Modern ways of spatial data publication

Report of the research results in the Geonovum testbed "Spatial Data on the Web"

22/08/2016

Authors
T. van Sprundel, Spotzi BV
E. Nagelkerke, Spotzi BV
R. Dolman, Spotzi BV
M. Pegels, Spotzi BV

Table of Contents

Abstract 1

Motivation 2

Approach 3

- **Data testing** 4
- **Web application** 8
- **Connection to the data** 9
- **Other connections** 11
- **Adding logic to the web application** 12

Evaluation 13

Appendixes 14

Abstract

Motivation

Last year Spotzi applied for topic 2 of the project Spatial data on the web. In a few months we've created a platform that can be used by different authorities to share their data. During this project Spotzi researched the weaknesses and strengths of an open spatial data platform.

A lot of the participating and spectating parties responded positively on the platform and were amazed by what is possible in such a short timeframe. Spotzi experienced an excellent cooperation with Geonovum during this project and therefor Spotzi is willing to follow up this project with the newest tender from Geonovum.

Because Spotzi participated in the previous part of the Spatial data on the web project, we are familiar with all the ins and outs of the project. In addition to the information provided on the project Github page, we also have knowledge of what was discussed in the meetings and conversations during the last project. This knowledge can not only help us to work with the data, but also helps us in reviewing the lessons learned. Are there still missing parts or are the lessons learned clear enough.

Spotzi believes that it has a good overview on the market and its needs, therefor Spotzi can create an application that actually will be used by different users. We know there is an ongoing increase in the demand for simple and understandable applications to save time for the user.

Furthermore, Spotzi is a spin off from the Risk Management Firm, D-Risk. One of the projects D-Risk was involved in was the renovation of the Rijksmuseum in Amsterdam in 2006. On behalf of the involved insurance companies D-Risk monitored the potential risks during the renovation works. Besides the renovation of the Rijksmuseum, D-Risk was also involved in many sewer renewal projects throughout the Netherlands in for example the municipality of Zaanstad. These projects have one thing in common: lack of insight and understanding of invisible, e.g. vibrations, and often unknown effects, like the sphere of influence of ground-water pumping, on its surrounding and specifically cultural heritage sites.

Construction projects in or near cultural heritage sites can affect the condition of the building and the objects inside. We found that there is still a lack of understanding on the relationship between the effects of construction works and its immediate surroundings.

Spotzi believes we can create an application that is simple to use and makes use of the delivered data sources from task 1. The construction knowledge of D-Risk in combination with the geo knowledge of Spotzi creates a strong combination. In construction, you have an average of five to fifteen percent failure costs, with this project we want to contribute to lowering this failure costs and contribute to the protection and preservation of Cultural Heritage sites in the Netherlands. Furthermore, we want to contribute to the quality of our environment in compliance with, and extension of the environmental law. The Noise Abatement Act exists in the Netherlands, but there is for example no legislation to prevent nuisance or damage caused by vibrations (*Kenniscentrum InfoMil, Beleid, wet- en regelgeving, 2016*). That makes creating our application (see chapter 3) even more important.

Approach

This report discusses the following aspects:

- *Testing the data provided by task 1*
- *Creating the web application*
- *Create a connection to the data*
- *Create other connections*
- *Add logic to the web application*
- *Evaluate*

In the next chapters we explain what we've done for per aspect.

Research

Data testing

Triply acts as data publisher, they will publish data, like the Cultural Heritage data, in compliance with the lessons learned from the project Spatial data on the Web. This section focuses on testing the Cultural Heritage data provided by Triply. Answers will be provided on the questions: Is the Cultural Heritage data findable and understandable? Is it possible for us to retrieve data from this dataset and how did the retrieval of the data go? We will also look into the different content-types Triply delivers. Finally, is the provided data findable by crawlers? Finding, accessing and using data is often difficult for non(geo)-experts.

Findable

Triply realized the Geo API which can be accessed via http://146.185.182.204/. This data providing platform provides (accessed on 1st July 2016) BGT data.

We noticed this data was about the BGT because of the word BGT on the landing page. BGT is also mentioned on the tabular page. We, Spotzi, know the BGT is a geo-spatial dataset, others may not have knowledge of the existence of the BGT. For some of our developers it was not clear what data was available on the platform. Adding a simple table of contents containing the current data makes it a lot easier to see the full data content of the platform. This together with a short description about the data and for example links to describing wikis makes it even more clear.

After some period of development the Geo API was accessible through the web-URL http://geonovum.triply.cc/ (accessed on 18th August 2016). At this time the Cultural Heritage data was published under nsid:dataset/monuments. We looked at this data through the different endpoints the website provided. These endpoints will be further explained in the chapter *"Connecting to the data".*

Understandable

The same criteria, about foreknowledge, as described above most be fulfilled concerning so the published data becomes understandable to developers. The Semantic data model or vocabulary of the published data should be implemented with an easy overlook of the full dataset or should be at least easy accessible, e.g. a simplified representation of the full dataset. Currently, the vocabulary can be accessed, but this itself is quite complex.

Firstly, we advise Triply to take a good look at the report of topic 4, and especially chapter *Representations.* We noticed structure within the data on the GEO API of triply, but this is mainly because we have foreknowledge about this dataset.

Secondly, we advise Triply to take a look at their website and improve the usability of their website. Usability is one of the key elements of understanding. For instance, making it available to switch between response formats on the website will make it easier to understand the difference between them.

Connection to the data

Connecting

For testing the connection to the data provided on the data platform, we've let different developers from Spotzi create their own connection. All of these developers are experienced programmers, but not all of them have experience with linked and/or geo data.

Concerning the time to the first successful call (TTFSC), we've encountered no problems. All of our developers where able to make a connection to the platform and retrieve data within 10 minutes. This short TTFSC was mainly the result of the clear examples found on the Github of topic one.

Parameters

We were able to use a number of mandatory parameters in our calls to the platform, like the *lat*, *lng*, *page* and *page_size*. We would recommend to also list these parameters in an API reference on the website, so it can be made clear what each parameter means and what syntax should be used for each parameters.

We are also interested in other parameters that we could add to our request. For example, a way to filter the data, so we could search for the type "*polygon*", close to the location we're requesting data from.

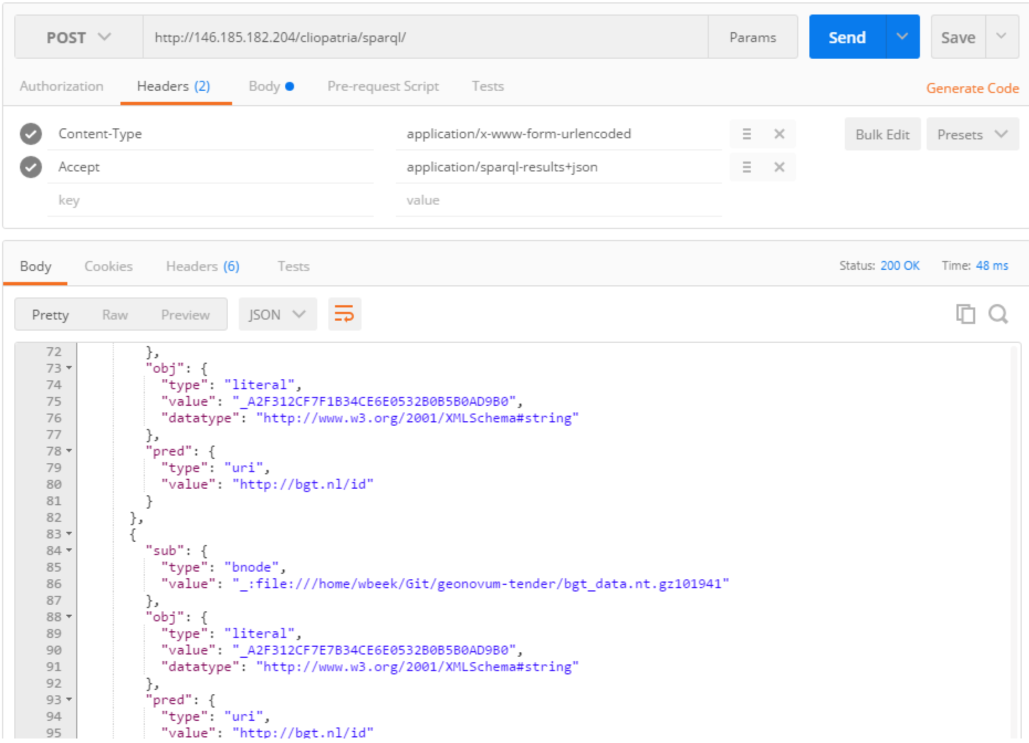

```

Geonovum testing environment

string(1725) "{
  "number_of_results":1,
  "page":1,
  "page_size":1,
  "results": [
    {
      "geometry": {
    "coordinates": [
      [
        [5.470273326791917, 51.33993468399041 ],
        [5.470214406849209, 51.33999313484354 ],
        [5.469136032543656, 51.339562989169565 ],
        [5.468690401805991, 51.339385145779175 ],
        [5.468311265800884, 51.339233411511195 ],
        [5.4677751681695845, 51.33902005930252 ],
        [5.4677643377107525, 51.33900633202976 ],
        [5.467838400715708, 51.33898094144799 ],
        [5.467838400715708, 51.33898094144799 ],
        [5.46784005014113, 51.338980418957455 ],
        [5.46784393744009, 51.33897940952305 ],
        [5.467847681770863, 51.338978705804514 ],
        [5.467851254352364, 51.33897826287821 ],
        [5.46785461206854, 51.338978044819086 ],
        [5.467857726122368, 51.33897799771479 ],
        [5.467860854797295, 51.338978103408735 ],
        [5.467864184717932, 51.33897839772623 ],
        [5.467867730298763, 51.33897891661223 ],
        [5.467871448550913, 51.33897969605146 ],
        [5.467875267815624, 51.33898079002605 ],
        [5.467876890447036, 51.338981346201166 ],
        [5.467876890447036, 51.338981346201166 ],
        [5.468633621875931, 51.339284547748456 ],
        [5.468820083150952, 51.33935630884402 ],
        [5.469003561715456, 51.33942902454542 ],
        [5.469189773211241, 51.33950488408579 ],
        [5.469366850845165, 51.33957674081115 ],
        [5.469562185655451, 51.339649734891545 ],
        [5.469796781883229, 51.33974439955012 ],
        [5.470071532842744, 51.33985446876821 ],
        [5.470273326791917, 51.33993468399041 ]
      ]
    ],
    "type":"Polygon"
      },
      "properties": {
    "@id":"_:file:///home/wbeek/Git/geonovum-tender/bgt_data.nt.gz9406"
      },
      "type":"Feature"
    }
  ]
}"
```

SPARQL

We've also let our developers make a connection trough the [SPARQL endpoint](). The TTFSC here was a lot higher, with an average time over 30 minutes. This was mainly due the lack of documentation and explanation on the website. We will go deeper into this subject in the chapter "*Connection to the data*".



Content-types

We tried to request different content-types from the platform, according to [lesson 3A] of the lessons learned. At the moment the platform supports the following content-types:

Requested types trough the map endpoint:

- JSON - application/json
- JSON-LD - application/ld+json

Requested types trough the SPARQL endpoint:

- XML - application/sparql-results+xml
- JSON - application/sparql-results+json

On the topic-1-task-1 Github page, Triply also describes how to retrieve results in N-Triples (application/n-triples). However, when we tried to retrieve this content type (27th of june 2016) we were not able to retrieve them.

GeoJson

Regarding retrieving results in GeoJson, Drs. Bas Vanmeulebrouk asked Triply if it's possible to retrieve this content type from the platform. We're awaiting the answer on this question on Slack at this moment.

Findable by crawlers

XML sitemap

At this moment the data platform does not provide an XML sitemap.

Persistent URIs

At this moment the data platform does not yet use persistent URIs for its data. There are for example references to the local directory from the Triply server and nonexistent links are used. For example, http://bgt.nl/id.

Links

In the vocabulary graph there are links to formal definitions of the BGT. However, the links that we found in the BGT data graph are all nonexistent. They all link to http://bgt.nl/. We expect that this is a temporary substitute of the definitive link.

Web application

We've created a simple web application on http://geonovum.spotzi.com/task2. The web application provides a way to test the Triply-API in a real-time environment we created. That is, a map viewer with the functionality to import data from the Triply-API into the web browser. To test the Triply API we created three different types of search-functions.

Zip code search

The first function provides a way for the user to find cultural heritage date by entering a Dutch zip code (e.g. 9999ZZ). Once submitted, the zip code will be located through a Spotzi-service and the location will be used to retrieve data. Which will be send back to the browser to plot on the map. This function can be tested through the link.

Area search

The second function provides a way for the user to specify an area to find data. The user selects a range and clicks on the map to plot a circle. The user then has the option to submit the area to the server or drag the circle to overlap a different location. Once the user is satisfied about the location s/he selected and submits, the server will then take the input and look for data on the Triply-API to retrieve back. *This function is still being developed*.

Category search

The third function provides a way to find data by category. Ultimately, this function would work best if the available categories and types, within the Cultural Heritage data, are loaded live into the web application. This should be done when opening the page or at the first time use of function. Once that data is available to the web application, it can be placed in a list selection so that the user can select the desired category. On submit, the server will again connect to the Triply-API and retrieve the data which will be filtered with the submitted category. *This function is still being developed.*

Connection to the data

This section focuses on creating a connection to the Cultural Heritage data provided by Triply. Answers will be provided on the questions: Is the Cultural Heritage data clear for developers? How, and can we query and visualize the data? Delivers the applications a good user experience?

Usability

Firstly, we browsed the monument data through the **graph** endpoint with help of different response-formats. Here the data could be browsed/filtered on base of 'normal' non-geographic data. There was no notice of the availability of response formats within the /graph page on the browser, the only accessible format was through HTML. According to *10 Heuristics for User Interface Design*[1] of Jakob Nielsen, this is an usability issue on the point *Recognition rather than recall*. Which tells "*Instructions for use of the system should be visible or easily retrievable whenever appropriate*". Next to that, the following issues came up:

- **Clicking on Next within the /graph page would lead to the home page**
  This was fixed by setting the page attribute in the request-url
- **Responses on erroneous requests were not clear**
  Requesting too many items per page or an out of range page number lead to a 500 error
- **Links in data incorrect**
  Link to http://data.cultureelerfgoed.nl:3333/1000001 leads to an unavailable website

These points conflict with the findability because it slows down the developer's learning process of understanding the API.

Secondly there was a **geo** endpoint, here the data could be retrieved on base of location. Through the browser this could be done by clicking on a map. Then the browser would make a request and plot the retrieved objects in the response. Also here missed the link to the different response formats which will be used by computers.

Thirdly there was an **sparql** endpoint, for this endpoint there was no documentation/example available.

The documentation for this could be found in the **doc** endpoint of the Triply-website. Here the documentation on the graph and geo endpoint could be found. Documentation on the sparql-endpoint was still missing.

[1] https://www.nngroup.com/articles/ten-usability-heuristics/

Adding the data to the platform (without storage)

TODO: explain live data management

Application performance

TODO: perform tests in new server environment

The application performance decreased as the page number increased.

Graph endpoint performance

Page number	Load time (ms)
5	361
25	383
50	368
100	418
250	706
500	972
5.000	3.170
50.000	31.380

Other connections

In this section we are going to expand the application by adding other datasets from different data sources.

Connection BGT and Cultural Heritage data

TODO

Connection data, information models and metadata

TODO

Adding logic to the web application

Programmability

Zip code search

TODO: further results implementing this function

Area search

TODO: further results implementing this function

Category search

TODO: further results implementing this function

Usability end-user

Evaluation

Lessons learned

Improvements

Appendixes