BASIL: A Cloud Platform for Sharing and Reusing SPARQL Queries as Web APIs

Enrico Daga, Luca Panziera, and Carlos Pedrinaci

Knowledge Media Institute (KMI) - The Open University.
Walton Hall, MK76AA Milton Keynes, United Kingdom
{enrico.daga,luca.panziera,carlos.pedrinaci}@open.ac.uk
http://kmi.open.ac.uk

Abstract. One of the reasons why Web APIs are more used to consume open data compared to SPARQL endpoints is the expertise required to use the query language. Therefore, a tool for sharing and reusing existing real queries could help developers on adopting Linked Data. We propose BASIL, a cloud platform that supports sharing and reusing SPARQL queries. In BASIL, loaded queries generate Web APIs that can be used in applications instead of embedding the call to the SPARQL endpoints, thus facilitating query maintenance and evolution. Compared to similar solutions, BASIL aims on minimising the learning curve for users to promote its adoption. BASIL is a simple platform that does not introduce new specifications, formalisms and technologies for users that belong to both Web APIs and Linked Data communities.

1 Introduction

The availability of datasets as Linked Open Data (LOD) is rapidly increasing on the Web [4]. Billions of linked statements are publicly available as RDF triples through hundreds of SPARQL endpoints, potential data sources to develop distributed and rich Web application. However, the current most adopted approach for publishing and consuming open data are Web APIs. The higher adoption of Web APIs is visible by comparing information provided by both the *datahub.io*[1] Linked Data catalog and *ProgrammableWeb*[2], the most popular Web API catalog, which includes also SPARQL endpoints. According to *datahub.io*, more that 1000 SPARQL endpoints are currently available on the Web on June 2015. At the same time, *DBpedia* is the only SPARQL endpoint that appears in the top 500 most popular Web APIs on *ProgrammableWeb*.

In this scenario, we propose BASIL as a solution to support developers in consuming data from SPARQL endpoints. BASIL (Builging Apis SImpLy) is a cloud platform that allows data consumers to tailor their own Web APIs by specifying queries on SPARQL endpoints. Tailored Web APIs and related SPARQL queries can be shared, reused and cloned by other users. This demo paper

[1] http://datahub.io/
[2] http://www.programmableweb.com/

presents the cloud based service deployed at `http://basil.kmi.open.ac.uk`, also based on a dedicated user interface (PESTO).

2 The BASIL cloud platform

The design of the BASIL follows a specific philosophy: to minimise the learning curve for the users in order to foster its adoption. To reach this objective, BASIL (i) is modelled to be as simple as possible, (ii) is designed by using well-know Web API practices and (iii) minimises the introduction of new formalisms. The architecture of BASIL is provided in figure 2. The BASIL platform plays the role of a mediator between SPARQL endpoints and developer's applications. By submitting a SPARQL query and an endpoint to BASIL, a developer generates a Web API. Figure 1 shows an example, that is also reachable at `http://basil.`

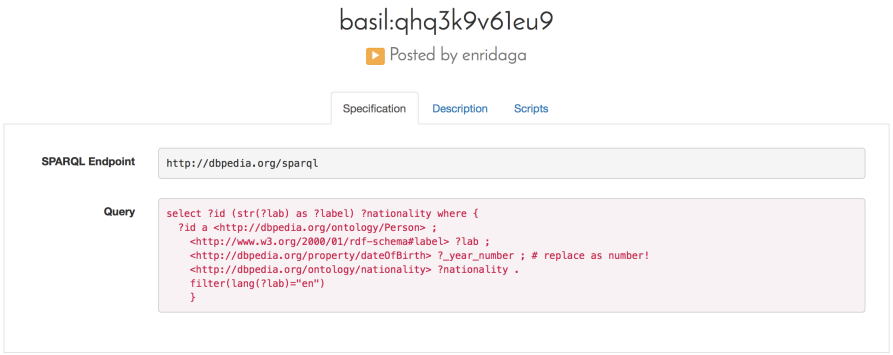

Fig. 1: A query as appears in the BASIL environment.

`kmi.open.ac.uk/basil/qhq3k9v61eu9`. The API extracts a list of people born on a given year from DBPedia.

There are few rules to specify if and how a variable maps to a request parameter of the API. By putting a "_" character in front of their label (e.g. `?_param`) users define mandatory parameters. With two underscores, the parameter is optional. After the name, they can specify how the value needs to be replaced in the SPARQL syntax. In the example of Figure 1, the variable `?_year_number` generates the mandatory query parameter "year", that is meant to be replaced as a number before query execution. When an API is consumed, the variable in the query is substituted by the input value specified in the related query parameter. The output data format can be specified through content negotiation. The supported response formats are plain XML, JSON and CSV without namespaces, for data consumers that are not familiar with Linked Data, and Semantic Web Standards (such as, RDF+XML, N3 and Turtle), for SPARQL experts.

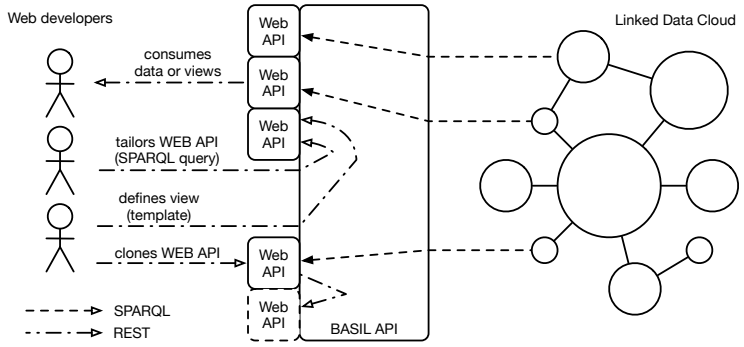

Fig. 2: The BASIL architecure

In addition, developers can specify *views* for each Web API. A view[3] is an alternative presentation of an API results based on a template or script. E.g., a view can be a HTML representation of the results ready to be embedded in a web site, with no further processing. The advantage of views is to adapt the output of a Web API to applications with special needs.

In order to support API integration, BASIL makes available an interactive documentation based on Swagger[4] for each API, which allow developers to test APIs before using them in applications. Specifications of APIs and their views are publicly accessible as subresources of the APIs to enable query analysis and reuse. Developers can also quickly create new APIs by cloning and customizing existing ones.

Different kinds of users can benefit from BASIL. SPARQL-aware developers can tailor Web APIs to support colleagues or business partners that are not expert in Semantic Web technologies, or simply because they think that its more handy to integrate a Web API in their application instead of embedding a SPARQL query. Users that are not SPARQL experienced can consume directly existing Web APIs and start learning by looking to existing queries or cloning and modifying APIs. Both type of users will benefit from using a Web API in applications instead of embedding the call to a SPARQL endpoint, in order to facilitate query maintenance and evolution.

Developers can access the platform functionalities through the BASIL API, a CRUD API over HTTP[5], or PESTO, a Web-based user interface[6]. BASIL source code is available on GitHub[7]. More details on BASIL are available in [1].

[3] Our definition of *view* is inherited from the popular Model-View-Controller (MVC) pattern.
[4] `https://github.com/swagger-api/swagger-spec`
[5] The interactive documentation is available at `http://basil.kmi.open.ac.uk/docs`
[6] Available at `http://basil.kmi.open.ac.uk`
[7] `https://github.com/the-open-university/basil`

3 Related Work

The Linked Data Platform[8] (LDP) is a W3C recommendation to perform CRUD operations on resources exposed as Linked Data. The specification enables consuming or modifying linked data resources through REST, by packaging a single Web API serving RDF data. However, LDP provides data as full RDF, and the specification does not recommend how to customise the data model. Approaches based on storing SPARQL queries on the server side have been proposed by the Linked Data API[9] specification, which have been implemented by ELDA[10] and Open PHACTS [2]. A similar facility is provided by The Data Tank[11], by defining a template language. As well as BASIL, both attempts hide the complexity of the SPARQL specification to the data consumer through a Web API. Nevertheless, the two approaches introduce additional formalisms, which highly increase the learning curve of potential adopters. The relation between Web services and Linked Data has been analysed in [3]. In this context, approaches to bridge the gap between services and linked data have been proposed. In [5], the authors propose a method to publish existing Web APIs as Linked Data. The same issue has been addressed by introducing functional descriptions of hypermedia services in [6]. Compared to [3, 5, 6], this paper addresses the opposite issue. BASIL exploits the benefits of Web APIs on top of SPARQL endpoints as simple and intuitive bridge between the Semantic Web and the Web developer communities.

References

1. Daga, E., Panziera, L., Pedrinaci, C.: A BASILar Approach for Building Web APIs on top of SPARQL Endpoints. In: Proc. of the Third Workshop on Services and Applications over Linked APIs and Data (SALAD) co-located with ESWC 2015 (2015)
2. Groth, P., Loizou, A., Gray, A.J., Goble, C., Harland, L., Pettifer, S.: API-centric Linked Data integration: The Open PHACTS Discovery Platform case study. Web Semantics: Science, Services and Agents on the WWW 29(0), 12 – 18 (2014)
3. Pedrinaci, C., Domingue, J.: Toward the next wave of services: Linked Services for the Web of data. Journal of Universal Computer Science 16(13), 1694–1719 (2010)
4. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I. pp. 245–260 (2014)
5. Speiser, S., Harth, A.: Integrating linked data and services with linked data services. In: The Semantic Web: Research and Applications, pp. 170–184. Springer (2011)
6. Verborgh, R., Steiner, T., Van Deursen, D., Coppens, S., Vallés, J.G., Van de Walle, R.: Functional descriptions as the bridge between hypermedia apis and the semantic web. In: Proc. of the WS-REST workshop. pp. 33–40. ACM (2012)

[8] http://www.w3.org/TR/ldp/
[9] https://github.com/UKGovLD/linked-data-api
[10] http://www.epimorphics.com/web/tools/elda.html
[11] http://docs.thedatatank.com/4.3/spectql