

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

# Overpass-Workshop

Nakaner

Salzburg  
12. Juli 2016

# Gliederung

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

1 Grundlagen

2 Set

3 Vereinigung

4 Recurse

5 Tags

# Tags abfragen (Grundlagen)

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

```
1 [out:json][timeout:25];  
2 node[amenity=post_box]({{bbox}});  
3 out;
```

- Befehle enden mit einem Semikolon
- Tag-Abfragen in eckigen Klammern
- andere Abfragen (Benutzer, Zeitstempel, räumlich) in runden Klammern
- (50.6,7.0,50.8,7.3) Abfrage für dieses Rechteck
- {{bbox}} ist ein Overpass-Turbo-Makro
- als Nodes gemappte Briefkästen in abgefragtem Gebiet

# Tags abfragen (Grundlagen)

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

```
1 [out:json][timeout:25];  
2 node[amenity]({{bbox}});  
3 out;
```

- alles mit amenity=\*

# Set

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

- Alles wird in einem Set gespeichert
- Default: `._`
- Sets beginnen mit einem Punkt

```
1 [out:json][timeout:25];  
2 node[amenity=post_box]  
3   ({{bbox}})->.briefkaesten;  
4 .briefkaesten out;
```

# Set

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

```
1 [out:json][timeout:25];  
2 node[amenity=post_box]  
3   ({{bbox}}) ->.briefkaesten;  
4 node.briefkaesten[operator];  
5 out;
```

- Ergebnis von amenity=post\_box in *briefkaesten* schreiben
- Briefkästen mit Operator in Set \_ schreiben
- Set \_ ausgeben

# Vereinigung (Union)

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

```
1 [out:json][timeout:25];
2 (
3   node[amenity]({{bbox}});
4   node[shop]({{bbox}});
5 );
6 node._[opening_hours];
7 out;
```

- Vereinigung aus Nodes mit amenity=\* und Nodes mit shop=\* und in Set \_ schreiben
- Set \_ nach opening\_hours=\* filtern und in Set \_ schreiben
- In Zeile 6 muss .\_ angegeben werden, sonst wird global nach Öffnungszeiten gesucht!

# Recurse

Von oben nach unten

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

```
1 [out:json][timeout:25];  
2 way[highway]({{bbox}});  
3 out;  
4 >;  
5 out;
```

- > heißt *Recurse down*
  - Alle Nodes aus der Eingabe, plus
  - alle Nodes und Ways, die Mitglied einer Relation der Eingabe sind, plus
  - alle Nodes von Ways, die zu einem Way aus dem Ergebnis gehören
- >> heißt *Recurse down relations* – Relation, die Relationsmitglieder sind
- < und << existieren auch

# Tags, Tags und Tags

Existenz, Gleichheit und Ungleichheit

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

sehr ähnlich zu MapCSS

Keine gültige Abfrage!

```
1 node [shop] // shop=*
2 node [shop=bakery] // shop=bakery
3 node [shop][wheelchair] // shop=* +
  ↪ wheelchair=*
4 way [highway][surface!=cobblestone] //
  ↪ highway=* + surface!=cobblestone
```

# Tags, Tags und Tags

## Reguläre Ausdrücke

Overpass-  
Workshop

Nakaner

Grundlagen

Set

Vereinigung

Recurse

Tags

Reguläre Ausdrücke sind notwendig für ...

Keine gültige Abfrage!

- 1 `node[shop~"^$"]` // shop="" (leerer Wert)
- 2 `node[shop!~"^Baeckerei"]` // shop=\*, Wert  
↳ beginnt mit "Baeckerei"
- 3 `node[shop~"."]` // gleichbedeutend mit `node`  
↳ `[shop]`
- 4 `node[shop!~"."]` // Nodes ohne shop-Tag
- 5 `node[~"^name:.*$"~"stadt$"]`; // Schluessel  
↳ beginnt mit "name" (name, name:de,  
↳ name:en), Wert endet mit "stadt"