

Overpass

Jan-Philipp Kolb

23 Oktober 2018

Themen dieses Abschnitts

- Die **Overpass API** von Roland Olbricht wird vorgestellt.
- Die API **Overpass Turbo**
- Wie man die OSM Daten graphisch darstellen kann.

Die Overpass API

- Die von Roland Olbricht geschriebene Overpass API ermöglicht es Entwicklern, kleine Auszüge von benutzergenerierten Inhalten von Openstreetmap nach vorgegebenen Kriterien herunterzuladen.
- Overpass ist eine read-only API, die durch den Benutzer ausgewählte Teile der OSM-Daten bereitstellt.
- Overpass kann als eine Datenbank über das Internet verstanden werden.
- Die API eignet sich besonders gut, wenn man nach ganz speziellen Map Features sucht.

Overpass Turbo

AusführenTeilenExportWizardSpeichernLadenEinstellungenHilfeoverpass turbo

KarteDaten

```
1  /*
2  This is an example Overpass query.
3  Try it out by pressing the Run button above!
4  You can find more examples with the Load
5  tool.
6  */
7  node
8    [highway=bus_stop]
9    ({{bbox}});
10 out;
```


geladen – Nodes: 39, Ways: 0, Relations: 0
angezeigt – POIs: 39, Linien: 0, Polygone: 0

Query Overpass

- In der folgenden Abfrage wird bei Overpass Turbo nach Bars im ausgewählten Fenster gesucht.

```
node
  [amenity=bar]
  ({{bbox}});
out;
```

Export bei Overpass

Exportieren 


▼ Daten


Speichere/Kopiere als GeoJSON


Speichere/Kopiere als GPX

Speichere/Kopiere als KML

Speichere/Kopiere als OSM Rohdaten

Rohdaten direkt von Overpass API 

In einen OSM-Editor laden: JOSM, Level0 

GeoJSON als gist  speichern

► Karte

Bei Export von Overpass

- GeoJSON
- GPX
- KML
- OSM Rohdaten

Import von Daten

```
library(XML)
dat <- xmlParse("../data/bus_stop_amsterdam.kml")
```

```
xmltop <- xmlRoot(dat)
xmltop[[1]][[1]]
```

```
## <name>overpass-turbo.eu export</name>
```


Xpath Abfragesprache

Beispiel: xpath wikipedia

```
xpathApply(dat, "Document")
```

```
## list()
```

```
## attr("class")
```

```
## [1] "XMLNodeSet"
```

JSON importieren

```
install.packages("rjson")  
library(rjson)
```

```
library(jsonlite)  
dat<-jsonlite::fromJSON("../data/amsterdam_busstop.geojson")  
typeof(dat)
```

```
## [1] "list"
```

```
names(dat)
```

```
## [1] "type"          "generator" "copyright" "timestamp" "feature"
```

Wie sehen die Daten aus

```
DT::datatable(dat$features$properties)
```

Show 10 ▾ entries

Search:

	@id	highway	name	public_transport	zone	cxx:code	cxx:code
1	node/447840083	bus_stop	Leidseplein	platform	5700		
2	node/495568909	bus_stop	Dam	platform	5700	57002550	3
3	node/502341044	bus_stop	Elandsgracht	stop_position			
4	node/534026003	bus_stop	Centraal Station / Nicolaaskerk		5700		
5	node/700343182	bus_stop	Prins Hendrikkade	platform	5700		
6	node/724232554	bus_stop	Dam	platform	5700	57002560	3
7	node/1079768926	bus_stop	Prins Hendrikkade	platform	5700		
8	node/1079768989	bus_stop	IJ tunnel	platform	5700		

GPX file importieren

```
library(plotKML)
```

```
## plotKML version 0.5-8 (2017-05-12)
```

```
## URL: http://plotkml.r-forge.r-project.org/
```

```
dat_gpx <- readGPX("../data/Amsterdam_busstop.gpx")  
head(dat_gpx$waypoints)
```

##		lon	lat	name
## 1	4.880870	52.36213	Leidseplein	
## 2	4.891237	52.37438	Dam	
## 3	4.877558	52.36953	Elandsgracht	
## 4	4.900331	52.37670	Centraal Station / Nicolaaskerk	
## 5	4.905498	52.37395	Prins Hendrikkade	
## 6	4.890181	52.37310	Dam	
##				
## 1				