GeoTriples

Publishing geospatial data as Linked Open Geospatial Data. GeoTriples generates and processes extended R2RML and RML mappings that transform geospatial data from many input formats into RDF. GeoTriples allows the transformation of geospatial data stored in raw files (shapefiles, CSV, KML, XML, GML and GeoJSON) and spatially-enabled RDBMS (PostGIS and MonetDB) into RDF graphs using well-known vocabularies like GeoSPARQL and stSPARQL, but without being tightly coupled to a specific vocabulary.

Quickstart

Use GeoTriples binaries (Unix)

Assuming Java >=7 installed:

Download GeoTriples binaries here

- Unzip the downloaded file `geotriples-<version>-bin.zip`
- Change directory to `geotriples-<version>-bin`
- Under the `bin` directory you can find the available starter script for GeoTriples

  > `bin/geotriples-cmd`

See more at Wiki pages

Execution by source

Install the source code by using

```
mvn package
```

Generate Mapping files:

```
java -cp <geotriples-core/ dependencies jar> eu.linkedeodata.geotriples.GeoTriplesCMD
generate_mapping -o <output file(.ttl)> -b <URI base> <input file>
```

- **-o output_file** the name of the produced mapping file (RML/R2RML)
- **-b URI_base** the base URI that will describe the entities

Transform file into RDF

```
java -cp <geotriples-core/ dependencies jar> eu.linkedeodata.geotriples.GeoTriplesCMD
dump_rdf -o <output file> -b <URI base> (-sh <shp file>) <(produced) mapping file
(.ttl)>
```

- **-o output_file** the path of the produced file
- **-b URI_base** the base URI that will describe the entities
- **-sh shp_file** if the input is a shapefile specify the .shp path using this flag

Execution by binaries

Generate Mapping files:

```
bin/geotriples-all generate_mapping -o <output_file (.ttl)> -b <URI base>
(-sh <shp file>) <(produced) mapping file (.ttl)>
```

Transform file into RDF

```
bin/geotriples-all dump_rdf -o <output_file> -b http://example.com
(-sh <shp file>) <path_to_the_mapping_file>
```

GeoTriples-Spark

GeoTriples-Spark is an extended version of GeoTriples capable of transforming big geospatial data into RDF graphs. To enable the transformation of big geospatial data, we extended GeoTriples to run on top of Apache Spark and Hadoop or Hops (a new distribution of Apache Hadoop developed by KTH, RISE SICS, and Logical Clocks AB). GeoTriples-Spark can run in a standalone machine or in a Hadoop based cluster, but it is more efficient when it runs on Hops as it is a write-intensive application.

Requirements

- Java 8
- Maven 3
- Apache Spark 2.4.0 or greater
- Apache Hadoop 2.7.0 or Hops

Build

```
mvn package
```

Execute

```
spark-submit --class eu.linkedeodata.geotriples.GeoTriplesCMD
<geotriples-core/ dependencies jar> spark -i <in_file> -o <out_folder> <rml>
```

- **-i input_file**: path to input dataset. To enter multiple files separate them using comma ",".

- **-o out_folder**: path to folder where results will be stored. The folder must not exist but in case it does, a new folder inside of it will be created.
- The **rml** file is the file generated by the "generate_mapping" procedure of GeoTriples.

Additional flags

- **-m mode**: set transformation mode. It can be either "partition" or "row"(default mode). In "partition" mode the RDF triples are written to the targeted file after the completion of transformation of the whole partition. In the "row" mode, each line is transformed into RDF triples and are directly written to the files. For small datasets the "partition" mode is faster, but otherwise we advise to use the "row" mode as it is more memory friendly.
- **-r partitions**: Using -r flag you can re-partition the input dataset. **WARNING** re-partitionig triggers data shuffling and therefore it can significantly increase the execution time.
- **-sh folder_path**: It is used in order to load a multiples ESRI shapefiles (each one must be stored in a separate folder) that exist in the "folder_path". For example the structure of the folder must look like

  ```
   folder_path/shapefile1/shapefile1.(shp, dbf, shx, etc)
  folder_path/shapefile2/shapefile2.(shp, dbf, shx, etc)
  ...
  ```

  For each Shapefile, a distinct RDF dataset will be created. Furthermore, it's important to mention that the RML mapping file must support all the input datasets.
- **-times n**: Load the input dataset "n" times.
- **help**: Print help