RESTo
restful semantic search tool for geospatial

Jérôme Gasperi
What is it?

RESTo provides a *semantic search* service on Earth Observation data.
RESTo implements OpenSearch [OGC13-026 - OpenSearch Extension for Earth Observation]
OpenSource

RESTo is written in PHP and uses PostgreSQL + PostGIS
RESTful

RESTo follows a RESTful approach to manage resources
RESTful

RESTo follows a RESTful approach to manage resources

http://localhost/resto/collection/identifier/

e.g. http://localhost/resto/Spirit/SPI_11005/
<table>
<thead>
<tr>
<th>URI</th>
<th>HTTP method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://localhost/resto/">http://localhost/resto/</a></td>
<td>GET</td>
<td>List all collections</td>
</tr>
<tr>
<td><a href="http://localhost/resto/">http://localhost/resto/</a></td>
<td>POST</td>
<td>Create a new collection</td>
</tr>
<tr>
<td><a href="http://localhost/resto/collection/_describe">http://localhost/resto/collection/_describe</a></td>
<td>GET</td>
<td>Describe collection OpenSearch service</td>
</tr>
<tr>
<td><a href="http://localhost/resto/collection">http://localhost/resto/collection</a></td>
<td>GET</td>
<td>Search collection</td>
</tr>
<tr>
<td><a href="http://localhost/resto/collection">http://localhost/resto/collection</a></td>
<td>POST</td>
<td>Insert a resource within collection</td>
</tr>
<tr>
<td><a href="http://localhost/resto/collection">http://localhost/resto/collection</a></td>
<td>DELETE</td>
<td>Delete collection</td>
</tr>
<tr>
<td><a href="http://localhost/resto/collection">http://localhost/resto/collection</a></td>
<td>PUT</td>
<td>Update collection</td>
</tr>
<tr>
<td><a href="http://localhost/resto/collection/identifier">http://localhost/resto/collection/identifier</a></td>
<td>GET</td>
<td>Show resource metadata</td>
</tr>
<tr>
<td><a href="http://localhost/resto/collection/identifier/download">http://localhost/resto/collection/identifier/download</a></td>
<td>GET</td>
<td>Download resource product</td>
</tr>
</tbody>
</table>
Architecture

(*) Collections can be stored within RESTo database or in external databases

RESTo

RESTo backend

uses

resto db

PostgreSQL

PostGIS

hstore

EO collections *

External services

mapshup

iTag

RESTo modules

Gazetteer

Collection Manager

Query Storage

Query Analyzer

Resource Manager

More to come...

RESTo frontend

Core

PostgreSQL

hstore

uses

resto db

EO collections *
RESTo in action
List all collections

$ curl -X GET http://localhost/resto/
Create a collection

$ curl -k -X POST -F "file[]=@Spot.json" https://admin:nimda@localhost/resto/
Create a collection

$ curl -k -X POST -F "file[]=@Spot.json" https://admin:nimda@localhost/resto/

Case 1: collection does not exist. Database is created within RESTTo backend
e.g. https://github.com/jjrom/resto/blob/master/_examples/collections/Spot.json

Case 2: collection exist in legacy database. Collection is referenced within resto
e.g. https://github.com/jjrom/resto/blob/master/_examples/collections/Charter.json
Delete a collection

$ curl -k -X DELETE https://admin:nimda@localhost/resto/Spot
Describe a collection

$ curl -X GET http://localhost/resto/Spot/_describe
Describe a collection

$ curl -X GET http://localhost/resto/Spot/_describe

Output formats are: Atom, HTML and GeoJSON (RDF to come soon !)
Insert a resource

$ curl -k -X POST -F "file[]=@r.json" https://admin:nimda@localhost/resto/Spot

e.g. https://github.com/jjrom/resto/blob/master/_examples/resources/resource_Spot.json
Insert a resource

During ingestion process, the resource is automatically tagged with location and land use.

github.com/jjrom/itag
Search

$ curl -X GET http://localhost/resto/Spot?q=Toulouse&cloudCover=[0,20]&...
Search

$ curl -X GET http://localhost/resto/Spot?q=Toulouse&cloudCover=[0,20] &...

Here are the OpenSearch parameters
Search

RESTo uses the Query Analyzer to translate query into a set of EO OpenSearch parameters
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern.
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern.
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern.

1. Split query string into list of unitary words
2. Extract «key=value» strings
3. Extract Platforms and Instruments
4. Remove excluded words and non dictionary words with length < 4 characters

Example:
- e.g. `orbitNumber=4`
- Platforms and instruments list are stored within common dictionary
  [Link](https://github.com/jjrom/resto/blob/master/resto/dictionaries/common.php)
- e.g. «area of Mexico in 2012»
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern.

Split query string into list of unitary words

Extract «key=value» strings

Extract Platforms and Instruments

Remove excluded words and non-dictionary words with length < 4 characters

Extract patterns and dated

- e.g. orbitNumber=4
- Platforms and instruments list are stored within common dictionary at https://github.com/jjrom/resto/blob/master/resto/dictionaries/common.php
- e.g. «area of Mexico in 2012»
- e.g. «acquired in the last 2 days»
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern

1. Split query string into list of unitary words
2. Extract «key=value» strings
3. Extract Platforms and Instruments
4. Remove excluded words and non dictionary words with length < 4 characters
5. Extract patterns and dated
6. Extract keywords

- e.g. `orbitNumber=4`
- Platforms and instruments list are stored within common dictionary [https://github.com/jjrom/resto/blob/master/resto/dictionaries/common.php](https://github.com/jjrom/resto/blob/master/resto/dictionaries/common.php)
- e.g. `area of Mexico in 2012`
- e.g. `acquired in the last 2 days`
- e.g. `urban area in France`
Search (Query Analyzer)

Query string analysis algorithm is based on simple recognition of words and pattern.

- Split query string into list of unitary words
- Extract «key=value» strings
  - e.g. `orbitNumber=4`
- Extract Platforms and Instruments
- Platforms and instruments list are stored within common dictionary
- Remove excluded words and non dictionary words with length < 4 characters
  - e.g. «area of Mexico in 2012»
- Extract patterns and dated
  - e.g. «acquired in the last 2 days»
- Extract keywords
  - e.g. «urban area in France»
- Extract location on remaining words
  - e.g. «images acquired in Toulouse»
Recognized patterns

<with> "keyword"
<without> "keyword"

"quantity" <lesser> (than) "numeric" "unit"
"quantity" <greater> (than) "numeric" "unit"
"quantity" <equal> (to) "numeric" "unit"
<lesser> (than) "numeric" "unit" (of) "quantity"
<greater> (than) "numeric" "unit" (of) "quantity"
<equal> (to) "numeric" "unit" (of) "quantity"
"quantity" <between> "numeric" <and> "numeric" ("unit")
<between> "numeric" <and> "numeric" "unit" (of) "quantity"
<today>
<yesterday>
<before> "date"
<after> "date"
<between> "date" <and> "date"
"numeric" "(year\day\month)" <ago>
<last> "(year\day\month)"
<last> "numeric" "(year\day\month)"
"numeric" <last> "(year\day\month)"
"(year\day\month)" <last>
<since> "numeric" "(year\day\month)"
<since> "month" "year"
<since> "date"
<since> "numeric" <last> "(year\day\month)"
<since> <last> "numeric" "(year\day\month)"
<since> <last> "(year\day\month)"
<since> "(year\day\month)" <last>
$dictionary = array(  
    'excluded' => array(  
        'than',
        'over',
        ...
    ),  
    'modifiers' => array(  
        'ago' => 'ago',
        'before' => 'before',
        'after' => 'after',
        ...
    ),  
    'units' => array(  
        'm' => 'm',
        'meter' => 'm',
        'days' => 'days',
        ...
    ),  
    'numbers' => array(  
        'one' => '1',
        ...
    ),  
    'months' => array(  
        'january' => '01',
        ...
    ),  
    'quantities' => array(  
        'resolution' => 'resolution',
        ...
    ),  
    'keywords' => array(  
        'continent' => array(  
            'europe' => 'europe',
            ...
        )
    )
);
$dictionary = array(
    'excluded' => array(
        'than',
        'over',
        ...
    ),
    'modifiers' => array(
        'ago' => 'ago',
        'before' => 'before',
        'after' => 'after',
        ...
    ),
    'units' => array(
        'm' => 'm',
        'meter' => 'm',
        'days' => 'days',
        ...
    ),
    'numbers' => array(
        'one' => '1',
        ...
    ),
    'months' => array(
        'january' => '01',
        ...
    ),
    'quantities' => array(
        'resolution' => 'resolution',
        ...
    ),
    'keywords' => array(
        'continent' => array(
            'europe' => 'europe',
            ...
        )
    )
);
Multilingual - current languages are EN, FR, IT and DE
Synonyms supported (e.g. unit «m» is «m», «meter» or «meters»)
Detectable words are stored within a dictionary

$dictionary = array(
    'excluded' => array(
        'than',
        'over',
        ...
    ),
    'modifiers' => array(
        'ago' => 'ago',
        'before' => 'before',
        'after' => 'after',
        ...
    ),
    'units' => array(
        'm' => 'm',
        'meter' => 'm',
        'days' => 'days',
        ...
    ),
    'numbers' => array(
        'one' => '1',
        ...
    ),
    'months' => array(
        'january' => '01',
        ...
    ),
    'quantities' => array(
        'resolution' => 'resolution',
        ...
    ),
    'keywords' => array(
        'continent' => array(
            'europe' => 'europe',
            ...
        )
    )
);

Multilingual - current languages are EN, FR, IT and DE
Synonyms supported (e.g. unit «m» is «m», «meter» or «meters»)
Each collection can define its own dedicated keywords
Multilingual - current languages are EN, FR, IT and DE
Synonyms supported (e.g. unit «m» is «m», «meter» or «meters»)
Each collection can define its own dedicated keywords
Automatic typing error correction using similarity function

Search  (Query Analyzer)

Detectable words are stored within a dictionary

```php
$dictionary = array(
    'excluded' => array(
        'than',
        'over',
        ...
    ),
    'modifiers' => array(
        'ago' => 'ago',
        'before' => 'before',
        'after' => 'after',
        ...
    ),
    'units' => array(
        'm' => 'm',
        'meter' => 'm',
        'days' => 'days',
        ...
    ),
    'numbers' => array(
        'one' => '1',
        ...
    ),
    'months' => array(
        'january' => '01',
        ...
    ),
    'quantities' => array(
        'resolution' => 'resolution',
        ...
    ),
    'keywords' => array(
        'continent' => array(
            'europe' => 'europe',
            ...
        )
    )
);```
Detectable words are stored within a dictionary

Multilingual - current languages are EN, FR, IT and DE

Synonyms supported (e.g. unit «m» is «m», «meter» or «meters»)

Each collection can define its own dedicated keywords

Automatic typing error correction using similarity function

Search (Query Analyzer)

e.g. https://github.com/jjrom/resto/blob/master/resto/dictionaries/dictionary_en.php

$dictionary = array(
    'excluded' => array(
        'than',
        'over',
        ...
    ),
    'modifiers' => array(
        'ago' => 'ago',
        'before' => 'before',
        'after' => 'after',
        ...
    ),
    'units' => array(
        'm' => 'm',
        'meter' => 'm',
        'days' => 'days',
        ...
    ),
    'numbers' => array(
        'one' => '1',
        ...
    ),
    'months' => array(
        'january' => '01',
        ...
    ),
    'quantities' => array(
        'resolution' => 'resolution',
        ...
    ),
    'keywords' => array(
        'continent' => array(
            'europe' => 'europe',
            ...
        )
    )
);
Search (Gazetteer)

RESTo embeds a Gazetteer service to detect location
Search (Gazetteer)

RESTo embeds a Gazetteer service to detect location

Based on geonames database
Search (Gazetteer)

RESTo embeds a Gazetteer service to detect location

Based on geonames database
More than 9 000 000 toponyms
Search (Gazetteer)

RESTo embeds a Gazetteer service to detect location

Based on geonames database
More than 9,000,000 toponyms
Multilingual
Search (example)

« Pleiades images of urban area in France acquired in 2013 with less than 25 % of cloud cover »
Search (example)

« Pleiades images of urban area in France acquired in 2013 with less than 25% of cloud cover »

platform    keyword    location    date    acquisition parameter
Search (result)

unique unambiguous url

keywords are links
Search (result)

Each search result has a unique url i.e. can be indexed by web crawler (i.e. google robots)
Search (result)

Each search result has a unique url i.e. can be indexed by web crawler (i.e. google robots)

Resource associated keywords are presented as search links on each keyword.
Search (result)

Each search result has a unique url i.e. can be indexed by web crawler (i.e. google robots)

Resource associated keywords are presented as search links on each keyword.

Thus they can be indexed by web crawler...and so on.
Search (result)

Each search result has a unique url i.e. can be indexed by web crawler (i.e. google robots)

Resource associated keywords are presented as search links on each keyword.

Thus they can be indexed by web crawler...and so on.

Theorically the whole database could be indexed by web crawlers
Demo

http://mapshup.info/resto
github.com/jjrom/resto