Bridging Web APIs and Linked Data with SPARQL Micro-Services

Franck Michel, Catherine Faron-Zucker, and Fabien Gandon

Université Côte d'Azur, Inria, CNRS, I3S (UMR 7271), France
franck.michel@cnrs.fr, faron@i3s.unice.fr, fabien.gandon@inria.fr

Abstract. Web APIs are a prominent source of machine-readable information that remains insufficiently connected to the Web of Data. To enable automatic combination of Linked Data (LD) interfaces and Web APIs, we present the SPARQL Micro-Service architecture. A SPARQL micro-service is a lightweight SPARQL endpoint that provides access to a small, resource-centric, virtual graph, while dynamically assigning stable, dereferenceable URIs to Web API resources that do not have URIs in the first place. We believe that the emergence of an ecosystem of SPARQL micro-services could enable LD-based applications to glean pieces of data from a wealth of distributed, scalable and reliable services from independent providers. We describe an experimentation where we dynamically augment biodiversity-related LD resources with data from Flickr, MusicBrainz and the Macauley scientific media library.

Keywords: Web API, SPARQL, micro-service, Linked Data, JSON-LD

1 Introduction

Web APIs are commonly used to enable HTTP-based, machine-processable access to all sorts of data. Similarly, Linked Data (LD) seek the publication of machine-readable data on the Web while connecting related resources accross datasets. Several approaches have been proposed to bridge these two worlds, based on bespoke wrappers, SPARQL extensions or RESTful APIs. Until now however, several challenges hinder the definition of standard approaches to enable automatic reconciliation of LD and Web APIs. Firstly, Web APIs usually rely on proprietary vocabularies documented in Web pages meant for developers but hardly machine-readable. Secondly, on-the-fly SPARQL querying of non-RDF data sources proves to be difficult, as attested by the many works on SPARQL-based access to legacy databases.

The *SPARQL Micro-Service* architecture aims to address this issue. The term *micro-service* refers to an increasingly popular architectural style where an application consists of a collection of lightweight, loosely-coupled, fine-grained services that are deployed independently [2]. They improve applications modularity thereby speeding up the development, testing and deployment process. Leveraging these principles may help in the design of modular LD-based applications structured as a collection of services such as RDF stores, SPARQL

endpoints, SPARQL micro-services or other services based on LD REST APIs for instance. A SPARQL micro-service is a lightweight method to query a Web API using SPARQL. It provides access to a small RDF graph describing the targeted resources from the data available at the API, while dynamically assigning dereferenceable URIs to Web API resources that do not have URIs beforehand.

2 SPARQL Micro-Services Principles

A SPARQL micro-service S_μ is a wrapper of a Web API service S_w. It complies with the SPARQL Query Language and protocol, and accepts a set Arg_w of arguments that are specific to S_w. These arguments are passed to S_μ as parameters on the HTTP query string, *e.g.* "http://hostname/sparql?param=value". S_μ evaluates a SPARQL query Q against an RDF graph that it builds at runtime as follows: it invokes the Web API service S_w with the arguments in Arg_w, translates the response into RDF triples (in an implementation-dependent manner), evaluates Q against these triples and returns the result to the client.

The semantics of a SPARQL micro-service differs from that of a standard SPARQL endpoint insofar as the SPARQL protocol treats a service URL as a black box, *i.e.* it does not interpret URL parameters. By contrast, a SPARQL micro-service is a configurable SPARQL endpoint whose arguments delineate the virtual graph being queried. Thence, each pair (S_μ, Arg_w) is a standard SPARQL endpoint. Arguably, other options may be adopted to pass the arguments to S_μ, that we discuss in details in [1].

Furthermore, bridging Web APIs and LD requires to create stable, dereferenceable URIs for Web API resources that are generally identified by mere proprietary identifiers. This is implemented quite easily with SPARQL micro-services: once a stable URI scheme is decided, a Web server is set up to handle look-ups for URIs matching that scheme: for each look-up, it invokes a suitable SPARQL micro-service along with an appropriate CONSTRUCT or DESCRIBE query form. Hence, by smartly designing services, we can come up with a consistent ecosystem where some micro-services respond to SPARQL queries while translating Web API identifiers into URIs that, in turn, are made dereferenceable by other micro-services.

3 Implementation and Experimentation

To evaluate this architecture, we have developed a prototype implementation[1] depicted in Figure 1, that handles JSON-based Web APIs. It maps a Web API response to RDF triples in two steps. First, the response is translated to JSON-LD by applying a JSON-LD profile. The resulting graph G is stored in an in-memory triple store. Then, for mapping cases that JSON-LD cannot describe (involving *e.g.* string manipulations), a SPARQL INSERT query augments G with triples based on well-adopted or domain vocabularies. Lastly, S_μ evaluates the client's

[1] https://github.com/frmichel/sparql-micro-service

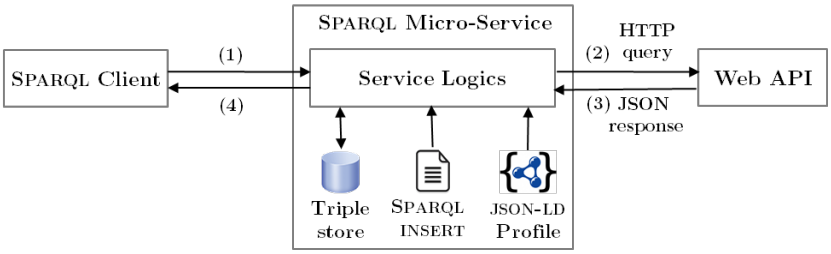

Fig. 1. Example SPARQL micro-service implementation for JSON-based Web APIs.

```
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix schema: <http://schema.org/>

CONSTRUCT {
  ?species
    schema:subjectOf ?photo; foaf:depiction ?img;     # from Flickr
    schema:contentUrl ?audioUrl;  # from the Macaulay Library
    schema:subjectOf ?page.       # from MusicBrainz
} WHERE {
  SERVICE <https://taxref.mnhn.fr/sparql>
    { ?species a owl:Class; rdfs:label "Delphinus delphis". }
  SERVICE <https://example.org/sparql-ms/flickr/getPhotosByGroupByTag
           ?group_id=806927@N20&tags=taxonomy:binomial=Delphinus+delphis>
    { ?photo foaf:depiction ?img. }
  SERVICE <https://example.org/sparql-ms/
           macaulaylibrary/getAudioByTaxon?name=Delphinus+delphis>
    { [] schema:contentUrl ?audioUrl. }
  SERVICE <https://example.org/sparql-ms/
           musicbrainz/getSongByName?name=Delphinus+delphis>
    { [] schema:sameAs ?page. } }
```

Listing 1.1. Querying SPARQL micro-services to enrich a LD resource with data from Flickr, the Macaulay Library and MusicBrainz.

SPARQL query against G and returns the response following a regular content negotiation.

Listing 1.1 exemplifies the use of SPARQL micro-services in a biodiversity-related use case. The query retrieves the URI of a resource representing the common dolphin (species *Delphinus delphis*) from a taxonomic register. Then, it invokes SPARQL micro-services to retrieve additional data from three Web APIs: photos from the Flickr photography social network[2], audio recordings from the Macaulay Library[3] and music tunes from MusicBrainz[4]. Figure 2 portrays a snippet of the response to this query in the Turtle syntax, along with photos, audio recordings pictures and a MusicBrainz Web page.

[2] https://www.flickr.org/
[3] https://www.macaulaylibrary.org/
[4] https://musicbrainz.org/

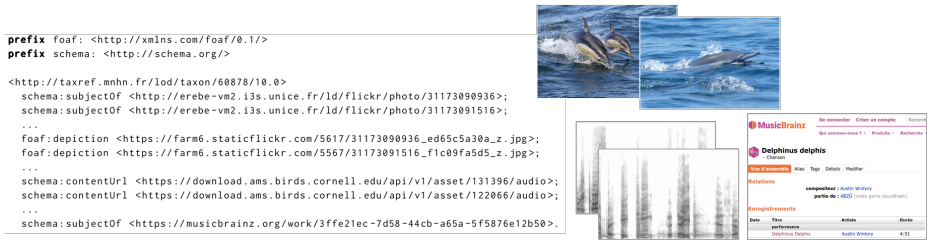

Fig. 2. Snippet of the response to query Q (Listing 1.1) along with snapshots of the images, audio recordings and Web page whose URLs are part of the response.

4 Future Works

For an ecosystem of SPARQL micro-services to emerge from independent service providers, two crucial issues shall be tackled. Firstly, to enable services discovery, SPARQL micro-services should provide self-describing metadata such as the query string parameters or the types of triples generated. In this respect, the smartAPI metadata specification may be leveraged [4]. Secondly, it should be possible to retrieve fragments by smaller pieces using a paging mechanism. To tackle those issues, *Triple Patterns Fragments* (TPF) expose a self-describing, uniform interface consisting of metadata and hypermedia controls [3]. A perspective would be to extend this approach to the case of SPARQL micro-services, stemming some sort of *Graph Pattern Fragment* interface, *i.e.* a generalized TPF interface able to process regular graph patterns instead of only triple patterns, but still complying with the TPF metadata and hypermedia controls specification. Let us finally mention that we have focused specifically on consuming Web APIs data with SPARQL, although the principles presented in this work could apply to other types of APIs. Furthermore, many APIs empower users not only to read but more importantly to interact with data. Hence, an interesting perspective would be to think of SPARQL micro-services as a way to support distributed SPARQL Update over Web APIs, thus eventually contributing to build an actual read-write Web of Data.

References

1. F. Michel, C. Faron-Zucker, and F. Gandon. SPARQL micro-services: Lightweight integration of web APIs and linked data. In *Proc. of LDOW 2018*, 2018.
2. S. Newman. *Building Microservices*. O'Reilly Media, 2015.
3. R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert. Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web. *J. Web Semantics*, 37–38:184–206, 2016.
4. A. Zaveri, S. Dastgheib, T. Whetzel, R. Verborgh, P. Avillach, G. Korodi, R. Terryn, K. Jagodnik, P. Assis, C. Wu, and M. Dumontier. smartAPI: Towards a more intelligent network of Web APIs. In *Proc. of 14th ESWC*, pages 154–169, 2017.