oder Funktionsnamen gekürzt, durch die statische Analyse des Codes werden eine Reihe von Optimierungen durchgeführt, wie zum Beispiel das Entfernen von nicht verwendetem Code oder Funktions-Inlining. Besonders interessant ist das Type-Checking und auch ein Syntax-Check, so dass viele Fehler, die sonst erst während der Ausführung auffallen würden, schon früh entdeckt werden.

Man kann OpenLayers 3 verwenden ohne mit dem Closure Compiler in Berührung zu kommen. Kompiliert man seine Anwendung allerdings zusammen mit OpenLayers, kommt man in den Genuss einiger Vorteile. Zuallererst, da der Compiler nicht verwendeten Code entfernt, wird nur der Teil von OpenLayers mit eingebunden, der tatsächlich in der eigenen Anwendung verwendet wird. Da oft nur ein Bruchteil der umfangreichen Funktionalität von OpenLayers benötigt wird, kann so die Build-Größe und damit auch die Ladezeit erheblich reduziert werden. Die Kompilierung zusammen mit OpenLayer macht es auch leichter OpenLayers durch eigene Komponenten zu erweitern. Und nicht zuletzt wird natürlich auch der Anwendungscode durch den Closure Compiler analysiert und überprüft, so dass man zum Beispiel vom Type-Checking profitieren kann.

Dieser Vortrag stellt den Closure Compiler vor, der eine robuste Plattform zur Entwicklung komplexer Anwendungen mit OpenLayers bietet.Vorteile, Besonderheiten und Erfahrungen aus Projekten werden angesprochen.

GeoExt3

Marc Jansen

Der Vortrag stellt GeoExt 3 [1] vor und wird auch die Vater-Bibliotheken ExtJS [2] und OpenLayers [3] erläutern. Schwerpunkte werden hier zunächst allgemeine Features der Bibliotheken / Frameworks sein, bevor der Fokus auf der Erstellung von 'universalen' WebGIS-Applikationen liegt. Unter 'universal' verstehen wir hierbei eine GeoExt3-basierte Applikation, die sowohl auf klassischen Desktop-Browsern aber auch auf mobilen Endgeräten wie Tablets und Smartphones funktioniert und ein ansprechendes Benutzererlebnis ermöglicht.

Insbesondere seit der GeoExt3 zugrundeliegenden Version 6 von ExtJS kann man aus einer einzigen Codebasis solche Applikationen erstellen, ohne jede Funktionialität doppelt entwickeln zu müssen. OpenLayers 3 verfolgt bereits seit den ersten Entwicklungen die Unabhängigkeit vom gewählten Webbrowser und Endgerät.

Im Vortrag wird also die OpenSource-Bibliothek GeoExt3 vorgestellt und ein konkretes Anwendungsbeispiel beleuchtet. Der Vortrag wird die Rahmenbedingungen von universalen WebGIS-Applikationen nennen und zeigen, wie die Bibliotheken und Entwicklungstools (etwa Sencha Cmd [4]) helfen, die konkreten Anforderungen an die jeweiligen Gegebenheiten zu erfüllen.

Die Vortragenden Christian Mayer (meggsimum) und Marc Jansen (terrestris) sind beide Kernentwickler und Mitglieder des Projektsteuerungskommitees von GeoExt.

Datenerfassung und Suchen mit Mapbender3

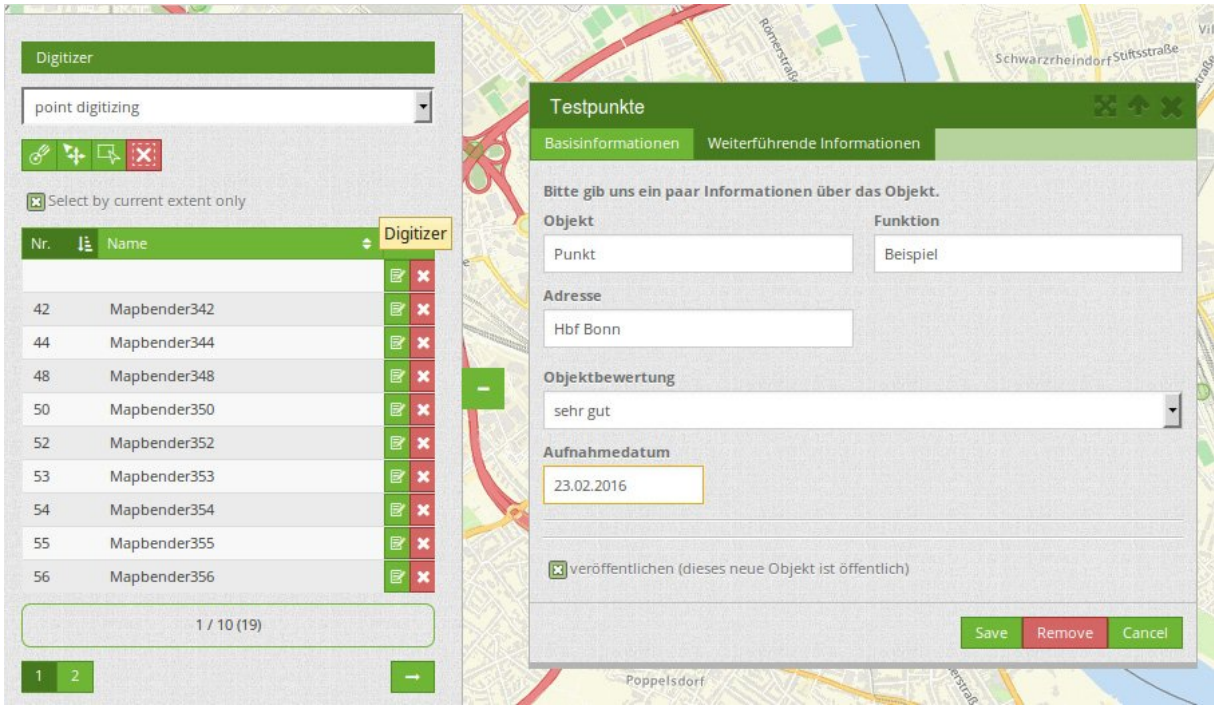
Astrid Emde, Charlotte Toma

Mapbender3 ist eine WebGIS Client Suite zur einfachen Erstellung von WebGIS Anwendungen. Die Software stellt eine Vielzahl an Funktionalität zur Verfügung. Im Folgenden sollen die Datenerfassung und Suche mit Mapbender3 beschrieben werden. Die leichte Konfigurierbarkeit ermöglicht es in ein paar Schritten Suchen und Datenerfassung auf den eigenen Daten bereitzustellen.

Datenerfassung mit dem Digitizer

Seit der Version 3.0.5.0 steht mit dem Element Digitizer die Möglichkeit zur Verfügung, Daten über Mapbender3 zu erfassen. Das Element wurde lang ersehnt und erfreut sich schon jetzt bei den Anwendern großer Beliebtheit.

Mit Digitizer kann durch eine YAML-Definition eine Erfassungsmaske für Punkte, Linien oder Flächen aufgebaut werden. Dabei wird bisher PostgreSQL (SQL) als Datenquelle unterstützt. Die Entwicklung wurde so durchgeführt, dass die Erfassung auf andere Datenquellen wie z.B. OGC WFS erweitert werden kann. Die ausführliche Dokumentation zum Digitizer-Element finden Sie in der Dokumentation auf der Mapbender3-Webseite [1]. Hier finden Sie auch ein Beispiel für den Aufbau einer eigenen Digitalisierung mit Testdaten. Wenn Sie die Erfassung ausprobieren möchten, können Sie dies über die Digitize Demo tun [2].



Mit dem Digitizer können für die Erfassung von Sachdaten sehr komplexe Formulare generiert werden. Hierbei haben wir sehr elegante Möglichkeiten, um komplexe Formulare z. B. mit Reitern und Elementgruppierungen aufzubauen.

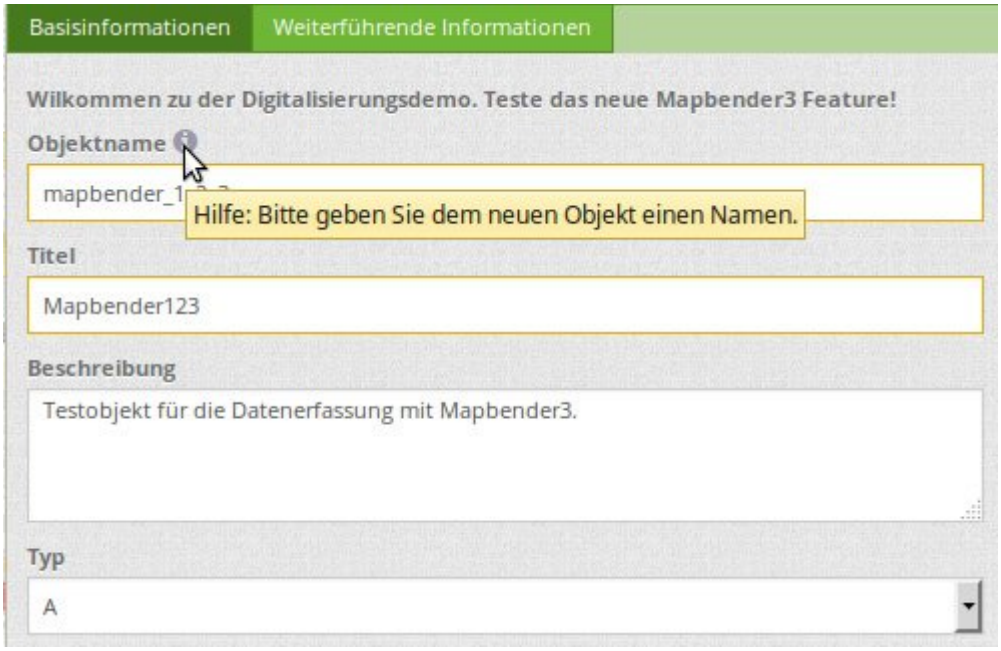
Datenerfassung und Suchen mit Mapbender3

Der neue Digitizer bietet für die verschiedenen Objekte unterschiedliche Erfassungsmöglichkeiten:

- Erfassen von Punkten, Linien und Flächen (Rechteck, Polygon, Kreis und Ellipse)
- Verschieben von Objekten
- Einfügen und Verschieben von Stützpunkten (Linien, Flächen)
- Erzeugung von Enklaven (Donuts)
- Löschen von Objekten
- Löschen von Stützpunkten

Pro Thema kann bestimmt werden, welche Felder zur Verfügung stehen soll. Sie können die Definition leicht im YAML-Syntax setzen.

Hier ein Beispiel für das Eingabefeld:




```
- type: input
  title: Objektname
  name: title
  mandatory: true
  mandatoryText: Bitte geben Sie dem POI einen Namen.
  infoText: "Hilfe: Bitte geben Sie dem neuen Objekt einen Namen."
```

Folgende Möglichkeiten stehen für den Aufbau von Formularen zur Verfügung:

- Definition von mehreren Datenquellen für die Erfassung (werden über eine Selectbox zur Auswahl angeboten)
- Als Datenquelle kann eine Tabelle angesprochen werden, wobei auch nur eine Auswahl der Daten über einen Filter herangezogen werden kann.
- Textfelder
- Selectboxen, Multiselectboxen (Füllen der Auswahlbox über eine feste Definition von Werten in der YAML-Definition oder über ein Select auf eine Tabelle)
- Radiobuttons, Checkboxen
- Textblöcke
- Datumsauswahl
- Dateiupload
- Definition von Reitern
- Definition von Trennlinien
- Definition von beschreibenden Texten
- Pflichtfelder, Definition von regulären Ausdrücken für die Formatvorgabe für den Inhalt von Feldern
- Hilfetexte

Der Digitizer kann über die Seitenleiste (Sidepane) aufgerufen werden. Sie könnten den Digitizer aber auch über einen Button in der Toolbar ansteuern. Über eine Checkbox können Sie steuern, ob Sie nur die Objekte des aktuellen Kartenausschnitts sehen möchten oder alle Objekte. Durch die Einschränkung auf den Kartenausschnitt bekommen Sie eine gute Übersicht über die einzelnen Punkte im Ausschnitt. Die Objekte werden in einer Tabelle aufgeführt. Sie können über die YAML-Definition bestimmen, welche Spalten in der Tabelle erscheinen sollen. Die Spalten sind sortierbar. Per Klick auf eine Zeile springen Sie zum Objekt in der Karte. Je nach Konfiguration haben Sie in jeder Zeile Schaltflächen zur Bearbeitung und zum Löschen.

Zur Erfassung und Bearbeitung können Sie Funktionalitäten aktivieren. Für Punkte können Sie beispielsweise die folgenden Aktionen aktivieren:

- Punkt verschieben
- Punkte erzeugen
- Punkt löschen
- Punkt auswählen

Das Formular

Nach dem Zeichnen eines Objektes öffnet sich sofort ein Formular, in das die Sachdaten zum Objekt eingegeben werden können. Hierbei werden Pflichtfelder farblich hervorgehoben. Über „mandatory“ können Sie auch Regeln für die Eingabe definieren. Weiterhin können Sie Hilfetexte für jedes Feld definieren. Dies erfolgt über „infoText“. Hilfetexte können auch mehrzeilig sein.

Das Formular kann über mehrere Reiter verfügen, die in der YAML-Definition leicht über „type:tabs“ angelegt werden konnten.

Alle Strukturen, die Sie erzeugen, gehören zu einem Typ. In diesem Fall „type:input“. Über „label“ können Sie das Textfeld mit einer Beschriftung versehen. Über „name“ geben Sie jeweils das zugehörige Feld der Datenstruktur an (hier eine Spalte aus einer PostgreSQL-Tabelle). Ein Textfeld für etwas längere Texte können Sie leicht über den „type:textArea“ definieren. Hier können Sie über „rows“ die Größe angeben. Für jedes Feld können Sie eigene Stilangaben machen. So können Sie beispielsweise definieren, dass ein Feld nur einen bestimmten Bereich belegen soll. Die Angabe „type:label“ definiert dabei einen freien Text, der im Formular angezeigt wird.

Anstehende Entwicklungen und Ausblick

Das Element Digitizer bietet schon jetzt enorme Möglichkeiten. Dennoch gibt es noch viele Erweiterungen, die bereits in der Entwicklung sind bzw. denkbar sind.

In den nächsten Mapbender3 Versionen erwarten Sie die folgenden Funktionen:

- Filterung der Anzeige/Erfassung nach Mapbender Anwender oder Gruppe
- Suche in den Daten
- Erfassung von Objekten ohne Geometrie
- Kontextmenü
- Clustering von Objekten
- Aufruf von Skripten nach der Erstellung von Objekten
- Aktion vor/nach dem Speichern (AfterInsert)

Weitere Ideen für die Entwicklung sind:

- Linien verlängern, Objekte zusammenführen / zerschneiden / klonen
- Snappen
- Unterstützung weiterer Datenquellen
- Export von Daten in andere Formate (csv, Shape, Excel)

Suchen mit dem SearchRouter, SimpleSearch (Solr) & Digitizer als Suchmodul

Eine der Entwicklungen ermöglicht nun die Suche in den Daten. Für die Suche gibt es drei Möglichkeiten:

- SearchRoutes – Suche über SQL
- SimpleSearch mit Apache Solr
- Verwendung des Digitizers als Suchmodul (Deaktivierung der Bearbeitung)

SearchRouter – Suche über SQL

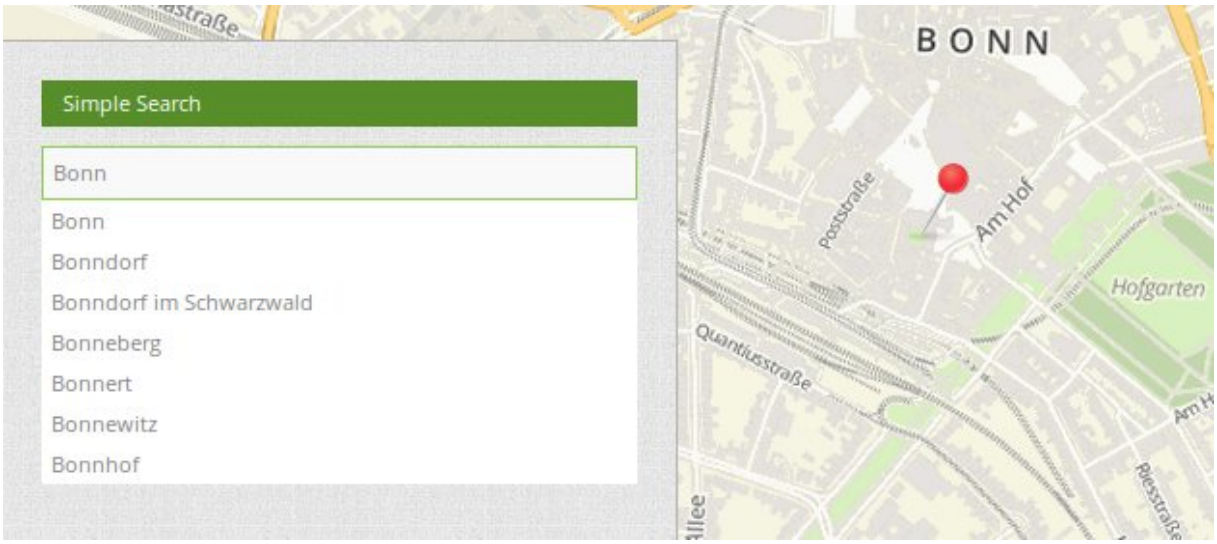
Der SearchRouter [3] erzeugt ein Suchformular mit Trefferausgabe. Das Formular und die Trefferausgabe sind konfigurierbar.

Die Suche greift hierzu auf Tabellen in einer Datenbank zu. Zur Zeit unterstützt der Search Router PostgreSQL, weitere Datenformate (Oracle, OGC WFS) sind denkbar.

SimpleSearch – Einfeldsuche mit Apache Solr

Das Element SimpleSearch [4] bietet eine Einfeldsuche. Hierbei kann nach Schlagworten gesucht werden. Die Suchanfrage wird dabei an einen Suchdienst z.B. Apache Solr geschickt. Die Antwort vom Suchdienst erfolgt beispielsweise als JSON und kann als Treffer ausgegeben werden.

Es wird ein Eingabefeld angeboten, welches direkt in die Toolbar oder in der Seitenleiste (Sidepane) eingebunden werden kann.



Digitizer als Suchmodul

Das bereits vorgestellte Digitizer-Element [2] kann ebenfalls als Suchmodul dienen. Der Digitizer stellt eine Objekttabelle bereit. Über diese kann auf die Objekte gezoomt werden und das Bearbeitsformular kann geöffnet werden. Die Objekttabelle ist sortierbar und durch den Parameter searchType kann der Inhalt der Tabellen aus dem derzeitigen Ausschnitt oder dem gesamten Kartenebereich durchsucht werden.

Kontakt zur Autorin:

Astrid Emde
WhereGroup GmbH & Co. KG
Eifelstraße 7
53119 Bonn
+49 (0)228 90 90 38 19
astrid.emde@wheregroup.com

Weiterführende Links:

[1] http://doc.mapbender3.org/de/bundles/Mapbender/CoreBundle/elements/digitizer.html

[2] http://demo.mapbender3.org/application/mapbender_digitize_demo

[3] http://doc.mapbender3.org/de/bundles/Mapbender/CoreBundle/elements/search_router.html

[4] http://doc.mapbender3.org/de/bundles/Mapbender/CoreBundle/elements/simplesearch.html

osm_address_db – Adressdaten in der OSM-Datenbank

Christopher Lorenz

Bereits auf der vergangenen FOSSGIS 2015 in Münster wurde das Projekt osm_address_db in einem Lighting-Talk vorgestellt. Entstanden ist die Idee für dieses Projekt im Jahr 2013 als mich jemand ansprach und wissen wollte, ob Adressdaten (Land, PLZ, Ort, Straße, Hausnummer, Position) aus der OSM-Datenbank für externe Projekte extrahiert werden können. Daraus entstanden ist eine Sammlung von Shell- und SQL-Skripten [0], die mithilfe des Tools osm2pgsql [1] und einer PostgreSQL/Postgis-Datenbank eine Tabellenstruktur aufbaut, die für Exporte und weitere Auswertungen, der in OSM vorhandene Adressdaten, verwendet werden kann.

Die OSM-Daten werden nach dem Karlsruher Schema [2],[3] analysiert, dabei werden auch associatedStreet-Relationen [4] für den Aufbau der Tabellen berücksichtigt. Folgende Daten und daraus resultierenden Tabellen werden hierzu aus dem osm2pgsql-Schema in ein eigenes Schema übernommen:

- osm_addresses - Adressen, die mindestens eine Hausnummer haben
- osm_admin - Administrative Grenzen
- osm_places - Nodes aller places
- osm_postcode - Polygone der Postleitzahlen
- osm_roads - Straßen und Wege, die einen Namen besitzen

Nach der Übernahme der Daten werden dann alle ggf. fehlenden Daten (Stadt, Postleitzahl, Land) aus den Grenz-, Postleitzahlen- und associatedStreet-Relationen in der Tabelle osm_addresses gefüllt, sodass eine vollständige Tabelle mit Adressen entsteht. Diese Tabelle kann dann als CSV-Datei exportiert werden. Die so exportierten Daten können mit einem Programm (z.B. Tabellenkalkulation) ausgewertet oder in andere Datenbanken zur Analyse eingelesen werden. Bei der Entwicklung der Datenstruktur wurde darauf geachtet, dass eine Aktualisierung der Datenbank mittels osc-Dateien ohne einen kompletten Neuaufbau möglich ist.

Um eine Verwendung und die Kontrolle der importierten Daten zu ermöglichen, wurde zunächst ein Prototyp auf Basis von PHP entwickelt, der die Auswahl eines Ortes über die Hierarchieebenen (admin_level [5]) ermöglicht und dann alle Informationen zu einem Ort bzw. einer Stadt liefern kann. Dazu zählen neben den zu den Straßen zuordenbaren Adressen auch die Postleitzahlen und nachgeordnete Hierarchieebenen (Orts- und Stadtteile). Der Prototyp ermöglicht, auf Basis der analysierten Daten, ebenfalls fehlerhafte Schreibweisen von Orts- und Straßennamen in den OSM-Tags der Adressen heraus zu finden. So können alle Adressen aufgelistet werden, die keinem Straßennamen der Stadt zugeordnet werden können. Während der Entwicklung konnten so kaum erkennbare Schreibfehler ausfindig gemacht und bereits manuell korrigiert werden. Als Beispiel ist hier der Name der Stadt (addr:city) Po**st**dam (richtig wäre Po**ts**dam) zu nennen, welcher in der brandenburgischen Landeshauptstadt bei rund 300 Adressen, über mehrere Stadtteile hinweg, fehlerhaft verwendet wurde.

Derzeit befindet sich eine Webanwendung, auf Basis des Java Frameworks Spring [6], in Entwicklung, die eine Umsetzung des Prototypen vorsieht. Dabei sind schon viele Ideen zusammen getragen und umgesetzt worden. So ist gegenüber des sehr textlastigen Prototypen die Anwendung um eine Karte ergänzt worden, siehe Abbildung 1. Die Karte zeigt die aktuelle gewählte Hierarchieebene und ermöglicht die Darstellung der Unterhierarchien oder, sofern eine Stadt ausgewählt wurde, auch Straßen bzw. Postleitzahlen der gewählten Region. Neben der Entwicklung der Webanwendung wurden auch die Datenstruktur und der Import weiter verbessert. Es wurden die ursprünglich erstellten Tabellen um Views für einfachere Abfragen ergänzt. Um komplexere Beziehungen zwischen den Daten performant abfragen zu können, wurden Materialized-Views [7] verwendet.

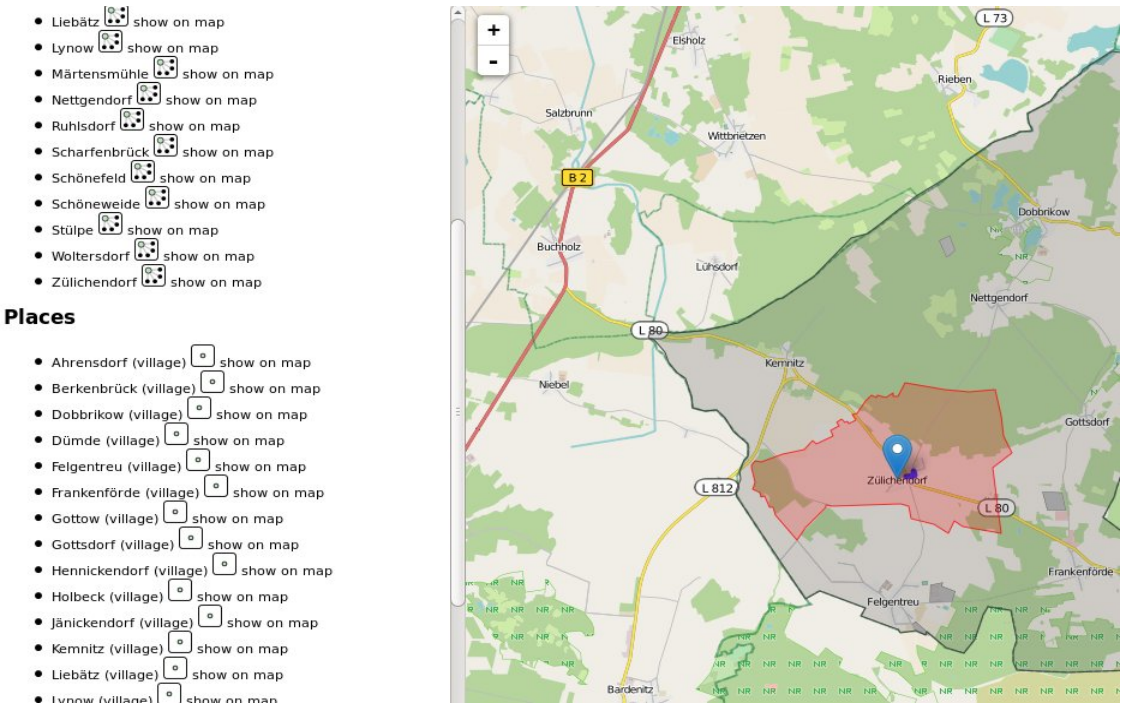


Abbildung 1: Stand der Webanwendung: Liste der Places, Auswahl eines Ortsteils und Anzeige des dazu gehörendes Places (Stand 05/2016)

Diese Anwendung soll keine Alternative, sondern eine Ergänzung der Hausnummern- und Straßenlistenauswertung von Dietmar Seifert (regio-osm.de) sein. Sie ermöglicht auch ohne einen Soll-Ist-Vergleich ein gewisses Maß an Qualitätssicherung der in OSM vorhandenen Hausnummern. Eine Verlinkung zwischen den beiden Projekte ist nach Fertigstellung der ersten Version vorgesehen.

Wer Interesse an einer Mitarbeit am Projekt oder auch Ideen hat, kann sich gern beim Autor melden.

Kontakt zum Autor:

Christopher Lorenz
osm@1011.link
OSM-User: Christopher
Twitter: @britiger

Repository Datenbankstruktur: https://github.com/britiger/osm_address_db
Seite des Prototypen: https://1011.link/osm_address_db/

Links:

[0] https://github.com/britiger/osm_address_db

[1] https://github.com/openstreetmap/osm2pgsql

[2] http://wiki.openstreetmap.org/wiki/Proposed_features/House_numbers/Karlsruhe_Schema

[3] http://wiki.openstreetmap.org/wiki/Key:addr

[4| http://wiki.openstreetmap.org/wiki/Relation:associatedStreet

[5] http://wiki.openstreetmap.org/wiki/Admin_level

[6] https://spring.io/

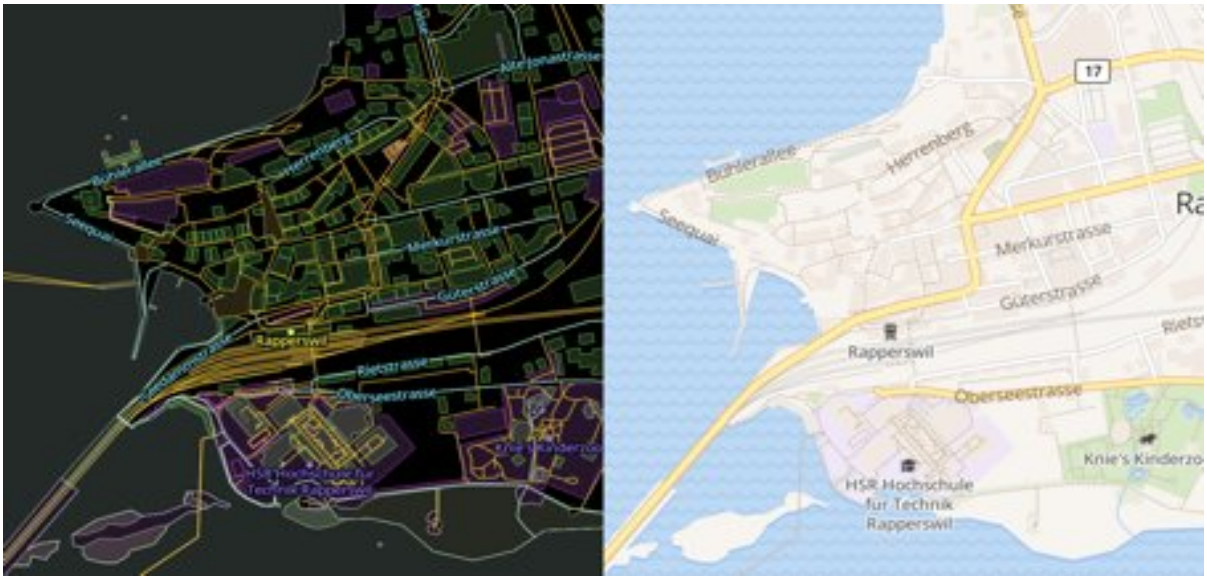
[7] http://www.postgresql.org/docs/9.5/static/sql-creatematerializedview.html

Vector Tiles from OpenStreetMap

Manuel Roth

Das Projekt OSM2VectorTiles bietet einerseits einen Workflow um selbst Vektor Tiles aus OpenStreetMap zu erstellen und bietet diese andererseits gratis zum Download an.

Der Workflow um die Vektor Tiles zu erstellen ist Open Source und kann verwendet werden um eigene Vektor Tiles zu erstellen. Dies ermöglicht jedem eine selbst gestaltete Karte der gesamten Welt zu erstellen und anzubieten, ohne Kenntnisse von PostGIS und Mapnik zu haben.



In der Präsentation zeigen wir, wie man in wenigen Minuten seine eigene Karte designen und verwenden kann. Wir führen zuerst durch ein simples Kartendesign in Mapbox Studio und zeigen wie man eine eigene Karte ohne Abhängigkeiten auf externe Anbieter zur Verfügung stellt. Danach zeigen wir wie man das Gleiche mit einer lokalen Kopie der Daten erreichen kann.

Die Zuhörer wissen nach unserer Präsentation wie sie eine selbst gestaltete Karte erstellen und veröffentlichen können. Zudem können sie die vom OSM2VectorTiles Projekt zur Verfügung gestellten Daten verwenden, um ihre Karte auch Offline anbieten zu können.

Weitere Informationen zum Projekt und wie du OSM2VectorTiles nutzen kannst, findest du auf unserer Projekt Webseite unter osm2vectortiles.org.

Kontakt zum Autor:

Vorname Name: Lukas Martinelli und Manuel Roth

Arbeitgeber: HSR Hochschule für Technik Rapperswil, Schweiz

Postadresse: Oberseestrasse 10, 8640 Rapperswil-Jona, CH

Telefon: 0041 79 256 58 00

eMail: manuel.roth@hsr.ch

Stand der Hausnummern in OSM und Hausnummerauswertung auf regio-osm.de

Dietmar Seifert

Die Erfassung von Hausnummern in OpenStreetMap hat sich in den letzten Jahren enorm gesteigert. Es wird der aktuelle Stand erläutert in Deutschland, aber auch in anderen EU-Ländern. In einigen Ländern wurden die Hausnummern aus verfügbaren Quellen importiert oder es gibt landesweite Listen zum Datenabgleich, in Deutschland ist die Situation je nach Bundesland sehr unterschiedlich und in Bewegung.

Die Auswertungsmöglichkeiten auf regio-osm.de werden vorgestellt. Neue Funktionen vereinfachen die Fehlerbeseitigung oder unterstützen Importe. Thematisch passende andere Auswertungen werden genannt.

Die klassische Erfassung der Hausnummern vor Ort, in Kombination mit der Qualitätssicherung durch Listenabgleich, deckt auch Fehler in den offiziellen Listen auf. Die Vorteile auch für die Datenbereitsteller sollen aufgezeigt werden.

Mapillary - Alltag

Lars Schimmer

Mapillary wurde schon einmal auf der FOSSGIS 2014 im Rahmen eines Lightning Vortrags vorgestellt, somit wird mein Vortrag den Alltag aus rund 2 Jahren mit Mapillary beschreiben.

Doch was ist Mapillary überhaupt?

Mapillary ist eine schwedische Firma, die ähnlich zu Google Street View Fotos von allen möglichen Plätzen der Welt sammelt und entsprechend aufbereitet über eine Webschnittstelle für die Nutzer bereit hält.

Als Besonderheit kann jeder Nutzer mit einem eigenem Account georeferenzierte Bilder zu Mapillary hochladen, die in deren Datenbank integriert werden. Ebenso stehen diese Bilder dann unter der CC BY-SA 4.0 Lizenz (frei zum Teilen und Verändern mit Angabe der Quelle, Lizenz muss erhalten bleiben) zur Verfügung. Ebenso können die Bilder aus Mapillary frei genutzt werden, um Metadaten für Open Street Map abzuleiten (so genanntes Foto-Mapping). Mapillary ist jedoch eine Firma und möchte auch Geld verdienen, somit werden die Bilder in der Datenbank mit diversen Algorithmen bearbeitet und weitere Daten daraus gewonnen, unter anderem 3D Punktwolken mit der (freien) Open Structure from Motion Methode, eine Auswertung der vorhandenen Verkehrsschilder, oder aber Erkennung der freien Parkplätze. Diese erweiterten Daten sind jedoch nicht frei und nur mit einem Businessplan von Mapillary erhältlich.

Mapillary stellt Apps für Android, iOS, Windows Phone und Amazon Geräte bereit, neben Skripten zum manuellen Hochladen der Bilder. Aber auch Bilder aus Action- oder Dash-Cams können mit GPS Daten versehen hochgeladen werden, und sogar Videos können mit einem separat aufgezeichnetem GPS Datentrack hochgeladen und zu Fotos in der Datenbank verarbeitet werden. Bilder in einer Aufnahme-Session werden von den Apps zu Tracks zusammengefasst, um die Zugehörigkeit zueinander besser verwalten zu können.

Auf dem https://www.mapillary.com Webservern werden diese Tracks dann auf einer Weltkarte (diverse Stile aus OSM werden bereit gehalten) dargestellt und mittels Klick auf einem Punkt der Karte erscheinen dann die entsprechenden Bilder, durch die man sich im Browser hindurch navigieren kann. Ebenso können die 3D Punktwolken zu der Position dargestellt werden. Falls zu einem Ort mehrere Fotos vorhanden sind, kann im Webinterface auch zwischen diesen (zeitlich unterschiedlichen) Fotos gewechselt werden. Dieses ist ein nettes Point-In-Time Feature, um z.B. die Historie eines Platzes verfolgen zu können.

Die Erfassung der Bilder zu Fuß, Fahrrad und Auto, und das entsprechende Hochladen und Bearbeiten der Daten werden Thema dieses Vortrags sein, auch die Probleme und Sorgen dabei werden nicht vergessen.

Kontakt zum Autor:

Lars Schimmer
TU Graz
Inffeldgasse 16c
+43-316-8735405
aceini@gmx.de

Leitstellensimulator goes OpenStreetMap

Ein Erfahrungsbericht

SERHAN ŞEN

1. Einleitung

1.1 LstSim und die Geodienste

LstSim ist ein nichtkommerzielles Browserspiel, bei dem man in die Rolle von Disponenten auf Rettungsleitstellen schlüpfen kann. Man nimmt Anrufe entgegen, disponiert Rettungsmittel wie zum Beispiel Rettungswagen oder Notarzteinsatzfahrzeuge und verwaltet – je nach Leitstellengebiet – nebenher noch die Abwicklung geplanter Krankentransporte.

Bei der Disposition der Einsätze werden die Spieler durch eine Kartenansicht unterstützt. Die Spiellogik ist in JavaScript implementiert und läuft im Browser ab. Aus diesem Grund wurde seit Beginn an auf die Dienste der Google Maps API gesetzt.[1] Es wird dabei nicht einfach nur eine simple Google-Maps-Karte verwendet, sondern auch die Directions- und Geocoding-Dienste der Google Maps JavaScript API.[2][3]

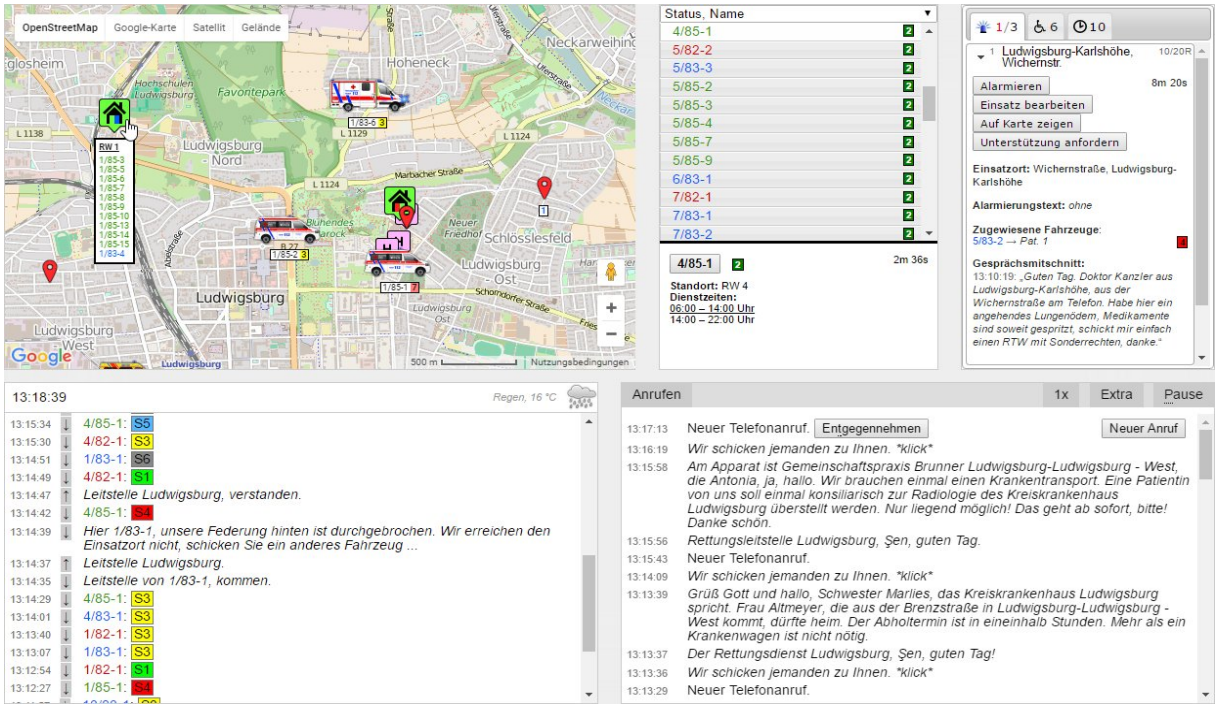


Abbildung 1: Screenshot aus einer Spielsitzung in LstSim

Diese Dienste werden beispielsweise genutzt, um Fahrzeugsymbole auf der Karte entlang der dargestellten Straßen mit realistischen Geschwindigkeiten zu animieren. Auch für die Generierung zufälliger Einsatzorte wird der Directions-Dienst herangezogen. Mit Hilfe des (Reverse-)Geocoding-Dienstes können diesen Einsatzorten genaue Straßen- und Ortsbezeichnungen zugeordnet werden. Außerdem können Fahrzeuge, die unterwegs sind, mit diesem Dienst – so wie in der Realität auch – auf Anforderung der Leitstelle ihren aktuellen Standort durchgeben.

Kurzum: Geodienste sind ein fester Bestandteil von LstSim und der Betrieb des Spiels ist ohne sie nicht möglich. Im Laufe einer mehrstündigen Spielsitzung können deshalb durchaus einige Hundert

Anfragen an die verschiedenen Geodienste entstehen. Diese Menge ist im Fall der Google Maps API noch im Rahmen der (in der Vergangenheit) angegebenen Nutzungsbeschränkungen – 2 500 Anfragen pro IP-Adresse und Tag. Da die Anfragen von den Browsern der Spieler erzeugt werden, stellte dies kein Hindernis dar.

1.2 Probleme mit der Google Maps API

Nach mehr als zwei Jahren nach der Entstehung von LstSim im Juni 2010 traten jedoch Fehler bei der Routenfindung und der Generierung neuer Einsätze in Spielsitzungen auf, die länger als eine Stunde dauerten. Bei einer näheren Untersuchung wurde klar, dass Anfragen an die Google Maps API fehlschlugen, nämlich wenn innerhalb einer geladenen Webseite mehr als 100 bis 200 Anfragen an die Google-Maps-Dienste gesendet wurden.

Dies stellte ein großes Problem dar. Als nichtkommerzielle Webanwendung verwendet LstSim die kostenlose Google-Maps-Schnittstelle und kann deshalb natürlich keinen persönlichen Kundendienst bei Google in Anspruch nehmen. Es wurde jedoch ein Fehler in der Google Maps API vermutet, da das Problem plötzlich mit einer neuen API-Hauptversion auftauchte und die Änderung im Changelog oder in den Nutzungsbedingungen nicht erwähnt wurde.

Aus diesem Grund erschien eine Meldung im Issue Tracker der Google Maps API sinnvoll. Dort wurde das Anliegen zwar bestätigt, doch es gab nie eine Erklärung, ob hier ein technisches Problem auf Benutzer- oder Betreiberseite vorliegt oder vielleicht eine Änderung der Nutzungsbeschränkungen einfach nicht kommuniziert wurde. Das entsprechende Issue hat noch heute den Status „Confirmed“ (Stand Mai 2016).[4]

1.3 OpenStreetMap als Lösung der Probleme?

In der LstSim-Community wurde deshalb früh der Vorschlag gemacht, von der Google Maps API zu OpenStreetMap zu wechseln. Hier fangen jedoch die Schwierigkeiten bereits an: Die Google Maps API ist eine komplett einsatzbereite Schnittstelle für den Produktivbetrieb. OpenStreetMap ist ein Projekt, bei dem es primär um das Zusammentragen freier Geodaten geht. Dies bedeutet, dass für die effektive Nutzung der OpenStreetMap-Daten Erfahrung im Betrieb mit Serveranwendungen im Allgemeinen, eine Einarbeitungsbereitschaft in typische OpenStreetMap-Werkzeuge im Besonderen und schließlich auch die Anpassung der LstSim-Codebasis an die neuen Dienste nötig sind.

Wie bereits erwähnt sind die Geodienste der Google Maps API tief in die Spielmechanik von LstSim integriert. Der Programmcode ist, wie das Projekt selber auch, organisch gewachsen, weshalb sich verwendete Schnittstellen nicht einfach austauschen lassen. Es werden teilweise Informationen aus den Geodiensten für bestimmte Spielfunktionen „zweckentfremdet“, für die es bei der Verwendung von OpenStreetMap-Anwendungen nicht immer ein entsprechendes Gegenstück gibt. Dazu gehört beispielsweise die Erkennung von Fährenstrecken bei der Routenfindung anhand der zurückgelieferten Wegbeschreibungstexte.

Auch der finanzielle Aspekt stellt beim Betrieb eigener OpenStreetMap-Anwendungen eine Herausforderung dar. Die Serverbetriebskosten beliefen sich auf rund 25 Euro pro Monat und wurden seither problemlos und gerne aus privater Hand bezahlt. Um jedoch Geodienste bereitstellen zu können, die den ganzen Planeten abdecken, werden leistungsstarke, dedizierte Server benötigt, die auch einen entsprechenden Kostenpunkt implizieren. Mit dem bisherigen Finanzierungsmodell schien dies nicht realisierbar zu sein.

Für die verschiedenen Geodienste kann man natürlich auch kleinere Datenauszüge verwenden (zum Beispiel nur aus Deutschland), wodurch auch die Serveranforderungen sinken würden. Jedoch würde das im Fall von LstSim signifikante Einschränkungen mit sich bringen.

Engagierte LstSim-Spieler können seit einigen Jahren durch die Leitstellenbaukastenfunktion eigene Leitstellenbereiche erstellen und für alle Benutzer spielbar machen. Dabei gab es keine regionalen Einschränkungen, wodurch es mittlerweile Leitstellen auf allen Kontinenten gibt. Auch wenn für die meisten Spieler sicherlich die Leitstellen in Deutschland und den angrenzenden Ländern interessant sind, sollen die Spieler und Ersteller der anderen Bereiche nicht einfach ausgeschlossen werden.

1.4 OpenStreetMap als Lösung der Probleme!

Interessanterweise hatten die Einschränkungen bei der Spieldauer nie einen messbaren Einfluss auf die Anzahl der LstSim-Spieler, was auf eine gleichbleibende Nachfrage deutete. Im Herbst 2015 fiel deshalb die Entscheidung, die Umstellung auf die OpenStreetMap-Dienste erneut ernsthaft zu prüfen und die vorherige Spielbarkeit wiederherzustellen. Da es jedoch keinen maßgeschneiderten OpenStreetMap-Masterplan für die Anforderungen von LstSim gibt, bedeutete dies viele Dinge erst mal ausprobieren und testen zu müssen.

Auch war damit zu rechnen, dass die Serverkosten signifikant steigen würden. Aus dem Grund sollte für die LstSim-Spieler die Möglichkeit eingerichtet werden, sich durch Spenden an den Serverkosten beteiligen zu können. Dies stellte eine neue Art der Beteiligung der Spielergemeinschaft an LstSim dar, was natürlich immer mit einer gewissen Unsicherheit verbunden ist. Wie groß würde die Bereitschaft zur finanziellen Unterstützung letzten Endes wirklich sein?

Das größte Problem war die Routenfindung, da die meisten API-Anfragen von dieser Art waren und ihr Ausfall am ehesten bemerkt wird – ohne Routen „fahren“ die Fahrzeuge im Spiel zwangsläufig Luftlinie. Wie sich später jedoch herausstellte, schlugen nach ein bis zwei Stunden Spielzeit auch die Anfragen an den Geocoding-Service fehl. Dies machte sich primär durch das Ausbleiben neuer Einsätze an Zufallsorten bemerkbar. Einzig mit der Kartenfunktionalität der Google Maps API gab es im weiteren Spielverlauf keine Schwierigkeiten.

Der Plan war es also, schrittweise auf die Geodienste rund um OpenStreetMap zu migrieren, und zwar in der Reihenfolge ihrer Wichtigkeit für LstSim. Während die Routenfindung, dicht gefolgt vom Reverse Geocoding, essenziell ist, hatten das Kartenmaterial und die Google Maps JavaScript API an sich nachrangige Priorität. Außerdem ist es wichtig, dass die Dienste stabil betrieben werden können und es einen Plan für Serverausfälle gibt. Das langfristige Ziel ist die komplette Unabhängigkeit von der Google Maps API – auch wenn sie über lange Zeiträume hinweg gute (und kostenlose) Dienste erwiesen hat.

2 Technische Umsetzung

2.1 Leitfragen

Bei den Tests mit den unterschiedlichen OpenStreetMap-Anwendungen wurde schnell klar, dass für den Betrieb der Programme diese drei Fragen stets von großer Bedeutung sind:

1. Welche Ressourcen werden für den initialen Datenimport benötigt?
2. Welche Ressourcen werden für den Betrieb benötigt?
3. Welche Ressourcen werden für die stetige Aktualisierung der Daten benötigt?

Mit Ressourcen sind dabei vor allem drei Größen gemeint: Speicherplatzmenge, Speicherplatz-Performance sowie RAM-Größe. CPU-Performance hat sich praktisch nie als Flaschenhals erwiesen, da die meisten Anbieter dedizierter Server moderne Prozessoren einsetzen.

Außerdem stand relativ zügig die Notwendigkeit für insgesamt zwei OpenStreetMap-Server fest. Die Importvorgänge können mehrere Wochen dauern, was bedeutet, dass bei einem (Hardware-)Ausfall eines einzelnen Geodienste-Servers für die gesamte Dauer des Re-Importvorgangs kein Spielbetrieb möglich ist. Dies ist vor allem im Hinblick auf die Tatsache, dass der Betrieb durch Spenden finanziert werden soll, kein akzeptables Risiko.

Durch die zwei Server entstehen gewisse Kombinationsmöglichkeiten. Auf welchem Server soll welcher Dienst im Produktivbetrieb laufen? Welche Serverkonfiguration ist dafür nötig? Wegen der großen OpenStreetMap-Datenmengen bringen die einzelnen Dienste die Server schnell an die Grenzen ihrer Leistungsfähigkeit.

Das Angebot des Serverbetreibers Hetzner ist umfangreich und weist ein grundsätzliches gutes Preis-Leistungs-Verhältnis auf. Es wurde deshalb eine Serverkombination gesucht, die nicht viel mehr als etwa 100 Euro pro Monat kosten sollte.

2.2 Routing

Bei der Routing-Software gibt es einige Kandidaten, die zumindest auf dem Papier die Anforderungen erfüllen. Dazu gehören eine zügige Berechnung der Routen sowie idealerweise eine integrierte REST-Schnittstelle. Hier fiel die Entscheidung auf OSRM, auch weil sich das Aufsetzen und der Betrieb der Software vergleichsweise einfach bewerkstelligen lässt.[5]

Die Serveranforderungen für den Datenimport der Planet-Datei sind interessant: Es werden unter anderem über 120 GB RAM benötigt. Ein Server mit so viel RAM war deutlich außerhalb des anvisierten Budgets. Es stand deshalb die Frage im Raum, ob sich die Daten auch mit weniger RAM importieren lassen.

Der erste Importversuch auf einem Server mit 32 GB RAM und einer ausreichend großen Auslagerungsdatei auf normalen Festplatten (RAID 1) funktionierte erwartungsgemäß nicht. Der Importvorgang kam auch nach mehreren Tagen über das Anfangsstadium nicht hinaus, lastete aber dafür die Festplatten voll aus.

Glücklicherweise ist der große RAM-Bedarf von OSRM hauptsächlich dadurch begründet, dass wahlfreier Zugriff auf die zu importierenden Daten benötigt wird, welcher von der Bandbreite her aber nicht zwangsweise so schnell sein muss, wie es mit physikalischem RAM möglich ist. Mit anderen Worten: Die Hoffnung war, dass der Importvorgang mit einer Auslagerungsdatei auf einer SSD funktionieren würde.

Die in Frage kommende Servervariante hat 32 GB RAM, zwei Festplatten (je 2 TB) sowie zwei SSDs (je 240 GB). Die Idee war es, die SSDs als Cache für die zwei Festplatten zu verwenden. OSRM würde dann nur 2 TB großen Speicherplatz sehen, dessen Lese- und Schreibzugriffe transparent durch die SSD beschleunigt würden. Dies ist von Vorteil, weil OSRM neben dem hohen Speicherverbrauch auch eine 200 GB große temporäre Datei für sequenzielle Zugriffe benötigt.

Tatsächlich: Der Importvorgang läuft mit Hilfe der bcache-Funktionalität im Linux-Kernel innerhalb eines Tages durch.[6] Es ist jedoch wünschenswert, die Komplexität der Serverkonfiguration möglichst gering zu halten, zum Beispiel im Fall eines Festplattendefekts. Bei weiteren Tests stellte sich heraus, dass der Importvorgang ähnlich schnell abläuft, wenn die Auslagerungsdatei auf den beiden SSDs in einer einfachen RAID-1-Konfiguration und die restlichen Dateien auf den Festplatten gespeichert werden.

Im Betrieb benötigt OSRM mit der derzeitigen Planet-Datei rund 33 GB RAM. Der Server ist so konfiguriert, dass die Daten im RAM tendenziell früh in die Auslagerungsdatei ausgelagert werden (vm.swappiness = 100). OSRM ist im Normalbetrieb die einzige speicherintensive Anwendung und nach einigen Tagen Betrieb werden rund 40 % der Daten im RAM ausgelagert – es werden schließlich nicht ständig Routen für alle Strecken auf der Erde benötigt. Auf diese Weise bleiben sogar 10 GB für den Linux Page Cache übrig.

Um den Datenbestand zu aktualisieren ist ein kompletter Importvorgang nötig. Der OSRM-Prozess wird dabei bis auf 10 GB komplett ausgelagert. Dies hat jedoch keine spürbare Auswirkung auf die Performance im Produktivbetrieb des Dienstes. Es reicht auch, diesen Vorgang alle paar Monate oder bei Anlass (zum Beispiel bei Korrekturen bei OpenStreetMap für bekannte Routingprobleme) zu starten.

2.3 Geocoding

Für das (Reverse) Geocoding bietet sich Nominatim als Standardlösung an.[7] Die Planet-Datei wird dabei mit osm2pgsql in eine PostgreSQL-Datenbank mit PostGIS-Erweiterung importiert.[8] Um den Importvorgang so gut wie möglich zu beschleunigen, kann man die Größe des Node Cache auch auf einen höheren Wert als den verfügbaren RAM einstellen, mit dem Wissen, dass gegebenenfalls Teile des Speichers auf SSD ausgelagert werden. Der Importvorgang dauert auf dem im vorherigen Abschnitt genannten Server rund zwei bis drei Wochen.

Das Aktualisieren des Datenbestands stellt eine gewisse Herausforderung dar. Man kann die Datenbank zwar mit Replication Diffs von OpenStreetMap aktuell halten, doch es ist wichtig, dass OpenStreetMap-Änderungen für einen bestimmten Zeitraum (zum Beispiel 24 Stunden) in einem höchstens gleich langen, besser kürzeren Zeitraum in die Datenbank übertragen werden können. Erfreulicherweise scheint auf dem verwendeten Server das verwendete Datenformat dafür geeignet zu sein, in rund

60 bis 70 % der Zeit, den ein Änderungsdatensatz abdeckt, importiert werden zu können. Dies bedeutet, dass nach mehreren Wochen stetiger Datenaktualisierung Änderungen bei der OpenStreetMap-Quelle praktisch live in das eigene System eingespielt werden.

Die Daten in der Datenbank belegen derzeit rund 800 GB und können deshalb ohne Weiteres nicht auf SSD-Speicher abgelegt werden. Für den Betrieb ist deshalb möglichst viel freier RAM wichtig, der von Linux und PostgreSQL für den Cache verwendet wird. Ohne einen aufgewärmten Cache können Reverse-Geocoding-Anfragen – insbesondere bei mehreren parallelen Anfragen – bis zu einer halben Minute oder länger dauern. Mit den Daten aus dem Cache sind stattdessen Antwortzeiten von unter 100 Millisekunden möglich.

2.4 Karten

Der Import für die Kartendaten läuft ähnlich wie der für das Geocoding. Ein kompletter Import läuft sogar in weniger als einer Woche durch. Bei der Aktualisierung der Daten tritt allerdings genau der Fall auf, der beim Geocoding befürchtet wurde: Der Import eines Änderungsdatensatzes für einen Tag dauert mit rund eineinhalb bis zwei Tagen deutlich zu lange. Der Abstand zum aktuellen Datenstand bei OpenStreetMap würde so immer größer werden. Es wäre jedoch akzeptabel, wenn das Kartenmaterial nicht wochen- oder gar monatsaktuell ist.

Stattdessen bietet sich hier die gleiche Strategie wie bei OSRM an: Immer wenn die Datenbasis auf den aktuellen Stand gebracht werden soll, wird nebenher ein kompletter Import gestartet. Nach Beendigung des Imports nimmt die neue Datenbank den Platz der alten ein und die alte Datenbank wird nach einer Prüfung verworfen. Da der Kartendienst auf dem zweiten Server betrieben wird, gerät der Importprozess mit dem Geocoding-Dienst aus Performance-Sicht nicht in Konflikt.

Sämtliches Kartenmaterial ist bis Zoomstufe 12 vorgerendert. Dieser Vorgang dauert rund eine Woche. Größere Zoomstufen sind von der Geschwindigkeit her dazu geeignet, erst bei Bedarf im Live-Betrieb gerendert zu werden.

2.5 JavaScript API

Die Google Maps API stellt nicht nur die bloße Karte und die Geodienste bereit, sondern bietet eine JavaScript-Schnittstelle an, um Objekte auf der Karte anzuzeigen, zu steuern und interaktiv zu gestalten. Um sich von dieser letzten Abhängigkeit zu lösen, ist es notwendig, die tiefe Integration aus dem LstSim-Code zu entfernen und durch eine freie Alternative, wie zum Beispiel Leaflet oder OpenLayers, zu ersetzen.[9][10] Diese Umstellung ist bei LstSim mit Stand Mai 2016 allerdings noch nicht abgeschlossen.

2.6 Serverinfrastruktur

Tabelle 1 enthält eine Übersicht über die Serverinfrastruktur, die bei LstSim für die OpenStreetMap-Dienste verwendet wird.

	Server A	**Server B**
RAM	32 GB	64 GB
Speicherplatz	2x2 TB HDD, 2x240 GB SSD	2x2 TB HDD
Routing	Betrieb	Stand-by
Geocoding	Betrieb	Stand-by
Karte	Stand-by	Betrieb

Tabelle 1: LstSim-Serverinfrastruktur für OpenStreetMap-Dienste

„Betrieb“ bedeutet Produktivbetrieb mit aktuell gehaltenen Daten. „Stand-by“ steht für einen theoretisch einsatzbereiten Geodienst, allerdings mit einmalig importierten und danach nicht mehr aktualisierten Daten. Mit dieser Aufteilung kann der Betrieb von LstSim sichergestellt werden, auch wenn ein Server komplett ausfallen sollte und neu eingerichtet werden muss.

Für diese Aufteilung musste auf Server A jedoch ein Kompromiss in Kauf genommen werden. Die Geocoding-Daten belegen wie bereits erwähnt rund 800 GB Speicherplatz. Ein komplett importierter Datensatz für das Kartenmaterial benötigt zusammen mit der Datenbank und vorgerenderten Kartenausschnitten etwa 700 GB.

Diese Konstellation würde Importvorgänge für OSRM praktisch unmöglich machen, da dieser Vorgang bereits über 300 GB freien Speicherplatz benötigt. Aus diesem Grund wurden für das Kartenmaterial auf Server A nur die Daten für Deutschland, Österreich und die Schweiz importiert, welche weniger als 100 GB belegen. So können bei einem Ausfall von Server B immer noch Karten für die wichtigsten Gebiete bei LstSim ausgeliefert werden.

2.7 Dateisystem ZFS

ZFS ist ein Dateisystem und Volume Manager, das ursprünglich von Sun Microsystems entwickelt und später für Linux portiert wurde.[11][12] Es hat im Vergleich zu herkömmlichen Dateisystemen wie ext4 viele zusätzliche Funktionen. Eine, die für die Serverkonfiguration von LstSim besonders interessant ist, ist die Möglichkeit der transparenten Komprimierung der Dateien.

Insbesondere die Daten der PostgreSQL-Datenbank lassen sich sehr gut komprimieren, da diese mehr auf Geschwindigkeit als auf Speicherplatzeffizienz ausgelegt sind. Auf Server B belegen die importierten Daten für das Geocoding und das Kartenmaterial – theoretisch – insgesamt 1,3 TB. Da die Daten jedoch auf einem ZFS-Dateisystem mit transparenter Komprimierung liegen, belegen sie nur 472 GB – ein Komprimierungsfaktor von über 2,7. Durch die Ersparnis beim Speicherplatz ist beispielsweise ein vorerst ausreichend großer Puffer für das Wachstum der OpenStreetMap-Datenbasis geschaffen.

Ein interessanter Nebeneffekt ist die Tatsache, dass Daten, die sich gut komprimieren lassen, mit ZFS zusätzlich schneller geschrieben und gelesen werden. Dies ist nachvollziehbar: Wenn weniger Daten geschrieben und gelesen werden müssen, geht dies insgesamt auch schneller, vor allem da der Standard-Komprimierungsalgorithmus bei ZFS auf Geschwindigkeit optimiert wurde.

Es gibt jedoch auch Nachteile. Anwendungen, die mehr RAM benötigen, als vorhanden ist und dann entsprechend auf die Auslagerungsdatei zurückgreifen würden, lassen sich nicht stabil betreiben. So war der Import von Daten für OSRM auf Server A auch mit unterschiedlichen System- und Anwendungskonfigurationen stets nicht erfolgreich, weshalb auf diesem Server der Einsatz von ZFS derzeit nicht möglich ist. ZFS on Linux wird jedoch aktiv weiterentwickelt und soll beispielsweise mit der nächsten Hauptversion von Debian Linux mit Bordmitteln installiert werden können. Weitere Versuche mit neuen Versionen von ZFS on Linux können deshalb durchaus lohnenswert sein.

3 OpenStreetMap-Herausforderungen

3.1 Grundsätzliche Einordnung

Bei einem Projekt mit dem Umfang von OpenStreetMap versteht es sich von selbst, dass die zusammengetragenen Daten nicht immer automatisch alle Anforderungen eines Browserspiels voll erfüllen. Durch die vielen LstSim-Spieler, die in verteilten Regionen spielen, werden solche Fälle schnell und zuverlässig entdeckt.

Die genaue Einordnung dieser Fälle ist bereits eine eigene Herausforderung: Zum einen kann ein Problem mit den OpenStreetMap-Daten vorliegen, zum anderen kann es sich natürlich auch um einen Fehler in der Datenverarbeitung auf der Seite von LstSim handeln. Seit den ersten Umstellungen auf OpenStreetMap bei LstSim traten Probleme hauptsächlich bei zwei Funktionen auf: Bei berechneten Routen sowie bei der Bezeichnung von Ortsteilen.

3.2 Routing-Vorschläge

Routingprobleme treten vergleichsweise selten auf. Zu ihnen gehören falsch eingestellte Barrieren, Abbiegevorschriften oder Straßenverknüpfungen an Kreuzungen. Die Ursache für diese Art von Problemen können im Normalfall schnell identifiziert werden. Zu Hilfe kommt dabei der Demo-Server von OSRM, der stets aktuelle OpenStreetMap-Daten verwendet.[13] Spieler können bestimmte Strecken verlinken, auf denen man das Problem nachvollziehen kann und beispielsweise auch mit Strecken vergleichen, wie sie von Google Maps vorgeschlagen werden. Mit diesen Informationen können dann die betroffenen Stellen in einem OpenStreetMap-Editor näher untersucht werden.

Jedoch kann nicht bei allen Routingprobleme die Ursache so einfach gefunden werden. Es gibt beispielsweise Strecken, bei denen in der Realität Autofahrer grundsätzlich eine bestimmte Schnellstraße nehmen würden. OSRM schlägt jedoch manchmal unterschiedliche angrenzende, kleinere Straßen vor, „springt“ aber hin- und her, wenn man Start- und Zielpunkt geringfügig variiert. Hier gestaltet sich die Ursachenforschung besonders schwierig: Liegt ein Datenproblem vor (zum Beispiel ein falsch eingestelltes Tempolimit)? Liegt ein Konfigurationsproblem bei OSRM vor? Oder vielleicht ein Problem mit OSRM selber?

3.3 Ortsteilbezeichnungen

Die bei LstSim am häufigsten gemeldeten OpenStreetMap-Probleme beziehen sich jedoch auf die Bezeichnung von Ortsteilen, die durch Nominatim per Reverse Geocoding gebildet werden. Anrufer im Spiel melden sich nicht einfach mit dem Namen der Stadt, in der sie sich befinden, sondern geben auch Ortsteile an (sofern vorhanden). Dies ist ein wichtiger Beitrag für das Realismusempfinden im Spiel und sorgt für eine zusätzliche Übersicht bei der Verwaltung von Einsätzen.

Während Gemeindegrenzen in Deutschland erfasst sind, enthalten Ortsteile in der Regel nur einen Node in der Mitte, der den Namen des Ortsteils angibt. In vielen Fällen reicht das auch tatsächlich aus, damit Nominatim daraus richtige Ortsteilbezeichnungen bilden kann. Es gibt jedoch auch Stellen, etwa in grenznahen Gebieten, bei denen der Ortsteil-Node eines Nachbarorts näher ist als der des eigentlichen gesuchten Orts. Nominatim erzeugt dann einen Ortsnamen mit Hilfe des nächsten (falschen) Ortsteil-Node. Das Problem kann auch auftauchen, wenn es für den gefragten Ort gar keinen Ortsteil gibt, aber ein anderer Ortsteil zufälligerweise in der Nähe ist.

3.4 Lösungsansätze

OpenStreetMap ist ein Projekt, das durch die Arbeit vieler einzelner Mitwirkenden getragen wird. Idealerweise würden deshalb Benutzer, denen ein Datenproblem bei OpenStreetMap auffällt, dieses Problem „einfach“ selber lösen. Bei der Umsetzung in der Realität gibt es jedoch eine Reihe von Hürden, die diesen Ansatz nicht als pauschale Herangehensweise ermöglichen.

Die unterschiedlichen Probleme erfordern unterschiedliche Arten von Änderungen an der OpenStreetMap-Datenbasis. Während die Bearbeitung von Nodes nach einer überschaubaren Einarbeitungsphase vergleichsweise leicht zu bewerkstelligen ist, gibt es bei der Bearbeitung von Grenzen, also Relationen, mehrere Aspekte, die berücksichtigt werden müssen. Bei einer unachtsamen Bearbeitung können schnell neue Probleme verursacht werden, die auch nicht unmittelbar ersichtlich sind. Diese Arten von Änderungen werden deshalb häufig nur durch erfahrene OpenStreetMap-Benutzer vorgenommen.

Da immer wieder Probleme in den OpenStreetMap-Daten gemeldet werden, lohnt es sich, dafür einen Standardablauf zu entwickeln. Die Benutzung der Notes-Funktion bei OpenStreetMap stellt dabei einen möglichen Weg dar. Es gibt jedoch in der LstSim-Spielergemeinschaft Benutzer, die bereit sind, sich auch in neue, technisch komplexe Sachverhalte einzuarbeiten und sich selber an die Problemlösungen zu wagen.

Dieses Potenzial kann genutzt werden, indem beispielsweise spezialisierte Anleitungen für die Lösung bestimmter Problemarten zur Verfügung gestellt werden. In solchen Anleitungen kann dann etwa auch auf typische Stolpersteine und andere zu beachtende Aspekte hingewiesen werden. Auch wenn es am Anfang vermutlich noch zu Schwierigkeiten kommen kann, könnte man das Verfahren im Laufe der Zeit schrittweise verbessern.

4 Fazit

Die Umstellung auf die Geodienste auf OpenStreetMap-Basis bei LstSim ist fast abgeschlossen und hat sich bisher als sehr interessantes und lehrreiches Projekt erwiesen. So wurden etwa die Vor- und Nachteile des Betriebs eigener Geodienste deutlich. Auf der technischen Seite konnten viele Erfahrungen im Betrieb anspruchsvoller Anwendungen auf einer eingeschränkten Serverumgebung gewonnen werden.

Die Erfahrungen mit OpenStreetMap an sich sind bei LstSim noch jung und es muss sich erst zeigen, welche Herangehensweise bei der Zusammenarbeit sich für beide Seiten als geeignet erweist. Dennoch wäre es sehr zu begrüßen, wenn beide Projekte letztlich voneinander profitieren können.

Kontakt zum Autor

Serhan Şen
LstSim.de
Twitter: @srhnsn
E-Mail: serhan@lstsim.de

Verweise

[1] Google Maps JavaScript API: https://developers.google.com/maps/documentation/javascript/

[2] Google Maps Directions Service: https://developers.google.com/maps/documentation/javascript/directions

[3] Google Maps Geocoding Service: https://developers.google.com/maps/documentation/javascript/geocoding

[4] Google Maps API Issue 4805: https://code.google.com/p/gmaps-api-issues/issues/detail?id=4805

[5] OSRM: http://project-osrm.org/

[6] bcache: https://bcache.evilpiepirate.org/

[7] Nominatim: https://github.com/twain47/Nominatim

[8] osm2pgsql: https://github.com/openstreetmap/osm2pgsql

[9] Leaflet: http://leafletjs.com/

[10] OpenLayers: http://openlayers.org/

[11] ZFS bei Wikipedia: https://en.wikipedia.org/wiki/ZFS

[12] ZFS on Linux: http://zfsonlinux.org/

[13] OSRM-Demo-Server: http://map.project-osrm.org/

Turf.js – Geoverarbeitung im Browser

Numa Gremling

Einführung

Die Open Source JavaScript-Bibliothek Turf.js ermöglicht geographische Analysen und Abfragen. Anders als bei klassischen Python-Geoverarbeitungsbibliotheken, wie PyQGIS oder auch ArcPy, ist bei Turf.js keine GIS-Anbindung erforderlich. Die Analysen können in einem beliebigen Browser ausgeführt werden. Diese Flexibilität ist besonders für Webmapping-Anwendungen von hoher Bedeutung, da die Kombination mit beliebigen Mapping-Bibliotheken wie z.B. OpenLayers oder Leaflet und damit auch eine Visualisierung der Ergebnisse möglich ist.

Klassische GIS-Programmierung

Es gibt eine Vielzahl von Bibliotheken, um Geodaten programmatisch zu bearbeiten oder zu analysieren. Hierzu gehören zum Beispiel Klassiker wie GDAL und OGR, die ganz einfach in einer Shell benutzt werden können. Alle wichtigen GIS-Lösungen bieten mittlerweile Schnittstellen an, um programmatisch auf die Werkzeuge und Prozesse zugreifen zu können. Die bekanntesten Beispiele dafür sind ArcPy und PyQGIS, Bibliotheken, die es ermöglichen auf ArcGIS bzw. QGIS zuzugreifen. Vor allem Python hat sich zu einer der wichtigsten Sprache in der GIS-Programmierung entwickelt und etliche Module wurden bereitgestellt, um von einfacher Bearbeitung der Daten bis hin zu komplexen Analysen der Daten alles Mögliche zu vereinfachen und zu automatisieren.

Web-GIS und Programmierung

Seit der Veröffentlichung von Google Maps in 2005 haben wir uns daran gewöhnt, dass man zu jedem Zeitpunkt auf eine interaktive Webkarte zugreifen kann. Die Suche nach einem Restaurant, die Berechnung der Route zu dem nächstgelegenen Kino oder die Planung eines mehrwöchigen Urlaubs, wir benutzen Webkarten mittlerweile im Alltag. Karten können inzwischen nicht nur konsumiert werden sondern auch leicht selbst erstellt werden. Neben vorgefertigten Lösungen wie CartoDB, Mango Map oder ArcGIS Online, die zwar die schnelle Erstellung einer Webkarte ermöglichen aber nicht bis ins kleinste Detail angepasst werden können, gibt es auch Schnittstellen, die die komplette Programmierung von Webkarten ermöglichen. Das Benutzen dieser Bibliotheken erfordert das Verständnis von Grundkonzepten der Webentwicklung, sowie den Umgang mit HTML, CSS und JavaScript. Sowohl im Open Source Bereich (OpenLayers, Leaflet, Mapbox) als auch im proprietären Bereich (Google Maps API, Bing Maps API, HERE Maps API, ArcGIS API for JavaScript) gibt es eine ganze Menge von Webmapping-Bibliotheken, die die Erstellung von einfachen und komplexen Karten und die Darstellung von Geodaten ermöglichen.

Web-GIS und Analysen

Die genannten Webmapping-Bibliotheken konzentrieren sich allerdings fast ausschließlich auf die Visualisierung der Daten und ignorieren das, was dem Desktop-GIS die wirkliche Kraft verleiht: die Analyse von Geodaten. Einfache Funktionen wie Puffer, Intersect oder lagebezogene Abfragen, die in jedem Desktop-GIS zur Standardausrüstung gehören, sind im Web so direkt nicht verfügbar. Das heißt nicht, dass mit diesen Bibliotheken keine Analyse-Anwendungen entstehen können, aber es gibt einen großen Nachteil: die Analysen werden nicht in der Anwendung selbst, also im Browser, sondern serverseitig ausgeführt. Mit anderen Worten: es werden Parameter an den Server geschickt, dort werden die Parameter an ein Skript oder an eine GIS Software übergeben, die Analyse wird durchgeführt und

anschließend wird das Ergebnis vom Server zurück an die Webanwendung geschickt. Das kann vor allem bei schlechter Internetanbindung eine ganze Weile dauern, da die Datenpakete im XML-Format verschickt werden und häufig einen großen Speicherplatzbedarf haben. Zudem erfordert die Nutzung eines Servers auch eine komplexe und eventuell teure serverseitige Infrastruktur.

Turf.js: Web-GIS und Analysen im Browser

Turf.js, das von Morgan Herlocker entwickelt wurde und mittlerweile ein Open Source Projekt von Mapbox ist, löst die Probleme, die eine Server-Infrastruktur mit sich bringt. Turf.js wird clientseitig, also direkt im Browser, ausgeführt. Es wird also nie eine Anfrage an den Server geschickt, um dort weiterverarbeitet zu werden, sondern die Analysen werden direkt im Browser ausgeführt. Turf.js benutzt als Eingabe-Datenformat GeoJSON und ist somit sehr flexibel, da das Datenformat quasi als Standard im Web-GIS Bereich angesehen wird. Interessant ist auch, dass Turf.js vollkommen unabhängig von einer Webmapping-Lösung funktionieren kann: es wird also gar keine Karte benötigt, um die Daten analysieren zu können. Die Visualisierung ist optional. Weil die visuelle Komponente der Geodaten bei den meisten Anwendungen aber eine wichtige Rolle spielt, wird Turf.js fast immer im Zusammenhang mit einer Webmapping-Bibliothek benutzt. Solange in dieser GeoJSON unterstützt wird, können die Outputs der Werkzeuge problemlos visualisiert werden, sprich Turf.js kann jede moderne Web-GIS Bibliothek erweitern (OpenLayers, Leaflet, Google Maps, usw.). Turf.js bietet mehr als 70 Funktionen, von Klassikern wie Puffererzeugung und Verschneidungen bis hin zu Klassifizierung, Filtern, Berechnung von Zentroiden und Abfragen. Turf.js ist umfangreich, dennoch schlank, sehr schnell und flexibel anpassbar.

Links

http://geosysnet.de/

http://turfjs.org/

https://www.mapbox.com/blog/coffee-with-turf/

https://www.mapbox.com/blog/turf-gis-for-web-maps/

https://www.mapbox.com/help/intro-to-turf/

Kontakt zum Autor:

Numa Gremling

geoSYS

Pflügerstr. 56

12047 Berlin

+49-30-82070657

numa.gremling@geosysnet.de

OpenStreet mal ohne Map

Frederik Ramm

In den Augen der meisten Nutzer ist OpenStreetMap vor allem für Karten gut - Tiles, WMS, Garmin-karten, Kartendownload aufs mobile Gerät, das sind die großen Themen. Aber abseits von diesem Mainstream gibt es viele andere Anwendungsbereiche, in die sich OSM langsam vortastet - zum Beispiel das Geocoding in seinen verschiedenen Ausprägungen oder die Netzwerkanalyse mit Routing und verwandten Problemen.

Dieser Vortrag beleuchtet eine Reihe von etwas ungewöhnlicheren Use Cases für OpenStreetMap-Daten aus der Praxis und stellt jeweils Lösungsansätze mit geeigneten Open Source-Tools vor. Aber auch die Grenzen von OpenStreetMap werden aufgezeigt - nicht alles, was potentielle Nutzer sich von OpenStreetMap erhoffen, geben die Daten wirklich her. Für einige Problemstellungen ist OpenStreetMap noch nicht geeignet, für andere wird es nie geeignet sein.
Dieser Vortrag erzählt von verschiedenen - realistischen wie auch unrealistischen - Anforderungen, die potentielle Nutzer abseits von der Kartenerstellung an OpenStreetMap stellen, und skizziert Lösungen und Probleme.

Bezahlte und organisierte Edits - Vorteile und Gefahren für OSM

Joachim Kast

OpenStreetMap wird aufgrund seiner Aktualität, Detailtreue und verfügbaren Rohdaten zunehmend im kommerziellen Umfeld wie ÖPNV, Verkehrsplanung, Rettungsleitstellen, Touristik und Logistik verwendet. Nun entsteht seitens dieser Zielgruppen zunehmend der Wunsch, den OSM-Datenbestand entsprechend der eigenen Bedürfnisse zu ergänzen und zu verändern. Obwohl Kontaktmöglichkeiten über Impressum, Foren und Mailinglisten vorhanden sind, das gesamte OSM-Wissen im Wiki verteilt ist und zahlreiche Firmen mit fundierten OSM-Kenntnissen ihre Dienstleistungen anbieten, beginnen viele Firmen oder Interessengruppen ihren aktiven OSM-Einstieg als blutige Anfänger mit oftmals falschen Vorstellungen von der Funktionsweise. Oft fallen ihre Edits nur durch zufällig entdeckte Fehler oder Lizenzverletzungen auf bzw. sie melden sich aufgrund technischer Probleme erst in einem sehr späten Projektstadium auf den Kommunikationskanälen und bitten um Unterstützung.

Seit einem Jahr sind auch zunehmende Aktivitäten von Firmen aus dem Geschäftsbereich "Suchmaschinenoptimierung" (SEO) festzustellen. Sie haben die sozialen Medien und Netzwerke entdeckt, in denen sie ihre Kunden optimal präsentieren möchten. Dies führte in den bisherigen Fällen zu massiven Konflikten.

Anhand aktueller Fälle soll die Problematik dokumentiert, aber auch positive Beispiele und Möglichkeiten aufgezeigt werden, wie sich spezielle Interessen communityverträglich integrieren lassen.

Lokalisierung von Online-Karten aus Openstreetmap-Daten

Sven Geggus

Ausgangssituation

Online-Karten auf Basis von Openstreetmap (OSM) unterscheiden sich von anderen Karten auf Basis herkömmlicher Geodaten wie ATKIS in zwei wichtigen Aspekten. Zum einen ändern sich die zugrunde liegenden Vektordaten häufig, was bei deren Einsatz z.B. durch eine kontinuierlich aktualisierte Datenbank berücksichtigt werden muss. Zum anderen liegen diese Daten üblicherweise im mehreren Sprachen bezüglich des Namens von Objekten (Länder, Städte, Straßen usw.) vor.

Der vorliegende Beitrag zeigt Lösungsansätze zur Nutzung mehrsprachiger Daten für die Erzeugung von Online-Karten. Diese Beschränkung auf Online-Karten ist jedoch nicht technischer Natur. Die beschriebenen Lösungsansätze sind natürlich auch für die Erzeugung von Karten anwendbar, die für den Druck bestimmt sind.

Problemstellung

Im Gegensatz zu herkömmlichen Daten stellt Openstreetmap einen weltweiten Datensatz in einem einheitlichen Koordinatensystem (WGS84) zur Verfügung.

Dieser erfreut sich deshalb und insbesondere auch wegen seiner freien Verfügbarkeit zunehmender Beliebtheit, und zwar nicht nur bei Interessengruppen wie z.B. Radfahrern, Wanderern, Skifahrern usw., sondern auch bei international tätigen Nichtregierungsorganisationen, Behörden und sogar Militärs.

Openstreetmap verwendet für die Bezeichnung geografischer Objekte ein flexibles Schlüssel-Wert-System, das man aus modernen Programmiersprachen kennt (associative array).

Beispiel: Teil der Datenobjekte für Deutschland und Israel:

name	⇒	Deutschland	name	⇒	ישראל
...			...		
name:de	⇒	Deutschland	name:de	⇒	Israel
name:en	⇒	Germany	name:en	⇒	Israel
name:ar	⇒	ألمانيا	name:ar	⇒	إسرائيل
name:ja	⇒	ドイツ	name:ja	⇒	イスラエル
name:ru	⇒	Германия	name:ru	⇒	Израиль
name:is	⇒	גרמניה	name:is	⇒	ישראל

Wie man sieht, wird der gewöhnliche Name eines Objektes üblicherweise in der Landessprache verfasst (Deutschland, ישראל usw.). Im Gegensatz dazu wünscht sich ein Kartenanwender jedoch eine Karte in seiner Sprache (oder einer Lingua franca wie Englisch) , bei der die Landessprache allenfalls als Bezeichnung in Klammern erscheint, wie z.B.: Moskau (Москва). Nicht-lateinische Schriftzeichen sollten dabei immer alternativ dargestellt werden.

Lösungsansatz und Stand der Technik

Zur Kartenerzeugung importiert man Geodaten aus Openstreetmap üblicherweise in eine PostgreSQL/PostGIS[2] Datenbank. Frei verfügbare Importwerkzeuge wie osm2pgsql[2] oder imposm[3] halten Mechanismen bereit, um diese Datenbank kontinuierlich auf dem aktuellen Stand zu halten.

Die eigentlichen Karten werden dann aus dieser Quelle erzeugt. Mögliche freie Werkzeuge hierfür sind Mapnik, Mapserver oder Geoserver. Des Weiteren stehen auch kommerzielle Werkzeuge wie z.B. ESRI/ArcGIS zur Verfügung. Eine neue Entwicklung stellt die Darstellung von Kartenkacheln als Vektordaten dar. Hierbei werden in einem Zwischenschritt Daten aus der PostGIS-Datenbank kachelweise extrahiert und ggfs. geometrisch vereinfacht. Diese Kacheln werden dann bei Abruf durch den Server in Rasterkacheln umgewandelt oder sogar direkt im Client, z.B. in modernen Webbrowsern mit WebGL, gerendert.

Die Lokalisierung der auf der Karte dargestellten Beschriftungen ist zur Verwendung bei allen diesen Programmen und Techniken sinnvoll. Aus diesem Grund ist es naheliegend, solche Verfahren direkt in PostgreSQL als Datenbankfunktion (stored procedure) zu implementieren. Diesen Weg geht die Implementierung, die initial vom Autor dieses Beitrags für den Kartenstil der deutschen Openstreetmap-Webseite programmiert wurde. Die aktuelle Version[4] enthält zudem einige Verbesserungen von Mitgliedern der OSM Community.

Beispiel (SQL Abfrage mit der „stored procedure“ *get_localized_placename*):

```
SELECT get_localized_placename(tags->'name', tags->'name:de',
    tags->'int_name', tags→'name:en', false, way) AS NAME
FROM planet_osm_point
WHERE osm_id=424315709;

      name
----------------------
 Bulgarien (България)
(1 Zeile)
```

Probleme und Verbesserungsmöglichkeiten der aktuellen Implementierung

Die aktuell im deutschen Kartenstil verwendete Lokalisierung kann aus vielen Gründen niemals ganz perfekt sein. Das Projekt ist prinzipbedingt mit vielen technischen aber auch politischen Problemen behaftet. Diese werden im Folgenden beschrieben.

Automatische Erzeugung lateinischer Bezeichnungen

Viele unbedeutende Objekte in der Openstreetmap-Datenbank haben gar keine internationale Bezeichnung. Hier stellt sich insbesondere bei nicht-lateinischen Alphabeten die Frage nach der Vorgehensweise. In vielen Fällen ist eine lateinische Transliteration sinnvoll. Im Falle von Straßen wird dieses Verfahren derzeit angewendet.

Beispiel:

Eine Straße in Moskau heißt „Новоостаповская улица“. Nach Transliteration ergibt sich „Novoostapovskaâ ulica“, was in der Karte dargestellt wird.

Transliterationen sind jedoch mit einer Reihe von Problemen behaftet. Einige davon sind technisch lösbar.

Nachfolgend werden bekannte Probleme dieser Art beschrieben. Es darf als sicher gelten, dass weitere existieren, die dem Autor bisher unbekannt sind.

- In manchen Sprachen (z.B. arabisch und hebräisch) werden keine Vokale geschrieben.
  Beispiel: Die Transliteration von تهران (Teheran) liefert „thrn“
  Ein prinzipielles, kaum technisch lösbares Problem.
- Derzeit kommt zur Transliteration die ICU-Bibliothek (International Components for Unicode) von IBM zum Einsatz. Diese Bibliothek verwendet eine in der thailändischen Sprache unübliche Transliteration nach ISO11940, die nicht derjenigen lokaler Straßenschilder entspricht
  Beispiel: □□□□□□□□□, ICU: thnn k̂h̄āws̄ār, üblich: Thanon Khao San
  Dieses Problem ist prinzipiell durch die Verwendung einer anderen Bibliothek lösbar, sofern eine solche verfügbar ist. Man kann diese dann nur für die Zeichen der Thai Sprache verwenden.
- Die Transliteration chinesischer Zeichen ist nur in China selbst sinnvoll anwendbar. Die selben Zeichen werden in Japan unter der Bezeichnung Kanji verwendet und völlig anders ausgesprochen.
  Beispiel: Tokio, jap. 東京 Transliteration (chin): dōng jīng
  Dieses Problem ist prinzipiell dadurch lösbar, dass man den Ort der geografischen Objekte in die Transliteration einfließen lässt. Dieser Lösungsansatz wurde vom Autor implementiert und in die Software[1] eingebaut.

Darstellung verschiedener Schriftzeichen innerhalb von Beschriftungen

Es sind nur wenige Schriftfonts mit Schriftzeichen aller Sprachen der Unicode-Tabelle in gleichbleibend guter Qualität verfügbar . Dies ist insbesondere bei Karten ein Problem, bei denen der lokale Name in nicht-lateinischer Schrift in Klammern erscheinen soll, denn die verfügbaren Renderer können die Schriftart oft nicht innerhalb eines Bezeichners ändern.

Die Implementierung des deutschen Kartenstils nutzt daher derzeit nur bei lateinischen, griechischen und kyrillischen Zeichensätzen den Kompromiss, den lokalen Namen in Klammern zu schreiben.

Dieses Problem kann technisch gelöst werden, und zwar entweder durch Weiterentwicklung der Renderer oder durch Verwendung eines Schriftfonts mit besserer Qualität.

Eventuell wäre es auch möglich, einen speziellen Kartenschriftfont zu erzeugen, der aus mehreren frei verfügbaren Einzelfonts zusammengesetzt wird.

Politische Probleme bei der Internationalisierung von Karten

Viele Regionen dieser Welt gehörten in der Vergangenheit zum Einflussbereich anderer Staaten als heute. Bekannte Beispiele sind die ehemaligen Kolonien europäischer Staaten oder deutsche Siedlungsgebiete in den heutigen Ländern Polen und Frankreich.

Wie bei vielen politischen Problemen ist eine technische Lösung in solchen Fällen schwierig. Aufgrund der Geschichte haben beispielsweise selbst kleinste Dörfer im heutigen Polen, im Elsass und in Lothringen einen deutschen Namen. Ob diese heute noch gebräuchlich sind oder nicht, lässt sich oft auch nicht mit Sicherheit sagen.

Ein Beispiel eines heute nicht mehr üblichen Namens ist die Bezeichnung „Nanzig“ für die Stadt Nancy in Lothringen. Die Einstufung solcher Namen in der OSM-Datenbank wahlweise als „name“ oder

„old_name“ sollte den lokal aktiven Mappern überlassen bleiben, die es hoffentlich am besten wissen. Hier wurde name:de als „Nancy“ erfasst, weil der deutsche Name „Nanzig“ heute nicht mehr gebräuchlich ist.

Aus diesen Gründen wird bei Ortsnamen stets der aktuelle lokale Name in Klammern verwendet, wie z.B. *Stettin (Szczecin)*. Bei Straßennamen wird derzeit der deutsche Name abgekürzt in Klammern geschrieben, weil Straßennamen vor Ort selten mehrsprachig beschildert sind. So wird bei der Karlsgasse in Prag z.B. *Karlova (Karlsg.)* verwendet.

Über den Autor:

Sven Geggus betreibt am Fraunhofer IOSB mehrere Geodaten- und Kartenserver, die in Forschungsprojekten zum Einsatz kommen. In seiner Freizeit arbeitet er beim Projekt Openstreetmap mit und hat die Lokalisierung des deutschen Kartenstils programmiert.

Kontakt zum Autor:

Sven Geggus
Fraunhofer IOSB
Fraunhoferstr. 1
76131 Karlsruhe
0721/6091-422
sven.geggus@iosb.fraunhofer.de

---

[1] Das freie Datenbanksystem PostgreSQL und dessen Erweiterung PostGIS ist unter http://postgis.net verfügbar aber auch Teil gängiger Linux-Distributionen
[2] http://wiki.openstreetmap.org/wiki/Osm2pgsql
[3] http://imposm.org/
[4] https://github.com/giggls/mapnik-german-l10n

Das ist ja wohl die Höhe!

Lars Roskoden

Die Abdeckung der Welt mit OSM-Daten hat ein hohes Maß an Quantität und Qualität erreicht. Die Höhendaten von Objekten wie z.B. Schornsteinen, Kirchtürmen, Windrädern, Brückendurchfahrten und natürlich Gebäuden sind allerdings noch ausbaufähig. Bei Gebäuden kann man sich mit der Stockwerksanzahl behelfen - bei den anderen Objekten geht das nicht. Als Mapper vor Ort kann man die Höhe der Objekte nur schlecht schätzen und verzichtet deshalb beim Mappen häufig auf diese Information.
In dem Vortrag werden einfache Möglichkeiten erläutert, wie mit mehr oder weniger Aufwand die Höhendaten von Objekten erfaßt werden können. Zu jeder der erläuterten Möglichkeiten wird auf die Vor- und Nachteile hinsichtlich Zeitaufwand, Genauigkeit, Kosten und Praktikabilität eingegangen.

Darstellung einfacher Möglichkeiten für die Höhenmessung von OSM-Objekten. Vergleich der Vor- und Nachteile.

GraphHopper - Ein Schneller und Flexibler Routenplaner

Peter Karich

GraphHopper ist ein Open Source Routenplaner basierend auf OpenStreetMap Daten. In Abb. 1 ist eine Demo für die verschiedenen Fahrzeugtypen zu sehen. GraphHopper ist in Java geschrieben und läuft auf allen gängigen Betriebsystemen inkl. Android und iOS. Das schnelle und flexible Routing hat viele kleine, mittelständische und große Unternehmen überzeugt und kommt dort als selbst gehosteter Routenplaner oder über die GraphHopper Directions API zum Einsatz.

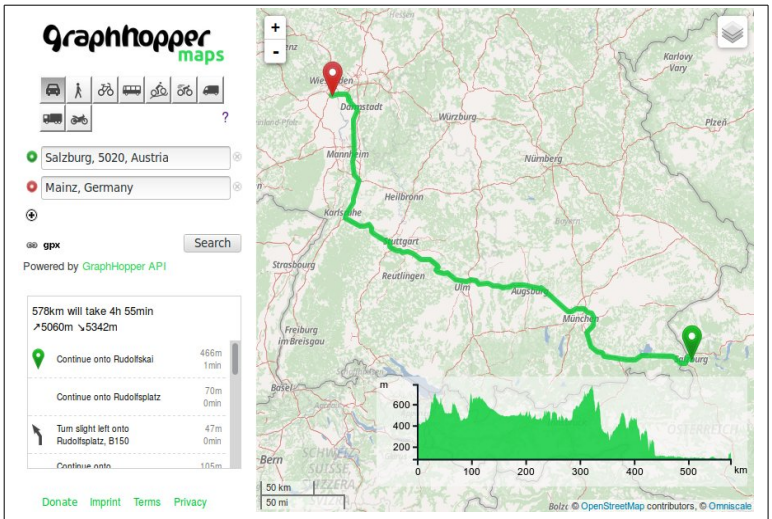


Abb. 1: GraphHopper Maps

Die GraphHopper Routingengine kann verschiedene Graphen-Algorithmen wie Dijkstra oder A* anwenden. Besonders schnell wird GraphHopper durch den Algorithmus „Contraction Hierachies“ und andere Vorverarbeitungsschritte. Alle Algorithmen sind optimal, zusätzlich können Heuristiken leicht integriert werden um so die Ausführungsgeschwindigkeit weiter zu steigern. Der Graph kann im RAM oder auf der Festplatte gehalten werden – generell werden verschiedene Techniken angewandt, damit möglichst wenig RAM verwendet wird.

Es gibt noch weitere Tools wie die sog. „Map Matching“ Komponente die es ermöglicht GPS Daten dem Straßennetzwerk zuzuordnen um den GPS Daten nachträglich z.B. Geschwindigkeitsbeschränkungen oder Abbiegehinweise zuzuordnen oder um reale Daten wie Verkehrsflussinformationen dem Straßennetzwerk zu übermitteln, also z.B. Staudaten ins Routing integrieren.

Kontakt zum Autor:

Peter Karich
GraphHopper GmbH
peter.karich@graphhopper.com

XPlanung für einen Flächennutzungsplan mit PostGIS und QGIS

Bernhard Ströbl

Einleitung

Im Flächennutzungsplan (FNP) wird die beabsichtigte Bodennutzung des Gebietes einer Gemeinde dargestellt (§5 BauGB). Er ist der vorbereitende Bauleitplan und damit ein zentrales Planungsinstrument der Kommune und hat üblicherweise eine Lebensdauer von 10 bis 15 Jahren, während der er in Teilen fortgeschrieben wird. Nach diesem Zeitraum wird der FNP in der Regel auf eine Gesamtneuerstellung hin geprüft. Da der aktuelle FNP der kreisfreien Stadt Jena aus dem Jahre 2006 datiert, wurde im Jahre 2015 mit einer Neuerstellung begonnen. Das BauGB fordert für den FNP lediglich eine „Darstellung“ der Flächen. Konsequenterweise war der letzte FNP 2006 eine digitale (CAD-)Zeichnung. Dieses Vorgehen hatte den Nachteil, dass der FNP nicht in einem GIS weiterzuverarbeiten war, insbesondere konnte er nicht mit anderen Daten verschnitten werden, weil z.B. Polygone nicht geschlossen waren. Die Neuerstellung sollte deshalb in einem GIS erfolgen, auch um den Plan später einfach über WMS zur Verfügung stellen zu können (INSPIRE-Richtlinie).

Konzept

Am Anfang stand die Überlegung, wie die Daten für einen FNP in einem GIS sinnvoll modelliert werden können. Ein möglicher Weg wäre es gewesen, die darzustellenden Informationen zu analysieren und ein entsprechendes GIS-Datenmodell zu entwickeln. Die andere Möglichkeit war die Benutzung eines vorhandenen und publizierten Datenmodells. Im Projekt XPlanung wird seit mehreren Jahren das Datenaustauschformat XPlanGML entwickelt, das den verlustfreien Austausch von raumbezogenen Planwerken ermöglichen soll [1]. Obwohl XPlanung schon seit 2008 vom Deutschen Städtetag und vom Deutschen Städte- und Gemeindebund zur Einführung empfohlen ist [2], ist es relativ unbekannt und wird in der kommunalen Raumplanung wenig genutzt. Dies kann daran liegen, dass, wie oben bereits beschrieben, gesetzlich nur eine Planzeichnung gefordert ist, was dann zu einer rein zeichnerischen Planerfassung führt. Andererseits könnte es aber auch daran liegen, dass eine Datenerfassung nach XPlanung deutlich aufwändiger ist, weil einzelne Klassen leicht 20 oder mehr Attribute haben können. Weiterhin muss ein gewisses Verständnis für das Datenmodell beim Datenerfasser vorhanden sein. Die Datenerfassung nach XPlanung ist also eine ungleich komplexere Aufgabe als eine rein zeichnerische Darstellung.

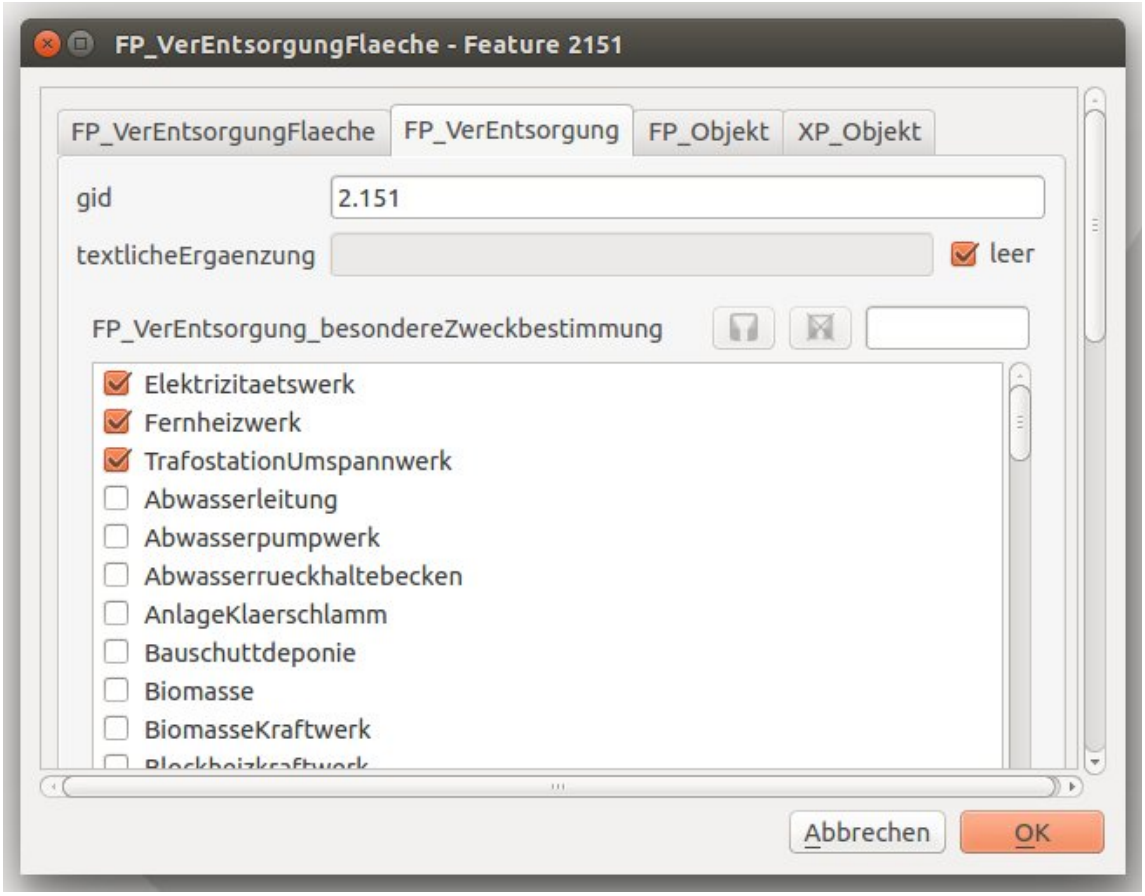


Abbildung 2: Datenmaske eines Features der Tabelle *FP_VerEntsorgungFlaeche*

Umsetzung

Die Stadt Jena hat seit mehreren Jahren das freie Desktop-GIS QGIS im Einsatz, als Datenspeicher dient eine PostgreSQL/PostGIS-Datenbank. Da es bisher keine frei verfügbare Umsetzung des XPlanungs-Standards für PostGIS gibt, wurde der Standard (Version 4.1) durch den Autor in der Datenbank implementiert [3]. Jede Objektart wird dabei in der Regel durch eine eigene Tabelle realisiert (Abb. 1); Trigger sorgen für die Datenkonsistenz zwischen „Eltern“- und „Kind“-Tabellen. Um von QGIS aus mit der XPlanungs-Datenbank interagieren zu können, wurde das Plugin *XPlanung* [4] entwickelt; die Eingabe von Attributdaten (Abb. 2) erfolgt über das QGIS-Plugin *DataDrivenInputMask* [5, 6]. Für jede FNP-Objektart gibt es eine Standarddarstellung in Anlehnung an die PlanZVo. Nutzer können jedoch zusätzlich eigene Darstellungsstile zu einem Layer speichern, die entweder generell oder nur in bestimmten Bereichen eines Plans zur Anwendung kommen sollen; Bereiche sind räumlich oder inhaltlich definierte Teile eines Plans.

Abbildung 3: Farbdarstellung FNP (Arbeitsstand)

Fazit

Für jeden Sonderfall gab es bisher eine sinnvolle Lösung innerhalb des Standards; da XPlanung bereits alle im Zusammenhang mit einem FNP denkbaren Inhalte abbildet, war es noch nicht nötig, weitere Klassen oder Attribute hinzuzufügen. Weiterhin ermöglicht XPlanung die redundanzfreie Erfassung ein und des selben Objekts, indem Objekte, die in mehreren Plänen (Bereichen) dargestellt werden, nur einmal erfasst werden müssen. Die Sachdaten lassen sich in QGIS gut editieren und eine PlanZVo-konforme Darstellung ist mit den umfangreichen Darstellungsoptionen von QGIS möglich (Abb. 3). Zukünftig sollen die derart systematisch erfassten Daten z.B. für Flächenauswertungen genutzt und auch im Internet über WMS bereitgestellt werden. Als Nachteil erwies sich bisher insbesondere die Tatsache, dass die Datenerfassung eine komplexe Aufgabe ist, die entsprechend qualifiziertes Personal und hohe Konzentration bei der Arbeit voraussetzt.

Noch gar nicht realisiert sind z.Zt. Schnittstellen, also die Übernahme von Plänen im XPlanGML-Format bzw. die Ausgabe eigener Pläne als XPlanGML. Sowohl speziell in diesem Punkt als auch generell ist die Mitarbeit von Dritten ausdrücklich erwünscht! Haben Sie Interesse an der Weiterentwicklung, so wenden Sie sich bitte an den Autor.

Kontakt zum Autor:

Bernhard Ströbl
Kommunale Immobilien Jena
Am Anger 26, D-07743 Jena
03641/49-5190
bernhard.stroebl@jena.de

Literatur

[1] XPlanung-Homepage http://www.iai.fzk.de/www-extern/index.php?id=679 abgerufen am 25.05.2016

[2] Krause, K.-U. (2012): Planwerke für Europa. In: Vereinigung für Stadt-, Regional- und Landesplanung (SRL) e.V. (Hrsg.): PLANERIN Heft 5_12, S. 18 ff

[3] https://github.com/bstroebl/xplanPostGIS

[4] https://github.com/bstroebl/xplanplugin

[5] http://plugins.qgis.org/plugins/DataDrivenInputMask/

[6] Ströbl, B (2013): QGIS-Eingabemasken für PostGIS-Layer leichtgemacht. In: FOSSGIS e.V. (Hrsg.): FOSSGIS 2013, Anwenderkonferenz für Freie und Open Source Software für Geoinformationssysteme, S. 115 ff

Zusammenspiel von GIS und CMS verdeutlicht die möglichen Folgen einer 2° Klimaerwärmung

Ing. Carina Waidhofer; Mag. Christoph Haselberger; Nikolaus Dürk, MAS; DI Philippe Brandner.

IMPACT2C Web-Atlas verknüpft Open Source Geoinformationssysteme (GIS) mit einem Content Management System (CMS)

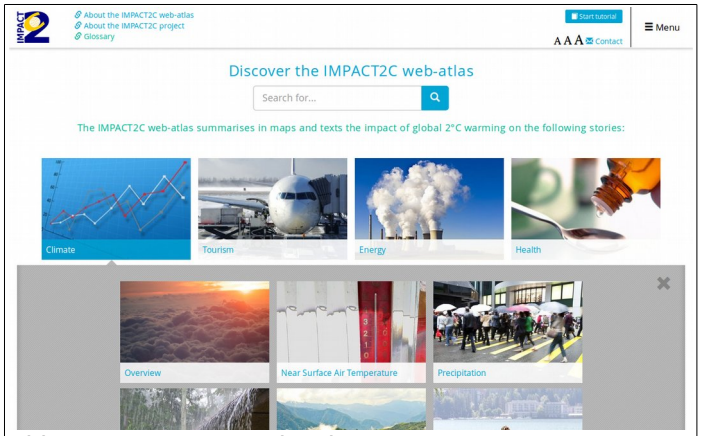


Abb. 1: IMPACT2C Web-Atlas

Die Linzer Unternehmen X-Net Services GmbH und blp GeoServices GmbH haben die technische Entwicklung des öffentlich zugänglichen IMPACT2C Web-Atlas im Auftrag des Climate Service Center Germany (GERICS) realisiert. Darin werden die möglichen Auswirkungen einer 2°C Klimaerwärmung visualisiert und in einem Dual-Screen (Map und Pages) für die Allgemeinheit verständlich erklärt. Durch die Darstellung in mehreren Layern kann die Situation vor und nach einer 2°C Klimaerwärmung verglichen werden. Der Klimawandel ist gesellschaftlich relevant. Die Auswirkungen auf den einzelnen Menschen sind schwer abzuschätzen. IMPACT2C nähert sich dieser Problematik, indem sie die Auswirkungen für spezielle Sektoren und darin erstehende Fragen aufgreifen. Anhand dieses Beispiels können neue Ergebnisse/Fragestellungen problemlos aufgenommen werden.

Der IMPACT2C Atlas ist in verschiedene Themenbereiche, sogenannte Stories gegliedert, u.a. Klima, Energie, Gesundheit, Tourismus, außereuropäische Hotspots. Innerhalb eines Themenbereichs sind weitere Inhalte (Topics) aufrufbar, z.B. erwartete Änderungen im Wintertourismus für Österreich. Die Inhalte des Atlas lassen sich über das CMS leicht erweitern. Zur Darstellung werden Pages des CMS (rechts) einem Set von Karten (links) in einem Dual-Screen zugeordnet.

Das Backend stellt einen eigenen Upload-Bereich für das IMPACT2C Projektteam zur Verfügung, um NetCDF-Files etc. einzustellen. Mittels Preview-Funktion kann der Benutzer direkt feststellen, ob die Daten richtig positioniert sind. Karten können einer Timeline zugeordnet werden, um einen Vergleich vor und nach einer 2°C und 3°C Klimaerwärmung zu ermöglichen. Dazu können parallele Darstellungen mehrerer Karten auf einem Screen genutzt werden.

Für den Web-Atlas wurde ein einzigartiges technisches Konzept entwickelt, das unterschiedliche Open Source Komponenten wie Geographische Informationssysteme, Content-Management-Systeme etc. miteinander verbindet. Alle Systeme, inklusive GIS, Datenbanken und Karten-Upload-Bereiche sind auf der kundeneigenen, skalierbaren IT-Infrastruktur installiert.

Ein Geoserver stellt die Karten bereit. In einer PostgreSQL Datenbank mit PostGIS Extension sind die Kartenmaterialien gespeichert. Zur Visualisierung der unterschiedlichen Kartenansichten kommt OpenLayers3 zum Einsatz. Das CMS ist mit dem auf Python basierenden DjangoCMS realisiert.

Der IMPACT2C Web-Atlas wurde 2015 gemeinsam von den Linzer Unternehmen X-Net Services GmbH und blp GeoServices GmbH im Auftrag des Climate Service Center Germany technisch umgesetzt. ForscherInnen aus 17 Nationen stellen ihre Ergebnisse und Inhalte ein und ergänzen diese – unabhängig von Plattformen und Endgeräten. Das über einen speziellen Upload-Bereich importierte Kartenmaterial wird dem GeoServer bereitgestellt, in dem die Zuordnung der Farbcodierung zu den

einzelnen Karten und die Zusammenführung der unterschiedlichen Daten zu einem Atlas erfolgt. Der Datenbestand liegt aufgrund unterschiedlicher Modelle und Formate sehr heterogen vor.

Previews im Intranet von IMPACT2C erlauben, die eingestellten Daten und Karten zu prüfen und gegebenenfalls zu überarbeiten. Das CMS bringt Texte, Grafiken und Karten in ein benutzerfreundliches Format und erleichtert das Bearbeiten. Mit dem leicht verständlichen Portal können die Projektpartner von IMPACT2C ihre Inhalte über eine Web-Verbindung regelmäßig mit aktuellen Ergebnissen ergänzen und erweitern.

Der IMPACT2C Web-Atlas wurde ausschließlich mit Open Source Komponenten realisiert.

Der Vortrag beschreibt die eingesetzten Komponenten und deren Zusammenspiel und gibt einen Einblick über die Herausforderungen in der Entwicklung, in den Betrieb des Web-Atlas Projektes, sowie in zukünftige Entwicklungen (z.B. interaktive Verbindung zwischen Chart und Map, Anbindung an weitere CMS Systeme wie Plone etc.).

Kontakt zu den Autoren:

Ing. Carina Waidhofer; Nikolaus Dürk, MAS
X-Net Services GmbH
Elisabethstraße 1, 4020 Linz
Telefon: +43 (0) 732 77 31 42 0
eMail: office@x-net.at

Mag. Christoph Haselberger; DI Philippe Brandner
blp GeoServices GmbH
Kapuzinerstrasse 84e, 4020 Linz
Telefon: +43 (0) 732 99 70 04
eMail: office@blpgeo.at

Literatur

[1] *Django-Software-Foundation*: Django (version 1.5), 2015 [computer software]. URL http://django-project.com/

[2] *Lauber, P.:* django cms (version 3.0), 2015 [web publishing platform]. URL http://django-cms.org/

[3] *Python-Software-Foundation:* Python (version 3.5), 2015 [programming language]. URL www.python.org

[4] *Refractions-Research, Ramsey, P., Blasby, D., Neufeld, K., Cave-Ayland, M., Obe, R., Santilli, S., Courtin, O., Avn, N., Park, B., Racine, P., Lounsbury, J., Hodgson, C., Arvalo, J., Loskot, M., Vine, N., Anderson, C., Mason, R., Foerster, K., III, B. W., Schaber, M*.: Postgis (version 2.2), 2015 [backend spatial database geographic information system]. URL http://postgis.net

[5] *The-OpenLayers-Development-Team*: Openlayers (version 3), 2015 [web mapping library]. URL www.openlayers.org

[6] *The-PostgreSQL-Global-Development-Group*: Postgresql (version 9.1), 2015 [object-relational database system]. URL http://postgresql.org

[7] *Open Source Geospatial Foundation:* GeoServer User Manual (version 2.6), dass , 2015 [gis server]. URL http://geoserver.org

Neuerungen im GeoServer

Daniel Koch

Der GeoServer ist ein bekannter und mächtiger OpenSource Kartenserver. Er ermöglicht die Veröffentlichung von Geodiensten aus zahlreichen Datenquellen auf Basis offener Standards.

Die sehr aktive GeoServer Community arbeitet laufend an Erweiterungen und Verbesserungen der Kernsoftware. Dieser Vortrag wird sich einigen (Neu-)Entwicklungen der jüngeren Vergangenheit widmen und an praktischen Beispielen den Nutzen vorstellen. Hierunter fallen u.a.:

•Die Importer Extension zum Hinzufügen von Geodaten in den GeoServer über das Webinterface.
•Die CSS Extension zum Stylen von Layern über Cascading Style Sheets.
•Der WFS (Web Feature Service) Datenspeicher zur Kaskadierung entfernter WFS-Server.
•Die Darstellung von Curved Geometries.
•Die GeoFence Integration.

Neue Werkzeuge für INSPIRE

JÜRGEN WEICHAND

Einleitung

Der INSPIRE-Zeitplan sieht die initiale Bereitstellung von INSPIRE-konformen Daten – für Themen aus Anhang I der Richtlinie – bis spätestens November 2017 vor. Für die Realisierung der benötigten Darstellungs- und Downloaddienste und die Durchführung der hierfür benötigen Datenmodelltransformationen stehen immer geeignetere Verfahren und Werkzeuge im FOSS-Bereich zur Verfügung. Hierzu werden im Folgenden zwei mögliche Workflows vorgestellt.

Workflow 1: Umsetzung einer on-the-fly Transformation mit HALE und GeoServer

Die Bereitstellung von INSPIRE-Datenmodellen auf Grundlage von existierenden Geodatenbanken („on-the-fly Transformation“) ist möglich, wenn bei der Datenmodelltransformation keine komplexen Transformationsfunktionen benötigt werden (vgl. [1]).

Die beliebte OpenSource-Software GeoServer [2] unterstützt die objektrelationale Abbildung („Mapping“) von komplexen Datenmodellen auf Grundlage bestehender Geodatenbanken. Hierzu ist die Installation der GeoServer-Erweiterung für die Unterstützung von Applikationsschemata [3] notwendig. Die Konfiguration des Mappings erfolgt GeoServer-untypisch über XML-Konfigurationsdateien und nicht über die Administrationsoberfläche.

HALE (The HUMBOLDT Alignment Editor) [4] unterstützt seit Version 2.9.4 die GeoServer-Erweiterung für Applikationsschemata und ermöglicht hierdurch ein bequemes Mapping zwischen bestehender Datenquelle und GML-Schema. Das Mapping („Alignment“) kann als Dienstekonfiguration exportiert und direkt an den GeoServer übertragen werden (vgl. Abb. 1). Aufgrund der Restriktionen einer on-the-fly Transformation stehen nicht alle in HALE vorhandenen Transformationsfunktionen zur Verfügung [5].

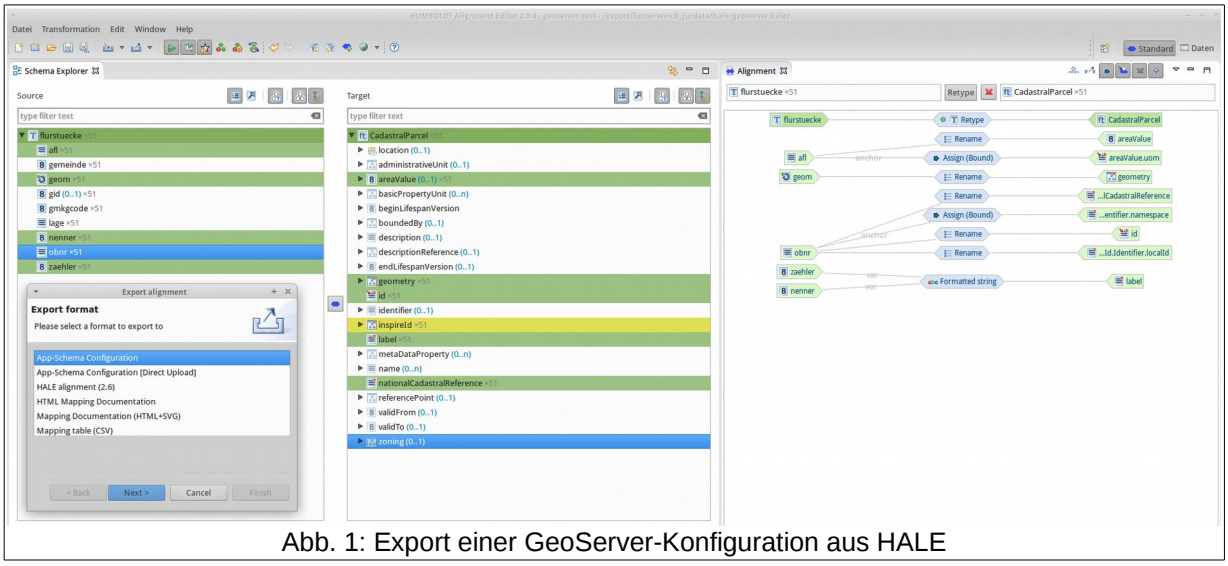


Abb. 1: Export einer GeoServer-Konfiguration aus HALE

Workflow 2: Umsetzung einer Sekundärdatenhaltung mit HALE und deegree

Werden komplexe Transformationsfunktionen (vgl. [1]) benötigt, ist der Aufbau einer Sekundärdatenhaltung notwendig. Dies kann beispielsweise der Fall sein, wenn mehrere Quellobjekte (z. B. Flurstücke) zu einem Zielobjekt (z. B. Gemarkung) aggregiert werden müssen.

Die OpenSource-Software deegree WebServices [6] ermöglicht die automatische Erstellung einer Sekundärdatenhaltung auf Grundlage eines GML-Applikationsschemas (vgl. Abb. 2). Das Ergebnis ist ein Datenbankschema sowie die korrespondierende Dienstekonfigurationen für einen transaktionellen Web Feature Service (WFS-T) [7].

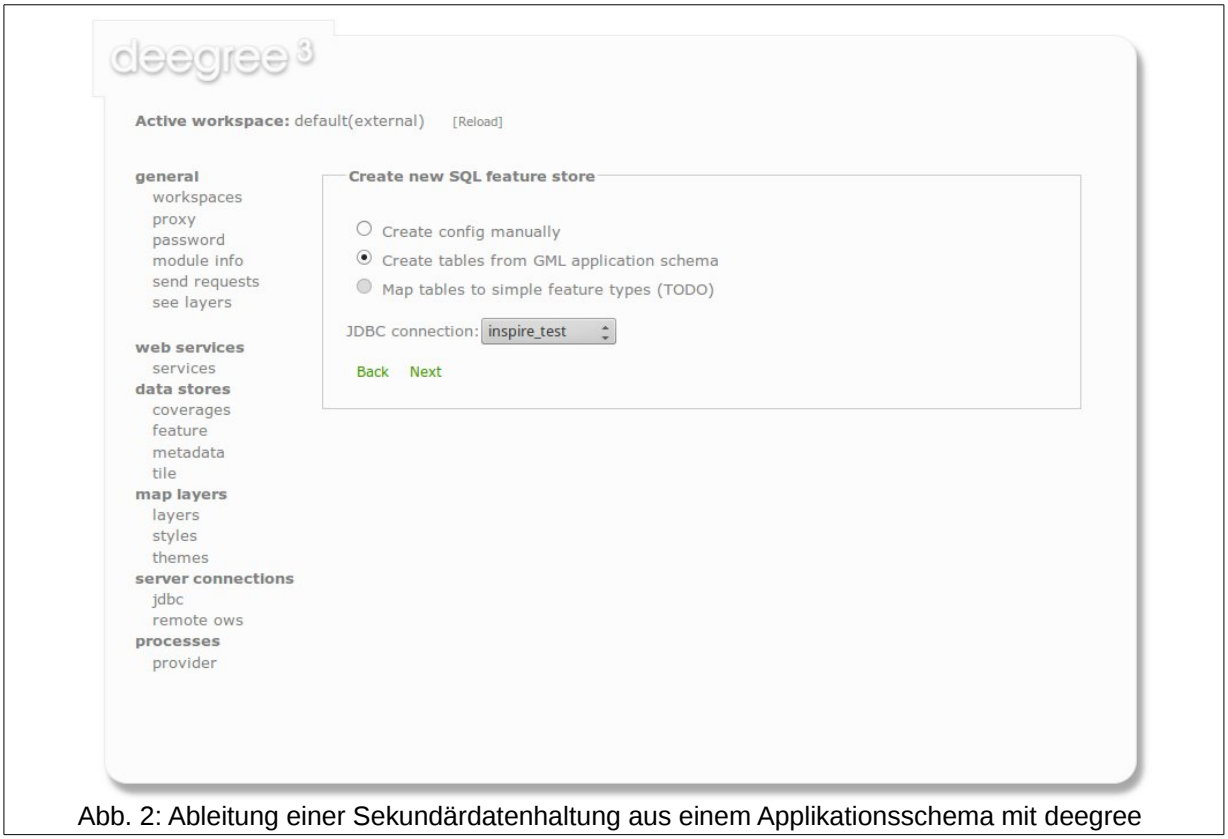


Abb. 2: Ableitung einer Sekundärdatenhaltung aus einem Applikationsschema mit deegree

Anschließend können die ins Zielschema überführten Geodaten (valides GML) über WFS-T in die Sekundärdatenhaltung importiert werden. Hierzu steht in HALE ein Exportwerkzeug zur Verfügung, das die transformierten Geodaten zusätzlich in für WFS-T geeignete Portionen aufteilt.

Fazit

Bei der Evaluierung von Transformationswerkzeugen und -verfahren sollte die spätere Bereitstellung über Darstellungs- und Downloaddienste bereits berücksichtigt werden. Ohne die gemeinsame Betrachtung aller Komponenten (Transformation + Dienstebereitstellung) werden Workflows verkompliziert und häufig zusätzliche, redundante Datenhaltungen notwendig.

Mittels GeoServer und deegree können komplexe Datenmodelle über Darstellungs- und Downloaddienste bereitgestellt werden. Weiterhin steht mit HALE ein komfortables Werkzeug für die Durchfüh-

rung von Datenmodelltransformationen zur Verfügung, insbesondere bei der Verwendung vom GML-Applikationsschemata.

Mit den OpenSource-Stacks GeoServer/HALE und deegree/HALE können durchgehende Workflows für die Bereitstellung INSPIRE-konformer Daten und Dienste realisiert werden (vgl. Tab. 1).

Tab.1: Workflows für die Bereitstellung komplexer Feature-Modelle

	GeoServer 2.9.0	**deegree 3.3.18**
Realisierung einer on-the-fly Transformation durch manuelle Konfiguration	Ja	Ja
Realisierung einer on-the-fly Transformation durch Konfiguration über HALE	Ja	Nein
Import von komplexem GML über WFS-T (z. B. über HALE)	Nein	Ja

Kontakt zum Autor

Jürgen Weichand
Landesamt für Digitalisierung, Breitband und Vermessung
Alexandrastraße 4, 80538 München
juergen.weichand@ldbv.bayern.de

Literatur

[1] *Weichand, Jürgen*: **Herausforderungen bei der Umsetzung der INSPIRE-Richtlinie**, Münster, 2015, http://www.weichand.de/download/fossgis_2015_INSPIRE-Herausforderungen-Weichand.pdf, letzter Zugriff: 31.05.2016

[2] *GeoServer:* **GeoServer Webseite**, http://geoserver.org, letzter Zugriff: 31.05.2016

[3] *GeoServer User Manual:* **Application schemas**, http://docs.geoserver.org/latest/en/user/data/app-schema/index.html, letzter Zugriff: 31.05.2016

[4] *HALE:* **HALE - The HUMBOLDT Alignment Editor**, http://www.dhpanel.eu/humboldt-framework/hale.html, letzter Zugriff: 31.05.2016

[5] *Simone Giannecchini:* **Developer's Corner: Simplify Complex Features with GeoServer and HALE**, 2015, http://www.geo-solutions.it/blog/developers-corner-simplify-complex-features-with-geo-server-and-hale/, letzter Zugriff: 31.05.2016

[6] *deegree:* **deegree WebServices**, http://www.deegree.org, letzter Zugriff: 31.05.2016

[7] *deegree Documentation:* **Features Stores**, http://download.deegree.org/documentation/3.3.18/html/featurestores.html#features-feature-types-and-application-schemas, letzter Zugriff: 31.05.2016

INSPIRE „instant“

ARMIN RETTERATH

Die Umsetzung der EU INSPIRE-Richtlinie zur Schaffung einer Geodateninfrastruktur auf der Ebene der Europäischen Union [1] beschäftigt die Öffentliche Verwaltung nun schon geraume Zeit. Der notwendige Rechtsrahmen, der die verbindlichen Vorgaben zur Bereitstellung von Geodaten definiert, existiert ebenfalls schon seit über 4 Jahren.

Schaut man sich die bisher publizierten Daten im INSPIRE Geoportal [2] genauer an, so fällt eine gewisse Heterogenität im Datenangebot auf. Vor allem kleinere Institutionen sind unterrepräsentiert.

Es stellt sich die Frage nach den Hinderungsgründen – warum läuft die Umsetzung so schleppend?

Spricht man mit betroffenen Institutionen und fragt nach den Ursachen, so hört man oft Folgendes:

1. Die zugrundeliegende Architektur ist zu komplex (es gibt keine Software, die die Anforderungen von INSPIRE *out of the box* erfüllt).
2. Man benötigt eine umfangreiche Infrastruktur (z.B. Datenbanken, Mapserver, Webserver sowie das notwendige KnowHow – was natürlich grundsätzlich kosten- und personalintensiv ist).
3. Für eine konkrete Bereitstellung von Geodaten über OGC-Interfaces, müssen die für die IT und die für die Daten zuständigen Organisationseinheiten eng zusammenarbeiten (leider sprechen die meist unterschiedliche Sprachen).
4. Man sieht in der standardisierten Bereitstellung - auf den ersten Blick - keinen Vorteil für die eigene Institution (aufgrund dessen wird das Thema in den meisten Fällen sehr stiefmütterlich behandelt).

Mit dem Aufkommen von cloud-basierten Softwareangeboten ergibt sich aktuell eine Chance die unter den Punkten 2. und 3. aufgeführten Probleme auszuschließen. Man lagert die Technologie einfach in die Cloud aus. Für Organisationen die keine Berührungsängste haben, ihre Daten von einem externen Anbieter hosten zu lassen, kann dieser Weg eine Alternative darstellen.

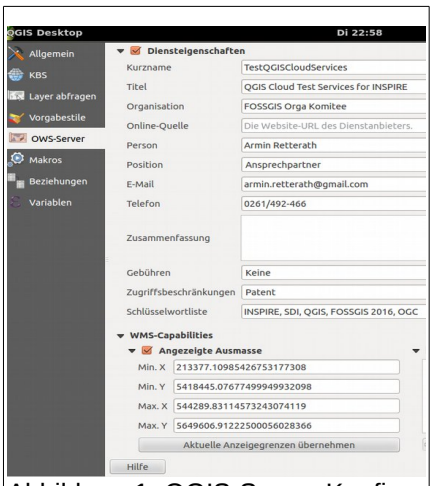


Abbildung 1: QGIS Server Konfiguration

Im Rahmen einer Live-Präsentation wird gezeigt, wie man auf einfachste Weise frei verfügbare Daten mittels QGIS Cloud und dem GeoPortal.rlp für INSPIRE bereitstellen kann. Im ersten Schritt wird ein - unter einer Open Data Lizenz stehender - Datensatz zu adminstrativen Einheiten aus dem Netz geladen und als QGIS Layer visualisiert. Im Anschluss daran erfolgt die Konfiguration der WFS und WMS Schnittstellen in QGIS sowie die Veröffentlichung der Daten über QGIS Cloud. Um die Dienste konform zu den Vorgaben der EU INSPIRE-Richtlinie bereitzustellen, werden sie im GeoPortal.rlp registriert und mit Metadaten angereichert. Zum Abschluss des Prozesses stehen Metadaten und Dienste zur Verfügung, die den Anforderungen der EU-Richtlinie, sowie deren Durchführungsbestimmungen genügen. Der gesamte Vorgang dauert keine 20 Minuten und zeigt auf, dass die Erfüllung der Anforderungen von INSPIRE kein Hexenwerk darstellt, und die vermeintlich komplexen Vorgaben auch ohne den Einsatz lizenzkostenpflichtiger Softwa-

re einfach umzusetzen sind. Die Verwendung von QGIS Cloud stellt - aufgrund seiner offenen OGC-Schnittstellen - eine Alternative zum Betrieb eigener Webserver dar. Das Verfahren eignet sich daher insbesondere für kleinere Behörden bzw. Institutionen, die nur Geodaten von geringem Umfang bereitstellen müssen.

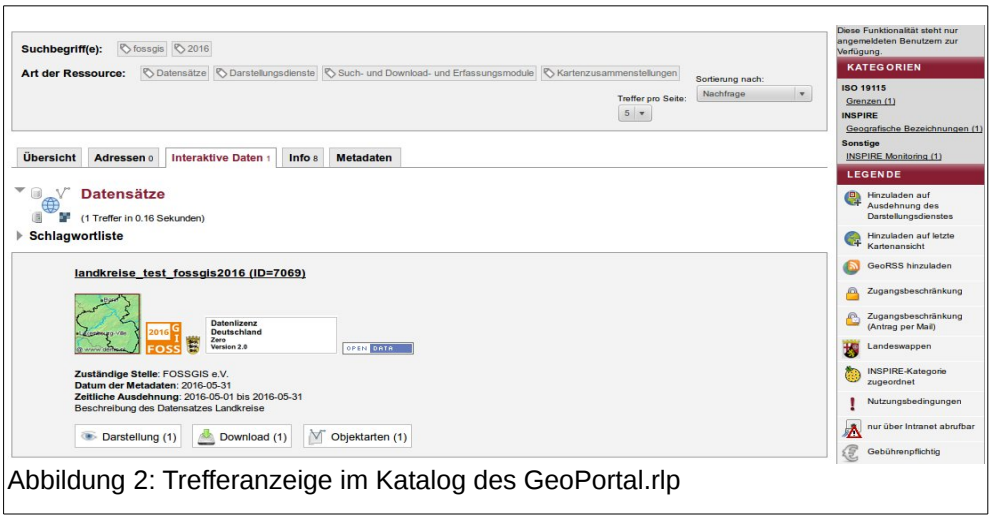


Abbildung 2: Trefferanzeige im Katalog des GeoPortal.rlp

Kontakt zum Autor:

Armin Retterath
Zentrale Stelle GDI-RP
56070 Koblenz
0261/492-466
armin.retterath@lvermgeo.rlp.de

Weitere Informationen

[1] Richtlinie 2007/2/EG des Europäischen Parlaments und des Rates vom 14. März 2007 zur Schaffung einer Geodateninfrastruktur in der Europäischen Gemeinschaft (INSPIRE) - http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:108:0001:0014:DE:PDF

[2] INSPIRE Geoportal - http://inspire-geoportal.ec.europa.eu/

[3] Geoportal Rheinland-Pfalz- http://www.geoportal.rlp.de

[4] QGIS Dokumentation - http://planet.qgis.org/planet/tag/wfs/

Summer of Code

Peter Barth

Seit vielen Jahren nimmt das OpenStreetMap-Projekt erfolgreich am Google Summer of Code teil. Dabei wurden bislang 33 verschiedene Projektideen von Studenten bearbeitet und von Communitymitgliedern betreut. Der Projektverlauf ist dabei sehr heterogen. So gibt es von sehr einfachen bis zu sehr komplexen Aufgaben eine große Spannweite. Auch die Studenten selbst unterscheiden sich stark in Bezug auf Herkunft, Studiengang und Fachsemester. Dieser Vortrag gibt einen detailierten Überblick über die bisherigen Teilnahmen von OpenStreetMap am Google Summer of Code. Neben der Darstellung der Fakten soll aber insbesondere auch aufgezeigt werden, warum die bisherige Beteiligung am Summer of Code für OpenStreetMap von Vorteil war. Vor allem aber soll der Vortrag dazu motivieren, dass OpenStreetMap weiterhin teilnimmt, dass sich langjährige, erfahrene Community-Mitglieder als Mentoren melden, aber auch, dass sich Studenten für eines der tollen Projektideen bewerben.

Bereits seit 2008 nimmt das OpenStreetMap-Projekt als Organisation am Google Summer of Code teil und hat in diesem Rahmen viele interessante und nützliche Projekte auf den Weg gebracht, aber auch dazu beigetragen die OpenStreetMap-Community zu vergrößern. Der Vortrag gibt einen Überblick über die vergangenen Jahre, die geleistete Arbeit, die positiven wie negativen Aspekte der Teilnahme, zeigt aber vor allem auch auf, warum sowohl Studenten als auch OpenStreetMap von Google Summer of Code profitieren.

OSM schön gemacht

Mark Padgham

Für viele Bereiche wäre es wünschenswert, Basiskarten aus OSM-Quellen in beliebigen Formaten visuell darstellen zu können. Aktuell ist diese Herausforderung mit Free oder Open Source Software nicht einfach zu erfüllen. In diesem Vortrag möchte ich ein neues R package namens “osmplotr” vorstellen. Osmoplotr ermöglicht die grafisch beliebige Darstellung ausgwählter OSM-Daten wie beispielsweise Straßen, Gebäude, Bäume oder Grünflächen. Dabei wird die Reihenfolge der Belegung vom User definiert, sodass z.B. Bäume über Gebäuden über Straßen belegt werden können. Der wesentliche Vorteil von “osmoplotr” ist jedoch, dass User definierte Daten direkt durch OSM Elemente visualisiert werden können. Traditionelerweise werden Daten oberhalb eine Karte dargestellt, sodass die graphische Formen von Karte und Daten unvermeidbar unterschiedlich sind – Karten im Hintergrund und Daten im Vordergrund. Mit „osmplotr“ ist es möglich, die OSM Grundbestandteilie einer Region durch Farben oder andere graphische Eigenschaften an User definierte Daten anzupassen (siehe Beispiel unten). Darüber hinaus ermöglichst „osmplotr“, ausgewählte Regionen mit Hilfe eines kontrastierenden Hintergrundes vergleichend dargestellt werden zu können. Diese Funktion ist besonders vorteilhaft für Personen, die in der räumlichen Analyse tätig sind und damit fokale Bereiche innerhalb eines visuell umfassenden Umfeldes kontextualisieren können. Fokalbereiche können durch unterschiedliche osmoplotr-Funktionen bestimmt werden. Es ist auch auch möglich, den Namen von umkreisenden Straßen einzutragen, damit osmoplotr den kürzesten Kreisweg ausrechnet, den alle genannten Straßen umlaufen. Durch die Bestimmung einer oder mehrerer Fokalbereiche kann eine Karte neu gestaltet werden, wobei die Fokalbereiche beliebig anders dargestellt werden als der kontrastierende Hintergrund.

Kontakt zum Autor:

Dr Mark Padgham
Fachbereich Geoinformatik, z_gis
Universität Salzburg
5020 Salzburg, Österreich
+4366280447546
mark.padgham@email.com

OSM2World hinter den Kulissen

Tobias Knerr

Mit OSM2World steht der Community seit Jahren eine freie Software zur 3D-Visualisierung von OpenStreetMap-Daten zur Verfügung. Im Laufe der Zeit sind dabei immer anspruchsvollere Darstellungen hinzugekommen, die auch interessante softwaretechnische Problemstellungen aufwerfen. Dabei geht OSM2World oft einen anderen Weg als die Welt des 2D-Renderings, auch wenn sich bisweilen überraschende Synergie-Effekte und Parallelen ergeben.

In diesem Vortrag beschreibt OSM2World-Gründer Tobias Knerr, wie OSM2World mit solchen Herausforderungen umgeht. Darunter finden sich in der OpenStreetMap-Community altbekannte Probleme wie das Processing von Küstenlinien ebenso wie spezifische Herausforderungen beim 3D-Rendering von Straßenflächen, detaillierten Gebäudemodellen oder der Unterstützung neuer OpenGL-Funktionalitäten.

Bei all dem soll aber auch die Perspektive des Mappers nicht zu kurz kommen. Daher ist auch ein Blick auf die zugrundeliegenden Tagging-Schemata fester Teil des Vortrags.

Dieser Vortrag widmet sich den Herausforderungen, die für die immer anspruchsvolleren 3D-Darstellungen in OSM2World zu bewältigen waren. Der freie 3D-Renderer geht dabei oft einen anderen Weg als die Welt der 2D-Karten.

Zu den Themen zählen das Zeichnen von Küstenlinien, Straßenflächen und detaillierten Gebäuden ebenso wie neue OpenGL-Funktionalitäten und Tagging-Schemata.

Transit-Routing und OSM

Dr. Arndt Brenschede

Der Kurzvortrag zum intermodalem Rad/Fuss/ÖNPV-Routing auf der letztjährigen FOSSGIS in Münster konnte nur kurz darstellen, was sich am Markt für Transit-Routing tut bzw. nicht tut, was für Fehler die verfügbaren Lösungen machen und welche Anwendungsfälle sich damit nicht realisieren lassen. Kurz darauf hat dasselbe Thema, temporär erweitert um Fernverkehrsdaten, den Preis in der Kategorie „Usability“ beim ersten Deutsche Bahn Hackathon gewonnen. Dieser Vortrag knüpft daran an und geht einen Schritt weiter: Systeme aus OSM und unfreien Transit-Daten sind nicht mehr ungewöhnlich, gleichzeitig tut sich was in Richtung freie Transit-Daten. Wie sieht die Killer-App aus, die aus solchen Daten wächst, und was wird OSM ausser dem Wegenetz dazu beitragen? Welche weiteren Verkehrsdienstleistungen werden enthalten sein? Dieser Vortrag kann diese Fragen nicht wirklich beantworten. Aber er kann deutlich machen, dass bessere Informationssyteme den nicht-individuellen Verkehr stärken und einen Beitrag zum Klimaschutz und zur Lebensqualität in urbanen Gebieten leisten können.

Transit-Routing ist in Bewegung und OSM spielt eine zunehmende Rolle dabei. Was kann OSM ausser dem Wegenetz dazu beitragen, was ist der Unique-Selling Point und warum ist das Thema mit Öffie, Google-Transit und der App vom Verkehrsverbund XY doch noch nicht entgültig besetzt?

Morituri

Philip Beelmann

Es gibt bereits viel freie Software, die mit Geodaten im OpenStreetMap-Format umgehen kann. Kommerzielle Geodaten hingegen können oft nur mit proprietärer Software genutzt werden. Besitzer kommerzieller Geodaten hatten daher bisher keine Wahl.

Insbesondere erschwert das auch die Migration - wer kommerzielle Geodaten einsetzt und einen Schwenk auf OSM erwägt, muss Datenmaterial und Software gleichzeitig tauschen.

In diesem Vortrag wird die freie Software "morituri" vorgestellt, die kommerzielle Geodaten in das OSM-Format konvertieren kann. Auf diese Weise kann man freie OSM-Software -

wie beispielsweise die Routing-Engines Graphhopper oder OSRM, aber auch Geocoder wie Nominatim und Rendering-Software - zusammen mit konvertierten kommerziellen Geodaten nutzen. Dabei ist zu beachten, dass diese Daten nicht(!) für den Upload zu OSM, sondern ausschließlich zur Verwendung mit anderer Software vorgesehen sind.

Derzeit werden nur Daten von HERE unterstützt. Da morituri eine Plugin-Architektur hat, kann man es einfach mit zusätzlichen Plugins um weitere kommerzielle Geodatenformate erweitern.

Die Software entstand im Rahmen des TOTARI-Projektes und ist auf Github verfügbar. Sie unterliegt der GNU General Public License v3.

Der Vortrag erläutert, was "morituri" schon kann, was noch fehlt, an welchen Stellen die Konvertierung von HERE-Daten in OSM-Daten simpel war, und an welchen unterschiedliche Konzepte und Datenmodelle Fallstricke für die Umwandlung dargestellt haben.

Es gibt bereits viel freie Software, die mit Geodaten im OSM-Format umgehen kann. Kommerzielle Geodaten hingegen können oft nur mit proprietärer Software genutzt werden. Besitzer kommerzieller Geodaten hatten daher bisher keine Wahl. In diesem Vortrag wir die freie Software „morituri“ vorgestellt, die derzeit kommerzielle Geodaten von HERE in das OSM-Format konvertieren kann.
So kann man freie OSM-Software - wie beispielsweise die Routing-Engines Graphhopper oder OSRM, aber auch Geocoder wie Nominatim und Rendering-Software - zusammen mit den konvertierten kommerziellen Geodaten nutzen.