
photon



photon is an open source geocoder built for OpenStreetMap data. It is based on elasticsearch - an efficient, powerful and highly scalable search platform.

photon was started by komoot and provides search-as-you-type and multilingual support. Find our public API and demo on photon.komoot.io. Until October 2020 the API was available under photon.komoot.de. Please update your apps accordingly.

Contribution

All code contributions and bug reports are welcome!

For questions please send an email to our mailing list.

Feel free to test and participate!

Licence

photon software is open source and licensed under Apache License, Version 2.0

Features

- high performance
- highly scalability
- search-as-you-type
- multilingual search
- location bias
- typo tolerance
- filter by osm tag and value
- filter by bounding box
- reverse geocode a coordinate to an address
- OSM data import (built upon Nominatim) inclusive continuous updates

Installation

photon requires Java, at least version 11.

Download the search index (72G GB compressed, 159GB uncompressed as of 2023-10-26, worldwide coverage, languages: English, German, French and local name). The search index is updated weekly and thankfully provided by GraphHopper with the support of lonvia. Now get the latest version of photon from the releases.

Make sure you have bzip2 or pbzip2 installed and execute one of these two commands in your shell. This will download, uncompress and extract the huge database in one step:

```
1 wget -O - https://download1.graphhopper.com/public/photon-db-latest.tar
  .bz2 | bzip2 -cd | tar x
2 # you can significantly speed up extracting using pbzip2 (recommended):
3 wget -O - https://download1.graphhopper.com/public/photon-db-latest.tar
  .bz2 | pbzip2 -cd | tar x
```

Building

photon uses gradle for building. To build the package from source make sure you have a JDK installed. Then run:

```
1 ./gradlew build
```

This will build and test photon. The final jar can be found in `build/libs`.

Usage

Start photon with the following command:

```
1 java -jar photon-*.jar
```

Use the `-data-dir` option to point to the parent directory of `photon_data` if that directory is not in the default location `./photon_data`. Before you can send requests to photon, Elasticsearch needs to load some data into memory so be patient for a few seconds.

Check the URL `http://localhost:2322/api?q=berlin` to see if photon is running without problems. You may want to use our leaflet plugin to see the results on a map.

To enable CORS (cross-site requests), use `-cors-any` to allow any origin or `-cors-origin` with a specific origin as the argument. By default, CORS support is disabled.

Discover more of photon's features with its usage `java -jar photon-*.jar -h`. The available options are as follows:

```
1 -h                      Show help / usage
2
```

```
3 -cluster          Name of elasticsearch cluster to put the server
   into (default is 'photon')
4
5 -transport-addresses The comma separated addresses of external
   elasticsearch nodes where the
6                       client can connect to (default is an empty string
   which forces an internal node to start)
7
8 -nominatim-import  Import nominatim database into photon (this will
   delete previous index)
9
10 -nominatim-update  Fetch updates from nominatim database into photon
   and exit (this updates the index only
11             without offering an API)
12
13 -languages         Languages nominatim importer should import and
   use at run-time, comma separated (default is 'en,fr,de,it')
14
15 -default-language  Language to return results in when no explicit
   language is chosen by the user
16
17 -country-codes     Country codes filter that nominatim importer
   should import, comma separated. If empty full planet is done
18
19 -extra-tags        Comma-separated list of additional tags to save
   for each place
20
21 -synonym-file       File with synonym and classification terms
22
23 -json              Import nominatim database and dump it to a json
   like files in (useful for developing)
24
25 -host              Postgres host (default 127.0.0.1)
26
27 -port              Postgres port (default 5432)
28
29 -database           Postgres host (default nominatim)
30
31 -user              Postgres user (default nominatim)
32
33 -password           Postgres password (default '')
34
35 -data-dir          Data directory (default '.')
36
37 -listen-port        Listen to port (default 2322)
38
39 -listen-ip          Listen to address (default '0.0.0.0')
40
41 -cors-any           Enable cross-site resource sharing for any origin
   (default CORS not supported)
42
```

```
43 -cors-origin      Enable cross-site resource sharing for the
    specified origin (default CORS not supported)
44
45 -enable-update-api Enable the additional endpoint /nominatim-update,
    which allows to trigger updates
46                    from a nominatim database
```

Customized Search Data

If you need search data in other languages or restricted to a country you will need to create your search data by your own. Once you have your Nominatim database ready, you can import the data to photon.

If you haven't already set a password for your Nominatim database user, do it now (change user name and password as you like, below):

```
1 su postgres
2 psql
3 ALTER USER nominatim WITH ENCRYPTED PASSWORD 'mysecretpassword';
```

Import the data to photon:

```
1 java -jar photon-*.jar -nominatim-import -host localhost -port 5432 -
    database nominatim -user nominatim -password mysecretpassword -
    languages es,fr
```

The import of worldwide data set will take some hours/days, SSD/NVME disks are recommended to accelerate Nominatim queries.

Updating from OSM via Nominatim To update an existing Photon database from Nominatim, first prepare the Nominatim database with the appropriate triggers:

```
1 java -jar photon-*.jar -database nominatim -user nominatim -password
    ... -nominatim-update-init-for update_user
```

This script must be run with a user that has the right to create tables, functions and triggers.

'update-user' is the PostgreSQL user that will be used when updating the Photon database. The user needs read rights on the database. The necessary update rights will be granted during initialisation.

Now you can run updates on Nominatim using the usual methods as described in the documentation. To bring the Photon database up-to-date, stop the Nominatim updates and then run the Photon update process:

```
1 java -jar photon-*.jar -database nominatim -user nominatim -password ... -nominatim-update
```

You can also run the photon process with the update API enabled:

```
1 java -jar photon-*.jar -enable-update-api -database nominatim -user nominatim -password ...
```

Then you can trigger updates like this:

```
1 curl http://localhost:2322/nominatim-update
```

This will only start the updates. To check if the updates have finished, use the status API:

```
1 curl http://localhost:2322/nominatim-update/status
```

It returns a single JSON string **"BUSY"** when updates are in progress or **"OK"** when another update round can be started.

For your convenience, this repository contains a script to continuously update both Nominatim and Photon using Photon's update API. Make sure you have Photon started with `-enable-update-api` and then run:

```
1 export NOMINATIM_DIR=/srv/nominatim/...
2 ./continuously_update_from_nominatim.sh
```

where `NOMINATIM_DIR` is the project directory of your Nominatim installation.

Search API

```
1 Search
  http://localhost:2322/api?q=berlin
```

```
1 Search with Location Bias
  http://localhost:2322/api?q=berlin&lon=10&lat=52
```

There are two optional parameters to influence the location bias. 'zoom' describes the radius around the center to focus on. This is a number that should correspond roughly to the map zoom parameter of a corresponding map. The default is `zoom=16`.

The `location_bias_scale` describes how much the prominence of a result should still be taken into account. Sensible values go from 0.0 (ignore prominence almost completely) to 1.0 (prominence has approximately the same influence). The default is 0.2.

```
1 http://localhost:2322/api?q=berlin&lon=10&lat=52&zoom=12&
  location_bias_scale=0.1
```

Reverse geocode a coordinate

```
1 http://localhost:2322/reverse?lon=10&lat=52
```

Adapt Number of Results

```
1 http://localhost:2322/api?q=berlin&limit=2
```

Adjust Language

```
1 http://localhost:2322/api?q=berlin&lang=it
```

Filter results by bounding box Expected format is minLon,minLat,maxLon,maxLat.

```
1 http://localhost:2322/api?q=berlin&bbox=9.5,51.5,11.5,53.5
```

Filter results by tags and values *Note: the filter only works on principal OSM tags and not all OSM tag/value combinations can be searched. The actual list depends on the import style used for the Nominatim database (e.g. settings/import-full.style). All tag/value combinations with a property 'main' are included in the photon database. If one or many query parameters named `osm_tag` are present, photon will attempt to filter results by those tags. In general, here is the expected format (syntax) for the value of `osm_tag` request parameters.*

1. Include places with tag: `osm_tag=key:value`
2. Exclude places with tag: `osm_tag=!key:value`
3. Include places with tag key: `osm_tag=key`
4. Include places with tag value: `osm_tag=:value`
5. Exclude places with tag key: `osm_tag=!key`
6. Exclude places with tag value: `osm_tag=:!value`

For example, to search for all places named `berlin` with tag of `tourism=museum`, one should construct url as follows:

```
1 http://localhost:2322/api?q=berlin&osm_tag=tourism:museum
```

Or, just by the key

```
1 http://localhost:2322/api?q=berlin&osm_tag=tourism
```

You can also use this feature for reverse geocoding. Want to see the 5 pharmacies closest to a location ?

```
1 http://localhost:2322/reverse?lon=10&lat=52&osm_tag=amenity:pharmacy&limit=5
```

Filter results by layer List of available layers:

- house
- street
- locality
- district
- city
- county
- state
- country

```
1 http://localhost:2322/api?q=berlin&layer=city&layer=locality
```

Example above will return both cities and localities.

Results as GeoJSON

```
1 {
2   "features": [
3     {
4       "properties": {
5         "name": "Berlin",
6         "state": "Berlin",
7         "country": "Germany",
8         "countrycode": "DE",
9         "osm_key": "place",
10        "osm_value": "city",
11        "osm_type": "N",
12        "osm_id": 240109189
13      },
14      "type": "Feature",
15      "geometry": {
16        "type": "Point",
17        "coordinates": [13.3888599, 52.5170365]
18      }
19    },
20    {
21      "properties": {
22        "name": "Berlin Olympic Stadium",
23        "street": "Olympischer Platz",
24        "housenumber": "3",
25        "postcode": "14053",
26        "state": "Berlin",
27        "country": "Germany",
28        "countrycode": "DE",
29        "osm_key": "leisure",
30        "osm_value": "stadium",
31        "osm_type": "W",
32        "osm_id": 38862723,
```

```
33     "extent": [13.23727, 52.5157151, 13.241757, 52.5135972]
34 },
35 "type": "Feature",
36 "geometry": {
37     "type": "Point",
38     "coordinates": [13.239514674078611, 52.51467945]
39 }
40 }
41 ]
42 }
```

Related Projects

- photon's search configuration was developed with a specific test framework. It is written in Python and hosted separately.
- R package to access photon's public API with R
- PHP Geocoder provider to access photon's public API with PHP using the GeoCoder Package