l10n-server Documentation

Release 0.1

Andrey Kalinovsky

Nov 15, 2017

Contents

CHAPTER 1

Introduction

This server's purpose is to interrogate an ElasticSearch index and a MongoDB database, and to return geolocation-related data. It relies on data found on http://www.geonames.org/. You can use this server to retrieve the approximative location of an IPV4 address, to get more details about a city identified by its geonameid, or to find the closest (or biggest) cities matching a given criteria.

The server accepts only HTTP GET requests, and returns appropriatly filled XML or JSON trees.

This server was created using Express.

It supports JSONP and CORS requests.

CHAPTER 2

Installation

2.1 Prerequisites

In order to make the Geonames Server run, you need to have installed MongoDB, ElasticSearch, PHP and NodeJS + NPM.

MongoDB

To install MongoDB, you should follow the official MongoDB guides.

ElasticSearch

If you use homebrew, you can run **brew install elasticsearch** in order to install ElasticSearch. Otherwise, follow the official guide.

NodeJS & npm

To install NodeJS, if you use **homebrew**, a simple **brew install node** is enough. Otherwise, you can download it from the Node.js official website.

To ensure the proper functioning of these operations, curl and mongo extensions for PHP are required.

See mongo extension install details here

Finally, if you want the geolocation to work, you will need the **libgeoip C library**, version **1.4.8** or higher. You can either install it through a package manager (such as **homebrew** or **aptitude**), or build it using the following commands (source):

```
wget http://geolite.maxmind.com/download/geoip/api/c/GeoIP-1.4.8.tar.gz
tar -xvzf GeoIP-1.4.8.tar.gz
cd GeoIP-1.4.8
./configure --prefix=/usr
make
sudo make install
```

2.2 Configuration

Once all required dependencies are installed, you must set the configuration files located in the **config** folder.

First copy the configuration sample files and fill appropriate values according to your system setup.

```
cp ./config/elasticsearch.cfg.sample ./config/elasticsearch.cfg
cp ./config/mongo.cfg.sample ./config/mongo.cfg
cp ./config/server.json.sample ./config/server.json
```

** ElasticSearch configuration**

```
elastic_host="127.0.0.1"
elastic_port="9200"
elastic_scheme="http"
elastic_index="geonames"
elastic_index_test="tests"
```

** MongoDB configuration**

```
mongo_host="127.0.0.1"
mongo_port="27017"
mongo_user=""
mongo_pass=""
mongo_database="geonames"
mongo_database_test="tests"
```

** Server configuration**

```
{
    "app": {
        "verbose": false,
        "port": 3000,
        "allowed_domains": ["*"],
        "max_result_per_page" : 30
    },
    "geo": {
        "geolitepath": "./resources/data/GeoLiteCity.dat"
    }
}
```

2.3 Setup

Once dependencies are installed, you need to fill the MongoDB database with geonames data, and then index this data with ElasticSearch.

To do so, make sure MongoDB and ElasticSearch are running then run the following command within the **root** folder:

```
make install
```

It will download in **resources** folder the necessary files from the geonames servers, format them to make them work with MongoDB, import them to MongoDB, and index the new entries in ElasticSearch.

Note: The installation process takes at least one hour.

From now on, you should be able to access to your ElasticSearch index through your web browser or through any request-forming tool (such as **curl**), as described here.

For instance, you can try:

```
curl -X GET "$elastic_host/$cluster_name/cities/_count"
```

This should return you a JSON object containing, under the "count" field, the number of entries indexed under your cluster.

2.4 Geonames Server

CHAPTER 3

Usage

To start the server, make sure you have **node** installed, and run:

```
node server
```

Then, you can send GET requests to it (through a web browser or any request tool such as **curl**).

3.1 Accepted content types

GeonamesServer can return data formated in two types, **json** or **xml**, according to the type specified within the header request (see http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html). The server supports qvalue ratings, choosing the return type by its rating. If * is specified, data will be returned as a **json** document. If neither **xml**, **json** nor * are specified, the server will answer with a *406 Not acceptable* error.

3.2 CORS Requests

All responses from the GeonamesServer include required headers to be CORS compliant

3.3 JSONP Requests

The GeonamesServer supports *JSONP <http://en.wikipedia.org/wiki/JSONP>* requests. The callback parameter is *callback*.

Request

/ip?ip=4.23.171.0&callback=myFunction

Response

```
myFunction(    {
    "geonames": {
          "ip": "4.23.171.0",
          "geoname": {
                "city": "New York",
                "country_code": "US",
                "country": "United States",
                "longitude": "-73.98",
                "latitude": "40.75",
                "fips": "New York"
          }
    }
})
```

3.4 Available routes

The following is a list of available routes. All these routes can only be accessed through GET requests.

Any other methods will result in a *405 Method not allowed* error.

The results will be sent as text/xml or application/json files, according to the accept field within the request header.

If the *sort* parameter is specified and set to *closeness* but the location of the request could not be determined (due to a lack of data within the GEOIP database or GEOIP module not being installed), the current sort will be replaced by the *population* sort.

If a mandatory parameter or unknown parameter value is detected the server will respond with a *400 Bad Request* error.

3.4.1 /

Returns a quick documentation in HTML format.

3.4.2 /city

Gets the list of all the cities in the database, limited to *max_result_per_page* results by default.

max_result_per_page is setted from configuration files or overridden with limit query parameter parameter up to 100.

Parameters

- **sort** (optional, string, default value : population) available values :
  - population : The results will be sorted by population.
  - closeness : The results will be sorted by closeness to the place the request was sent from.
- **ord** (optional, string, default value : desc) available values :
  - desc : The results will be displayed in descending order.
  - asc : The results will be displayed in ascending order.
- **name** (optional, string) : Filters city whose begins with a given name.

- **country** (optional, string) :

  Only cities located in countries whose name begins with this parameter will be returned.

- **limit** (optional, string) : The number of results.
- **sortParams (optional, array)** [This parameter is used within the *closeness*] sort parameter to provide a custom IP. */city?sort=closeness&sortParams[ip]=XXXX.XX.XX.XXXX*

Examples

Returns the list of all the cities in the database, limited by default.

```
curl -XGET "$SERVER_URL/city"
```

will return one of these results, according to the expected content-type:

```
<?xml version="1.0" encoding="UTF-8"?>
<geonames>
    <totalResultsCount>30</totalResultsCount>
    <geoname>
        <geonameid>1796236</geonameid>
        <title>Shanghai</title>
        <title_alt>Shanghai</title_alt>
        <title_match>Shanghai</title_match>
        <country>China</country>
        <country_match>China</country_match>
        <population>14608512</population>
        <latitude>31.22</latitude>
        <longitude>121.46</longitude>
        <region>Shanghai Shi</region>
    </geoname>
    <geoname>
        <geonameid>3435910</geonameid>
        <title>Buenos Aires</title>
        <title_alt>Buenos Aires</title_alt>
        <title_match>Buenos Aires</title_match>
        <country>Argentina</country>
        <country_match>Argentina</country_match>
        <population>13076300</population>
        <latitude>-34.61</latitude>
        <longitude>-58.38</longitude>
        <region>Buenos Aires F.D.</region>
    </geoname>
    <geoname>
        <geonameid>1275339</geonameid>
        <title>Mumbai</title>
        <title_alt>Mumbai</title_alt>
        <title_match>Mumbai</title_match>
        <country>India</country>
        <country_match>India</country_match>
        <population>12691836</population>
        <latitude>19.07</latitude>
        <longitude>72.88</longitude>
        <region>Mahārāshtra</region>
    </geoname>
</geonames>
```

```
{
  "geonames": {
    "totalResultsCount": "30",
    "geoname": [
      {
        "geonameid": "1796236",
        "title": "Shanghai",
        "country": "China",
        "match": {
          "title": "Shanghai",
          "country": "China"
        },
        "population": "14608512",
        "latitude": "31.22",
        "longitude": "121.46",
        "names": [
          "shanghai",
          "sha",
          "san'nkae",
          "sanchajus",
          "sangaj",
          "sangay",
          "sanghaj",
          "sanghay",
          "sanhaja",
          "sanhajo",
          "sanxay",
          "schanghai",
          "shang-hai",
          "shang-hai-shih",
          "shangai",
          "shangaj",
          "shanghai",
        ],
        "region": "Shanghai Shi",
        "title_alt": "Shanghai"
      },
      {
        "geonameid": "3435910",
        "title": "Buenos Aires",
        "country": "Argentina",
        "match": {
          "title": "Buenos Aires",
          "country": "Argentina"
        },
        "population": "13076300",
        "latitude": "-34.61",
        "longitude": "-58.38",
        "names": [
          "buenos aires",
          "bue",
          "baires",
          "bonaero",
          "bonaeropolis",
          "bonaëropolis",
          "bos aires",
          "bouenos aires",
          "bouonezar",
```

```
          "bouonézâr",
          "buehnos ajres",
          "buehnos-ajres",
          "buehnos-ajres osh",
          "buenos aires",
        ],
        "region": "Buenos Aires F.D.",
        "title_alt": "Buenos Aires"
      },
      {
        "geonameid": "1275339",
        "title": "Mumbai",
        "country": "India",
        "match": {
          "title": "Mumbai",
          "country": "India"
        },
        "population": "12691836",
        "latitude": "19.07",
        "longitude": "72.88",
        "names": [
          "mumbai",
          "asumumbay",
          "bom",
          "bombai",
          "bombaim",
          "bombaj",
          "bombay",
          "bombaya",
          "bombej",
          "bombejus",
          "bombėjus",
          "bumbaj",
          "bůmbaj",
        ],
        "region": "Mahārāshtra",
        "title_alt": "Mumbai"
    }
}
```


Gets the city whose name begins with the provided query.

```
curl -XGET "$SERVER_URL/city?name=paris"
```

will return one of these results, according to the expected content-type:

```
<?xml version="1.0" encoding="UTF-8"?>
<geonames>
    <totalResultsCount>30</totalResultsCount>
    <geoname>
      <geonameid>2988507</geonameid>
      <title>Paris</title>
      <title_alt>paris</title_alt>
      <title_match>Paris</title_match>
      <country>France</country>
      <country_match>France</country_match>
      <population>2138551</population>
      <latitude>48.85</latitude>
```

```
      <longitude>2.35</longitude>
      <region>Île-de-France</region>
    </geoname>
    <geoname>
      <geonameid>4717560</geonameid>
      <title>Paris</title>
      <title_alt>paris</title_alt>
      <title_match>Paris</title_match>
      <country>United States</country>
      <country_match>United States</country_match>
      <population>25171</population>
      <latitude>33.66</latitude>
      <longitude>-95.56</longitude>
      <region>Texas</region>
    </geoname>
    <geoname>
      <geonameid>3023645</geonameid>
      <title>Cormeilles-en-Parisis</title>
      <title_alt>cormeilles-en-parisis</title_alt>
      <title_match>Cormeilles-en-Parisis</title_match>
      <country>France</country>
      <country_match>France</country_match>
      <population>21973</population>
      <latitude>48.97</latitude>
      <longitude>2.2</longitude>
      <region>Île-de-France</region>
    </geoname>
        ...
</geonames>
```

```
{
    "geonames": {
      "totalResultsCount": "30",
      "geoname": [
        {
          "geonameid": "2988507",
          "title": "Paris",
          "country": "France",
          "match": {
            "title": "Paris",
            "country": "France"
          },
          "population": "2138551",
          "latitude": "48.85",
          "longitude": "2.35",
          "names": [
            "paris",
            "baariis",
            "bahliz",
            "gorad paryzh",
            "lungsod ng paris",
            "lutece",
            "lutetia",
            "lutetia parisorum",
            "par",
            "pa-ri",
            "paarys",
            "palika",
```

```
            "paname",
            "pantruche",
            "paraeis",
            "paras",
            "pari",
            "paries",
            "parigge",
            "pariggi",
            "parighji",
            "parigi",
            "pariis",
            "pariisi",
            "parij",
            "parijs",
            "paris",
            "parisi",
            "parixe",
            "pariz",
          ],
          "region": "Île-de-France",
          "title_alt": "paris"
        },
        {
          "geonameid": "4717560",
          "title": "Paris",
          "country": "United States",
          "match": {
            "title": "Paris",
            "country": "United States"
          },
          "population": "25171",
          "latitude": "33.66",
          "longitude": "-95.56",
          "names": [
            "paris",
            "prx",
            "parizh",
            ""
          ],
          "region": "Texas",
          "title_alt": "paris"
        },
        {
          "geonameid": "3023645",
          "title": "Cormeilles-en-Parisis",
          "country": "France",
          "match": {
            "title": "Cormeilles-en-Parisis",
            "country": "France"
          },
          "population": "21973",
          "latitude": "48.97",
          "longitude": "2.2",
          "names": [
            "cormeilles-en-parisis",
            "cormeilles",
            "cormeilles-en-parisis"
          ],
```

```
            "region": "Île-de-France",
            "title_alt": "cormeilles-en-parisis"
    }
}
```

The *title_match* and *country_match* fields show the parts of the initial request that match with the results. This might be used for highlighting the beginning of the world as the user types it in.

In cases where the request does not match with the default name of the city but does match with an alternate name (different language or different spelling), a *title_alt* field is displayed, so the *title_match* can still be relevant.

Returns the city in which the given IP address is located.

```
curl -XGET "$SERVER_URL/ip/ip=4.23.171.0"
```

will return one of these results, according to the expected content-type:

```
<?xml version="1.0" encoding="UTF-8"?>
<geonames>
     <ip>4.23.171.0</ip>
     <geoname>
       <city>New York</city>
       <country_code>US</country_code>
       <country>United States</country>
       <fips>New York</fips>
       <longitude>-73.98</longitude>
       <latitude>40.75</latitude>
     </geoname>
 </geonames>
```

```
{
    "geonames": {
          "ip": "4.23.171.0",
          "geoname": {
                "city": "New York",
                "country_code": "US",
                "country": "United States",
                "longitude": "-73.98",
                "latitude": "40.75",
                "fips": "New York"
          }
    }
}
```

3.4.3 /city/{id}

Returns the city which *geonameid* value is equal to the given id.

```
curl -XGET "$SERVER_URL/city/3435910"
```

will return one of these results, according to the expected content-type:

```
<?xml version="1.0" encoding="UTF-8"?>
<geonames>
    <totalResultsCount>1</totalResultsCount>
    <geoname>
```

```
      <geonameid>3435910</geonameid>
      <title>Buenos Aires</title>
      <title_alt>Buenos Aires</title_alt>
      <title_match>Buenos Aires</title_match>
      <country>Argentina</country>
      <country_match>Argentina</country_match>
      <population>13076300</population>
      <latitude>-34.61</latitude>
      <longitude>-58.38</longitude>
      <region>Buenos Aires F.D.</region>
    </geoname>
</geonames>
```


```
{
    "geonames": {
      "totalResultsCount": "1",
      "geoname": [
        {
          "geonameid": "3435910",
          "title": "Buenos Aires",
          "country": "Argentina",
          "match": {
            "title": "Buenos Aires",
            "country": "Argentina"
          },
          "population": "13076300",
          "latitude": "-34.61",
          "longitude": "-58.38",
          "names": [
            "buenos aires",
            "bue",
            "baires",
            "bonaero",
            "bonaeropolis",
            "bonaëropolis",
            "bos aires",
            "bouenos aires",
            "bouonezar",
            "bouonézâr",
            "buehnos ajres",
            "buehnos-ajres",
            "buehnos-ajres osh",
            "buenos aires",
            "buenos airés",
            "buenos ajres",
            "buenos ayres",
            "buenos-aires",
            "buenos-ajres",
            "buenos-ayres",
            "buenos-aýres",
            "buenosairesa",
            "buénos ayrés",
            "bwenoze",
            "bwènozè",
          ],
          "region": "Buenos Aires F.D.",
          "title_alt": "Buenos Aires"
        }
```

```
        ]
    }
}
```

CHAPTER 4

Testing

This server relies on Mocha and Supertest for unit testing. All you have to do is to run the following command in the root folder:

```
make test
```

CHAPTER 5

Recipes

CHAPTER 6

Contribute

You found a bug and resolved it ? You added a feature you want to share ? You optimized the code or made it more aesthetically pleasing ? You found a typo in this doc and fixed it ? Feel free to send a Pull Request on GitHub, we will be glad to merge your code.