LIBRE Documentation

Release 1.1

Roberto Rosario

December 13, 2013

Contents

Free, Open source, Django based FOSS data restructuring, versioning and exporting tool

LIBRE was created by the Office of the Chief Information Officer of the Commonwealth of Puerto Rico to power the liberation of government data. It has been made available as Free software with the hopes that other countries and individuals may benefit from it too.

CHAPTER 1

On the Web

- Source: https://github.com/commonwealth-of-puerto-rico/libre
- Video: http://www.youtube.com/watch?v=pbY_dKlniHM
- Documentation: https://libre.readthedocs.org/en/latest/
- PyPI: https://pypi.python.org/pypi/libre

Looking for specific information? Try the `detailed table of contents` otherwise below are the different part of the documentation.

CHAPTER 2

User Guide

2.1 Installation

2.1.1 OS dependencies

LIBRE supports Spatial queries as such is dependant on several libraries that are installed at the OS level.

If using Ubuntu Linux install the required libraries with:

```
$ sudo apt-get install libgdal-dev -y
```

On OSX using MacPorts:

```
$ sudo port install geos
$ sudo port install gdal
```

Proceed to install the actual files of **LIBRE**:

2.1.2 Using pip

Via pip Python packager installer

```
$ pip install libre
$ libre-admin.py syncdb --migrate
$ cat <<'EOF' > settings_local.py
DEBUG=True
DEVELOPMENT=True
EOF
$ libre-admin.py runserver --pythonpath=.
```

2.1.3 From GitHub

By cloning the code from the GitHub repository:

```
$ git clone https://github.com/commonwealth-of-puerto-rico/libre.git
$ cd libre
$ virtualenv venv
```

```
$ source venv/bin/activate
$ pip install -r libre/requirements.txt
$ ./manage.py syncdb --migrate
$ cat <<'EOF' > settings_local.py
DEBUG=True
DEVELOPMENT=True
EOF
$ ./manage.py runserver
```

2.1.4 Docker container

Or by using Yamir Encarnacion's Docker container:

Use this to build a new image, tagged for easier reuse

```
$ sudo docker build -t yencarnacion/libre-docker github.com/yencarnacion/libre-docker
```

Running the container

```
$ sudo docker run -d -p 8000:8000 yencarnacion/libre-docker
```

The default username and password for the Docker image are: Username: **admin** | Password: **libre**

Once up and running go to *<your ip>:8000* in your browser to use **LIBRE**.

2.2 Release History

2.2.1 1.1.0 (2013-12-13)

- New frontend for non technical users, dataset browser, dataset showcase
- Support for boolean values to LQL
- Support for clustering map features
- Fix handling of dates as key when using _as_dict_list
- Increased required version of Fiona to 1.0.2
- Updated Leaflet version used to 0.7
- Added boolean values support to LQL
- Added Leaflet marker clustering plugin support
- Optimize Leaflet's marker's use by encode markers as base64 PNG images and embedding them in the renderer's HTML output
- Menu reorganization and cleanup
- Add support to add an image to a source dataset
- Documentation updates
- Update required version of djangorestframework
- Origins module now copies local files in chunks and streams remote HTTP files improving memory usage during imports

2.2.2 1.0.0 (2013-11-19)

- Accepted: Added Command Line Interface (CLI) for update_admin_user (#10)
- Accepted: Added Pre-Installation Steps necessary to run on OSX (#9)
- Closed: Added missing docutils requirement (#8)
- Closed: Missing dependency on requirements (#4)
- Closed: inotify is not available on macosx-10.8-intel (#2)
- Accepted: Add slugify method for automatic slugs (#1)
- Fix CSV source issues with CSV file encodings (utf-8, iso-8859-1) by allowing users to specify the file encoding.
- Increased required version to Django to 1.5.5
- Add scheduling support to Sources
- Reduce the data source origin data check resolution to 45 seconds
- Fail gracefully when GIS features have no bounds
- Add new PythonScript origin

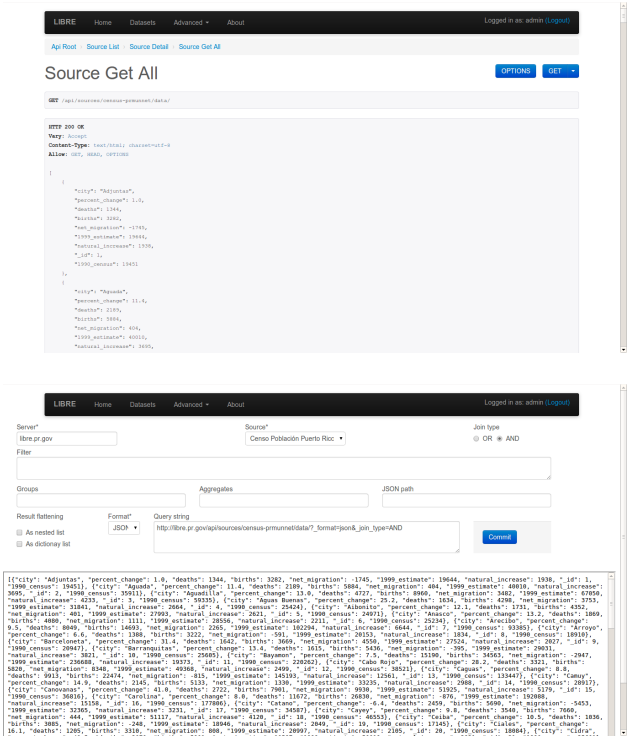
2.3 Features

2.3.1 Dataset browser

The dataset browser provides a simple detail view of each dataset properties and metadata.

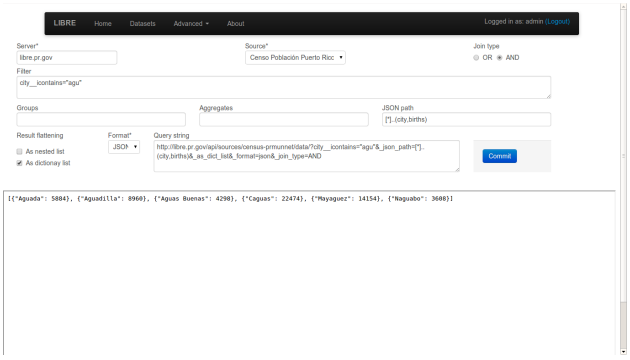
2.3.2 Browseable API

The browseable API renderer comes straight from Django REST framework but has been integrated in look and functionality. Allows for simple exploration of datasets without requiring any documentation.



2.3.3 Query builder

The query builder allows developers and users to create queries with an easy to use interface.



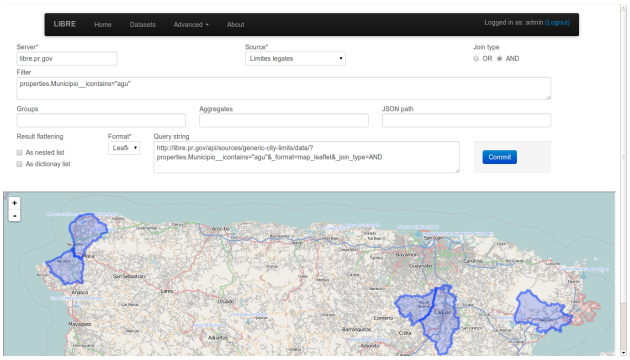
Query builder also allows for quick test of queries, returning the final LQL query to repeat the same resultset outside the query builder.

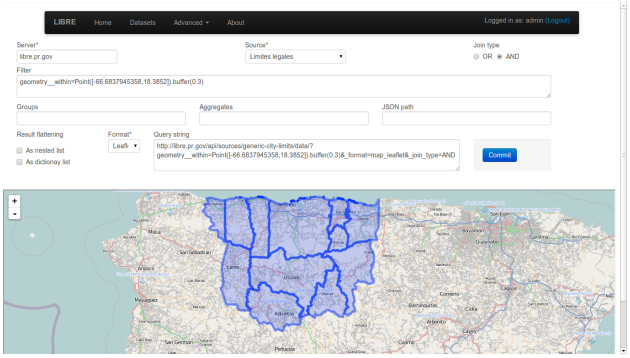
2.3.4 Query engine

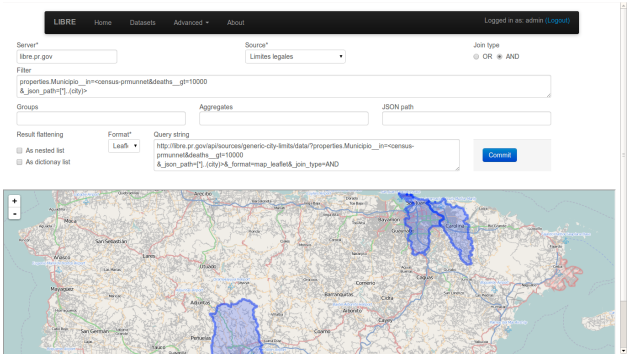
The LIBRE query engine support various types of source data, all of which are queried in the same manner. The engine also include various output renderers, such as this spatial render.

Spatial queries are also part of the LIBRE query language specification. Spatial queries are supported regarless the underlying database manager used for data storaged supports them or not, as they are implemented in the LIBRE query engine itself and not proxied to the database manager.

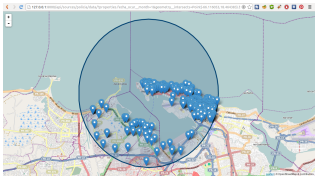
The LIBRE query engine also supports heterogeneous subqueries, where the results of a dataset can be filtered by the

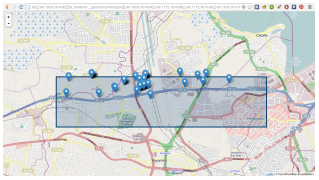

LIBRE
Home
Datasets
Advanced
About
Logged in as: admin (Logout)
Server*
libre.pr.gov
Source*
Limites legales
Join type
OR
AND
Filter
properties.Municipio__icontains="agu"
Groups
Aggregates
JSON path
Result flattening
As nested list
As dictionary list
Format*
Leaflet
Query string
http://libre.pr.gov/api/sources/generic-city-limits/data/?properties.Municipio__icontains="agu"&_format=map_leaflet&_join_type=AND
Commit


LIBRE
Home
Datasets
Advanced
About
Logged in as: admin (Logout)
Server*
libre.pr.gov
Source*
Limites legales
Join type
OR
AND
Filter
geometry__within=Point([-66.683794535,18.3852]).buffer(0.3)
Groups
Aggregates
JSON path
Result flattening
As nested list
As dictionary list
Format*
Leaflet
Query string
http://libre.pr.gov/api/sources/generic-city-limits/data/?geometry__within=Point([-66.683794535,18.3852]).buffer(0.3)&_format=map_leaflet&_join_type=AND
Commit


LIBRE
Home
Datasets
Advanced
About
Logged in as: admin (Logout)
Server*
libre.pr.gov
Source*
Limites legales
Join type
OR
AND
Filter
properties.Municipio__in=<census-prmunnet&deaths__gt=10000
&_json_path=[*].(city)=
Groups
Aggregates
JSON path
Result flattening
As nested list
As dictionary list
Format*
Leaflet
Query string
http://libre.pr.gov/api/sources/generic-city-limits/data/?properties.Municipio__in=<census-prmunnet&deaths__gt=10000
&_json_path=[*].(city)=&_format=map_leaflet&_join_type=AND
Commit

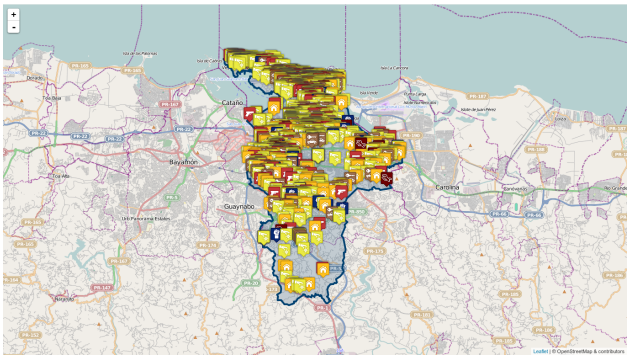
results of a query applied to a different dataset of a completely different data type. In this example shapefile features are being filtered based on the mortality rate that come from a fixed width column dataset.





Because the LIBRE Query Language was created from the start to be a RESTful query language and not depend on a specific database manager client software, complex geometries can be specified straight from the browser URL and used for geo fencing the results.

2.3.5 Performance



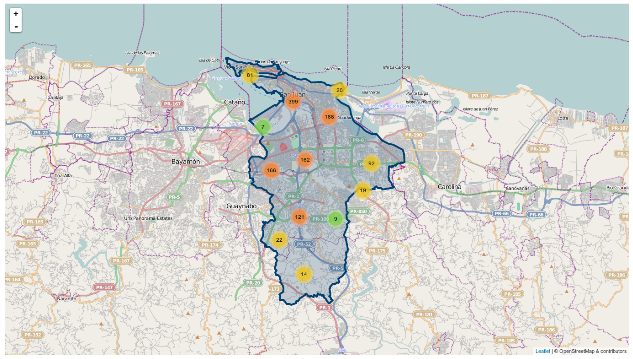
Even at the initial stages of development, the LIBRE query engine performance is very good, being able to do complex spatial filtering based on spatial subquering, rendering using geo fencing, custom markers and map geo fencing indicators in barely a few seconds. This example shows a crime map with city poliyon based geo fencing and custom crime markers with incident information popups. Many perfomance enhancements are already planned and can be found in the development section of the documentation.

2.3.6 Renderers

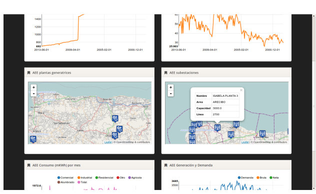
Aside from supporting multiple input data types, the LIBRE query engine also supports mutiple output renderers each itself with several plugins and specific options. This is the same crime map rendered with the marker cluster plugin enabled.

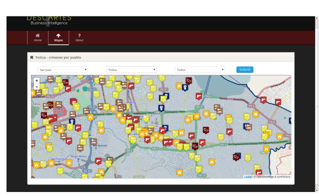
Exactly the same crime data in XML format.

Again the same crime data in Geo JSON format.




This XML file does not appear to have any style information associated with it. The document tree is shown below.
<root>
<list-item>
<geometry>
<type>Point</type>
<coordinates>
<list-item>-66.0955868941</list-item>
<list-item>18.338481303</list-item>
</coordinates>
</geometry>
<_id>50</_id>
<id>54</id>
<properties>
<date>2013-05-16</date>
<crime_type_id>5</crime_type_id>
<time>18:01:00</time>
</properties>
</list-item>
<list-item>
<geometry>
<type>Point</type>
<coordinates>
<list-item>-66.0955868941</list-item>
<list-item>18.338481303</list-item>
</coordinates>
</geometry>
<_id>3950</_id>
<id>3954</id>
<properties>
<date>2013-05-15</date>
<crime_type_id>3</crime_type_id>
<time>03:40:00</time>
</properties>
</list-item>
<list-item>
<geometry>
<type>Point</type>
<coordinates>
<list-item>-66.0955868941</list-item>
<list-item>18.338481303</list-item>
</coordinates>
</geometry>
<_id>4020</_id>
<id>4024</id>
<properties>
<date>2013-05-09</date>
<crime_type_id>6</crime_type_id>
<time>23:58:00</time>
</properties>





2.3.7 Integration

Integration was a design goal from day 0, as such LIBRE's output is meant to be easily captured for integration into other software, such a business intelligence software. This design philosophy allows developers to add many of LIBRE features to their software without writting a single line of code.

CHAPTER 3

API Documentation

If you are looking for information on a specific filter or function to use the data on **LIBRE** this part of the documentation is for you.

3.1 LQL LIBRE Query Language

Version 1.1 of the **LIBRE** Query Language specification. LQL is a mixture of SQL, Django's ORM, Python's syntax and geospatial queries constructs using the URL query strings to create a RESTful query language.

3.1.1 Changelog

- 2013-12-11 Added support for boolean values
- 2013-12-11 Bump version to 1.1

3.1.2 Values

LQL accepts as input:

- numbers - Any value not enclosed in double quotes.
- boolean - Any of the following two values, not enclosed in double quotes: `True` or `False`.
- strings - Any value enclosed in double quotes.
- lists - Any value enclosed with brackets.
- geometries - Any value enclosed in the geometry specifier `Point(coordinates)`, `LineStrings(coordinates)`, `LinearRings(coordinates)`, `Polygon(exterior[, interiors=None])`, `MultiPoint(points)`, `MultiLineString(lines)`, `MultiPolygon(polygons)` or `Geometry(GeoJSON)`.
- dates - Any value enclosed with the `Date` specifier.
- time - Any value enclosed with the `Time` specifier.
- date & time - Any value enclosed with the `DateTime` specifier.
- subqueries - Any string enclosed with the less than (<) and more than (>) simbols.

The only exception to this convention are special query directive values, such as those of the **join** directive, which are specified unquoted. Geospacial geometries also have special attributes which can be accesed and used for filtering, these are: _length, _area and _type

Examples:

A string: `"hello word"`

A number: `42`

A list: `['hello', 'world']` or `[1, 2, 3]`

A geometry: `Point(longitude, laitude)`

A date: `Date(2013-01-01)`

A time: `Time(10:00pm)` or `Time(22:00)`

A date and time: `DateTime(2013-01-01 1:00pm)`

A subquery: `births=<census-prmunnet&_aggregate__aggregated_most_births=Max(births)&_json_path=`

A boolean: `_format=map_leaflet&_join_type=AND&_renderer__enable_clustering=True`

3.1.3 Filtering

To filter a collection by a field, specify the field name appending a double underscore '__' (or the specified delimiter if overrided) appending again one of the following filters. Multiple filters can be specified on a single query.

Strings filters

contains

`contains=<string>`

Return the elements whose field values includes the specified string.

Example: `first_name__contains="John"`

icontains

`icontains=<string>`

Return the elements whose field values includes the specified string. Matches upper and lower cases.

Example: `last_name__icontains="smith"`

startswith

`startswith=<string>`

Return the elements whose field values start with the specified string.

Example: `state__startswith="North"`

istartswith

`istartswith=<string>`

Return the elements whose field values start with the specified string. Matches upper and lower cases.

Example: `city__istartswith="John"`

endswith

`endswith=<string>`

Return the elements whose field values end with the specified string.

Example: `state__startswith="Carolina"`

iendswith

`iendswith=<string>`

Return the elements whose field values end with the specified string. Matches upper and lower cases.

Example: `company_name__iendswith="corp"`

iequals

`iequals=<string>`

Return the elements whose field values match the specified string, matches upper and lower cases.

Example: `full_name__iequals="john carter"`

Number filters

lt

`lt=<number>`

Return the elements whose field values are less than the specified number.

Example: `ytd_sales__lt=1000000`

lte

`lte=<number>`

Return the elements whose field values are less than or equal than the specified number.

Example: `employees_count__lte=1000`

gt

`gt=<number>`

Return the elements whose field values are greater than the specified number.

Example: `spare_rooms__gt=3`

gte

`gte=<number>`

Return the elements whose field values are greater than or equal than the specified number.

Example: `month_sales__gte=200000`

Spatial filters

has

`has=<geometry>`

Return the elements whose interior geometry contains the boundary and interior of the geometry specified, and their boundaries do not touch at all.

Example: `city__has=Point(-66.16918303705927,18.40250894588894)`

disjoint

`disjoint=<geometry>`

Return the elements whose boundary and interior geometry do not intersect at all with the geometry specified.

Example: `country__disjoint=Point(-66.16918303705927,18.40250894588894)`

intersects

`intersects=<geometry>`

Return the elements whose boundary and interior geometry intersects the geometry specified in any way.

Example: `county__intersects=Point(-66.16918303705927,18.40250894588894).buffer(0.5)`

touches

`touches=<geometry>`

Return the elements who have at least one point in common with and whose interiors do not intersect with the geometry specified.

Example: `river__touches=LineString([-66.16918303705927,18.40250894588894])`

within

`within=<geometry>`

boundary and interior intersect only with the interior of the other (not its boundary or exterior).

Return the elements whose boundary and interior intersect only with the interior of the specified geometry (not its boundary or exterior).

Example: `crime__within=Polygon([[-66.16918303705927,18.402508945888894]])`

Other filters

in

`in=<list of strings or numbers>`

Return the elements whose field values match one entry in the specified list of strings or numbers.

Example: `crime_type_id__in=[1,4,8]`

range

`range=<list of two dates, two times, two date and times, two numbers or two strings>`

Return the elements whose field values's months are within the the specified values.

Example: `purchases_date__range=[Date(2013-01-01), Date(2013-03-01)]`

Negation

All filter can be negated by adding `__not` before the filter name, this will cause their logic to be inverted.

Return the elements whose field values do not match one entry in the specified list of strings or numbers.

Example: `city_id__not_in=[41,3,142]`

Directives

All directive are prepended by the underscore delimiter '_'.

join

`_join=<OR | AND>`

When multiple filters are specified per query the results of each filter are `ANDed` by default, this directive changes that behaviour so that results are `ORed` together.

json_path

Reduce the result set using JSON Path

`_json_path=JSON Path syntax`

JSON Path syntax: https://github.com/kennknowles/python-jsonpath-rw

renderer

Pass renderer specific key value pairs. The key and values are dependent on the renderer being used.

Values for the map_leaflet renderer:

- zoom_level
- longitude
- latitude
- geometry
- enable_clustering = <`True` or `False`>; Enable the Leaflet Marker Clustering plugin

Example: `_renderer__zoom_level=13&_renderer__longitude=-66.116079&_renderer__latitude=18.4643`

Aggregation

Aggregates asssist with the summarization of data.

Example: `api/sources/crimes/data/?properties.date__month=2&geometry__intersects=Point(-67,18.`

Return a count of all crimes committed in February and which occurred within the selected geographical area.

Count

Return the count of rows or occurences of a value in the specified list, returned as an alias.

`Count(<field to count> or <*>)`

Example: `_aggregate__total=Count(*)`

Sum

Return the sum of the values of the specified field.

`Sum(<field to sum>)`

Example: `_aggregate__total_score=Sum(score)`

Min

Return the minimum value of the specified field in the elements.

`Min(<field>)`

Example: `_aggregate__least_deaths=Min(deaths)`

Max

Return the maximun value of the specified field in the elements.

`Max(<field>)`

Example: `_aggregate__most_births=Max(births)`

Average

Return the average value of the specified field in the elements.

`Average(<field>)`

Example: `_aggregate__point_average=Average(points)`

Grouping

`_group_by=<comma delimited list of fields by which to group data>`

Example: `_group_by=city,region`

Transformations

_as_dict_list

Return the current values as a list of key value dictionaries

_as_nested_list

Return the current values as a nested list (list of lists)

CHAPTER 4

For developers

If you know your way around *Python*/*Django*/*Git* this part of the documentation is for you.

4.1 Development

LIBRE is under active development, and contributions are welcome.

If you have a feature request, suggestion, or bug reports, please open a new issue on the GitHub issue tracker. To submit patches, please send a pull request on GitHub. Contributors are credited accordingly on the *Authors* section.

4.1.1 Source Control

LIBRE source is controlled with Git

The project is publicly accessible, hosted and can be cloned from **GitHub** using:

```
$ git clone https://github.com/commonwealth-of-puerto-rico/libre.git
```

4.1.2 Git branch structure

LIBRE follows the model layout by Vincent Driessen in his Successful Git Branching Model blog post. Git-flow is a great tool for managing the repository in this way.

`develop` The "next release" branch, likely unstable.

`master` Current production release (1.1).

`feature/` Unfinished/ummerged feature.

Each release is tagged and available for download on the Downloads section of the **LIBRE** repository on GitHub

When submitting patches, please place your feature/change in its own branch prior to opening a pull request on GitHub. To familiarize yourself with the technical details of the project read the *internals* section.

4.1.3 Versioning

LIBRE follows the Semantic Versioning specification.

Summary:

Given a version number `MAJOR.MINOR.PATCH`, increment the:

`MAJOR` version when you make incompatible API changes, `MINOR` version when you add functionality in a backwards-compatible manner, and `PATCH` version when you make backwards-compatible bug fixes. Additional labels for pre-release and build metadata are available as extensions to the `MAJOR.MINOR.PATCH` format.

4.1.4 How To Contribute

LIBRE is always open for suggestions and contributions by developers. Here are a few tips to get you started.

Please:

- Obey PEP 8 and PEP 257.
- *Always* add tests and docs for your code.
- Add yourself to the AUTHORS.rst file in an alphabetical fashion.
- Write good commit messages.
- Ideally, squash your commits, i.e. make your pull requests just one commit.

Thank you for considering to contribute to **LIBRE**!

4.1.5 Debugging

LIBRE makes extensive use of Django's new logging capabilities. To enable debug logging for the `data_drivers` app for example add the following lines to your `settings_local.py` file:

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': True,
    'formatters': {
        'verbose': {
            'format': '%(levelname)s %(asctime)s %(name)s %(process)d %(thread)d %(message)s'
        },
        'intermediate': {
            'format': '%(name)s <%(process)d> [%(levelname)s] "%(funcName)s() %(message)s"'
        },
        'simple': {
            'format': '%(levelname)s %(message)s'
        },
    },
    'handlers': {
        'console':{
            'level':'DEBUG',
            'class':'logging.StreamHandler',
            'formatter': 'intermediate'
        }
    },
    'loggers': {
        'data_drivers': {
            'handlers':['console'],
```

```
            'propagate': True,
            'level':'DEBUG',
        },
    }
}
```

Likewise, to see the debug output of the `origins` app, just add the following inside the `loggers` block:

```
'origins': {
    'handlers':['console'],
    'propagate': True,
    'level':'DEBUG',
},
```

4.2 TO DO List

LIBRE already has an extensive set of functionality but there are things and functionality everybody would like to see added, here is a list of those things.

4.2.1 Database sources

- Add DB Source support
  - Pony ORM

4.2.2 Datastore

- Multiple DataStores support
- File-based DataStore
- DataStore router support
- DjangoStorage DataStore support

4.2.3 Documentation

- Getting started
- Add better sample source files
  - http://www.census.gov/population/estimates/puerto-rico/prmunnet.txt
- Examples using existing public data

4.2.4 Filebased sources

- Add compressed file support
- Skip blank lines?
- Switch from column widths to column ranges
- Toggable auto update via inotify, polling or python-watchdog

- Add internal support for open ranges for rows "10-"
- Migrate Spreadsheet regex import and skip solution to other filebased sources
- Add row number exclusion support during import

4.2.5 General

- Add Relationship support
- Specify number of versions to keep, deleting old ones
- Add instructions to sources, per source type
- Add row number exclusion support during import
- Stored JSON data index support
- JSON source descriptor export and import
- Rename 'timestamp' to 'version' and allow user defined version strings
- Data translation
- Remap JSON names
- Password reset view

4.2.6 Renderes

- Add D3 renderer
- Add Google Maps renderer

4.2.7 Job processing

- Add Celery support or subprocess

4.2.8 LQL

- Views support
- Dataset Namespaces
- Result reformating to allow including metadata in HTTP response
  - { "result": {"a": 1, "b": 2}, "count": 2, "limit": 100, "response_time": "100ms"}
- LQL based pagination (size and page number) (Andres Colón)
- Expand the _fields directive to support dot and index notations
- Sorting
  - _order=<field name>,<field name>
  - sort(+field_name,-field_name)
  - Ascending (field name)
  - Descending (-field name)

- Add range exclusion
  - _xrange, _not_in_range, _nrange
- Add regex support
  - _match
- Annotations
- Limiting
  - _limit=<soft limit of elements>
- Skipping results
  - _skip=<number of elements>
  - _first
  - _last
  - _one
    - Return error if more than one
- Combined
  - _limit=(count, start, maxCount)
- Joins between datasets
  - _join=<data set name>,<join type>,<current set field>__<foreign set field>,<current set field>__<foreign set field>
  - _relation=(field, subquery)
- Field selection
  - _select=(field_name, field_name)
- _distinct
- Not in = out
- _excludes

4.2.9 Output

- Add support for generating output formats other than JSON
  - Shapefiles
  - GeoJSON - DONE
  - CSV
  - Excel
  - XML - DONE
  - NIEM
  - Fixed width

4.2.10 Web services sources

- Add caching support to WS Sources
  - TTL support

4.2.11 Unsorted

- Improve output logging - INPROGRES
- Empty but valid queries should return HTTP404 or HTTP200 with '{"status": "Not found"}'
- Show required argument for WS
- Interpret WS arguments
- Result count
- Fix upload_to
- Calculate geometries area, size, lenghts in pin template
- Delete stored source files when a source is deleted
- Delete stored source files when a new file is uploaded
- Fix JsonField not returning dates or times only datetimes
- Move _fields parsing to allow being parsed on get_one method
- Optimize AND type join
- Use islice
- Dataset human browser
- Data store browser
- Add support for item-based and result-based evaluation
- Add support for JSON Pointer
- Add support for displaying map titles
- Add support for dynamic icons for the map renderer
  - http://tools.ietf.org/html/draft-ietf-appsawg-json-pointer-09
- Add support for RQL
  - http://www.sitepen.com/blog/2010/11/02/resource-query-language-a-query-language-for-the-web-nosql/
  - http://rql-engine.eu01.aws.af.cm/
- Add support for JSON Query
  - http://dojotoolkit.org/reference-guide/1.9/dojox/json/query.html
  - http://www.sitepen.com/blog/2008/07/16/jsonquery-data-querying-beyond-jsonpath/
- Add support for JSONgrep
  - http://blogs.fluidinfo.com/terry/2010/11/25/jsongrep-py-python-for-extracting-pieces-of-json-objects/
- Migrate DatabaseSource's get_one and get_all solution to other source classes
- Get rid of WSResultField WSArgument and use SourceColumnBase instead

- Icon preview in admin
- Add webhooks support
  - https://github.com/johnboxall/django_webhooks
- Regex support for Fixed width sources
- Add view type source
- Improve _flatten predicate
- Add dumb result caching
  - Hash query + hash of sources = key: value = result
- Add custom response header values
  - X-LIBRE-count
  - X-LIBRE-query
  - response = Response(result) response['X-LIBRE-count'] = count return response
- Get rid of fetch_all on the DB backend
  - cursor.rowcount
- Improve sort with Sort generators
  - https://gist.github.com/rbonvall/18896
  - http://www.ics.uci.edu/~eppstein/161/python/mergesort-generators.py
- Dynamic icons
- Add renderer directives text box entry in the query builder
- Add password change views and templates
- Update Twitter bootstrap version used to 3.0

CHAPTER 5

Credits

5.1 Authors

LIBRE is written and maintained by Roberto Rosario and various contributors:

5.1.1 Development Lead

- Roberto Rosario <rrosario@ogp.pr.gov> [http://robertorosario.com]

5.1.2 Patches

- Axel Rivera <github@riveralabs.com> [http://riveralabs.com]
- Yamir Encarnación <info@webninjapr.com> [http://www.webninjapr.com]

5.1.3 Ideas and Suggestions

- Andres Colón <levipr@gmail.com>

5.1.4 Bug reports

- Carlos Feliciano <c.feliciano2009@gmail.com>
- Soynerdito [https://github.com/soynerdito]
- Rafael Arce [https://github.com/ran-research-lab]