R4R: Template-based REST API Framework for RDF Knowledge Graphs

Carlos Badenes-Olmedo[0000−0002−2753−9917], Paola Espinoza-Arias[0000−0002−3938−2064], and Oscar Corcho[0000−0002−9260−0753]

Ontology Engineering Group, Universidad Politécnica de Madrid
Boadilla del Monte, Spain
{cbadenes, pespinoza, ocorcho}@fi.upm.es

Abstract. Knowledge graphs (KGs) are increasingly being used to make structured information available on the Web, by means of REST APIs and/or SPARQL endpoints. In many cases, these REST APIs are generated on top of the SPARQL endpoints, using existing technology approaches that are based on proprietary configuration files or ontologies to create the APIs. These approaches may impose content-based or structural constraints when composing Web resources. To relax these constraints we propose R4R, a more flexible solution based on Web standards and REST principles that creates and publishes customizable APIs exposing Web resources from SPARQL queries organized in file system directories. R4R features include individual and nested resources, paginated queries, optional fields, web authentication, query parameters and sorting lists.
Resource type: Software
License: Apache License 2.0
DOI: https://doi.org/10.5281/zenodo.3543320

Keywords: API · Knowledge Graph · REST · SPARQL

1 Introduction

Knowledge graphs (KGs) are drawing increasing attention from both academia and industry for representing, sharing and using knowledge in applications [7, 3]. They may be made available as RDF-based datasets, including a SPARQL endpoint (e.g., DBpedia), and/or via REST APIs (e.g., Google Knowledge Graph). In both cases, KGs share many commonalities from the data representation point of view (both use triples to represent facts), but they are radically different in terms of query capabilities: SPARQL provides a more expressive query language than what can be normally done with a REST API, but it can be a barrier for non-expert users. Web APIs usually present data according to REpresentative State Transfer (REST) architecture principles mapping HTTP verbs (POST, GET, PUT, DELETE) to CRUD operations (Create, Read, Update, Delete).

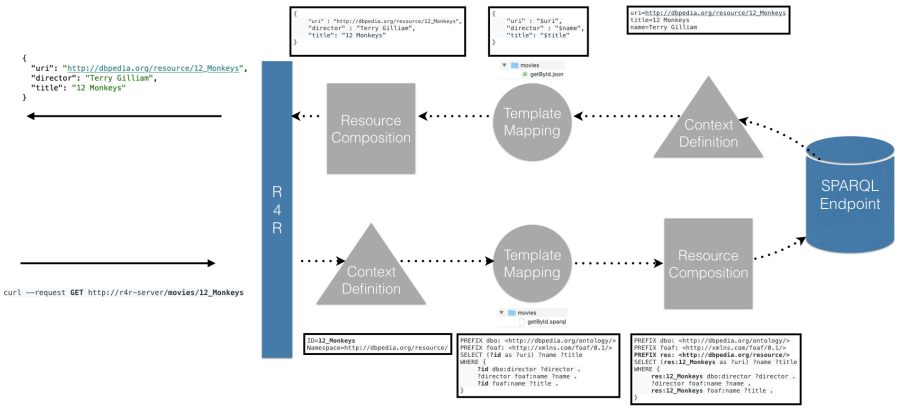

Fig. 1. Step-by-step processing of a request in R4R

However, API resources do not necessarily match to KG resources. A public procurement-focused API and a technology-focused API may present, in a different way, the information retrieved from the same KG about companies. One more focused on merit and the other on innovations.

In this demo we present R4R, an open source framework that facilitates the publication of a KG via a REST API over HTTP. Our approach proposes a fully customizable definition of resources, both naming and content (and even nesting), through a hierarchical organization of data (Figure 1). It deploys a web service based on SPARQL queries to retrieve the information and provides templates to compose resources that are organized in folders in a system directory. Finally, we describe a motivating example where R4R is used to enhance KG access.

2 KG data consumption via Web API

Several approaches are available to provide Web developers with mechanisms to ease KG data consumption without dealing with the complexity of Semantic Web standards and technologies, namely SPARQL. Some of these approaches have been focused on the provision of Web APIs that allow developers to interact with KG data. There are tools [1][5] that generate Web APIs from set of SPARQL queries but require setting up programming environments or use fixed structures for resources. Others [4] proposed a SPARQL Transformer to provide specific JSON structures from the SPARQL queries but this prevents validating SPARQL queries from any SPARQL endpoint, or [2] creates a Web service from the API paths, methods and SPARQL queries provided in a configuration file based on a key-value structure in proprietary format. In Table 1 we summarized the main features of the aforementioned approaches, and it also includes the features of R4R which will be described in the following sections.

Table 1. Web APIs tools for KGs consumption (✓= included, x = not included)

Characteristic	Proposal	BASIL[1]	grlc[5]	RAMOSE[2]	R4R
Resources	single	✓	✓	✓	✓
	multiple	✓	✓	✓	✓
	nested	x	x	x	✓
Methods	GET	✓	✓	✓	✓
	POST	✓	✓	✓	x
	PUT	x	x	x	x
	DELETE	x	x	x	x
Security	basic authentication	✓	✓	x	✓
Documentation	Swagger-compliant	✓	✓	x	✓
	static HTML	x	x	✓	✓
Parameters	filtering	x	x	✓	✓
	ordering	x	x	✓	✓
	pagination	x	✓	x	✓
Serialization	XML	✓	x	x	x
	CSV	✓	✓	✓	x
	JSON	✓	✓	✓	✓
	RDF	✓	✓	x	x
Content	JSON-oriented	x	✓	✓	✓
	SPARQL compatible	x	x	x	✓
Deployment	remote	✓	✓	✓	✓
	local	✓	✓	✓	✓
	isolated	x	✓	x	✓

3 R4R

The R4R[1] open source framework aims to bring it closer to non-expert Web service developers: (1) customizable resource abstraction and an (2) intuitive REST-based interface. The purpose of this tool is to facilitate access to KGs guided by use cases via a REST API. R4R supports **isolated deployment without the need to configure programming environments**. Unlike existing approaches, R4R provides **nested resources that let us reference complex objects**. Thanks to this feature, users can, for example, get the *characters* of a *movie* through an API call like */movies/{id}/characters* where the resulting characters will depend on the particular movie resource identified with the "{*id*}" value. It also maintains SPARQL compatibility by **avoiding any non-SPARQL variables in queries**, so that queries can be externally validated from any SPARQL endpoint. Our approach allows users to generate API documentation from a YAML file with a Swagger specification as well as with a **static HTML file, for those users not familiar with the Swagger specification**. In regard to parameters, our approach provides **filtering, ordering, and pagination options** which allows full flexibility when users have to deal with resources.

4 Motivating Example

We have prepared a short tutorial[1] to create a REST API over DBpedia that browses movies. Following an intuitive approach to the REST architectural style,

[1] https://github.com/oeg-upm/r4r

the characters of a movie, for example, are available thanks to a SPARQL-query (Listing 1.1) and a template (Listing 1.2) files located in a *resources/-movies/characters* folder. As a result, a JSON message with the list of characters of the movie is obtained by requesting, for example, */movies/WarGames/characters* (Listing 1.3).

Listing 1.1. SPARQL query to retrieve characters of a movie

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name ?birthDate ( ?starring AS ?uri )
WHERE {
        ?id dbo:starring ?starring .
        ?starring foaf:name ?name .
        ?starring dbo:birthDate ?birthDate .
        OPTIONAL {?name rdfs:label ?string . FILTER (lang(?string) = 'en') }
}
```

Listing 1.2. Template to return characters of a movie

```
[
    #foreach( $person in $results )
     {
        "uri" : "$person.uri",
        "name" : "$person.name",
        "birthDate" : "$person.birthDate"
     }
     #if ( $velocityCount < ${results.size()} )
        ,
     #end
    #end
]
```

Listing 1.3. Main characters of the movie *WarGames*

```
[
     {
        "uri" : "http://dbpedia.org/resource/John_Wood_(English_actor)",
        "name" : "John␣Wood",
        "birthDate" : "1930-07-05"
     }
    ,
              {
        "uri" : "http://dbpedia.org/resource/Ally_Sheedy",
        "name" : "Ally␣Sheedy",
        "birthDate" : "1962-06-13"
     }
    ,
     {
        "uri" : "http://dbpedia.org/resource/Matthew_Broderick",
        "name" : "Matthew␣Broderick",
        "birthDate" : "1962-03-21"
     }
    ,
     {
        "uri" : "http://dbpedia.org/resource/Dabney_Coleman",
        "name" : "Dabney␣Coleman",
        "birthDate" : "1932-01-03"
     }
]
```

5 Conclusions and Future Work

R4R facilitates the creation and publication of Web resources following REST principles through filesystem directories. Resource paths are constrained to follow a tree structure by automatically creating them from directories and avoiding manual editing. In addition, R4R eases KG consumption by providing a developer-friendly serialization of the SPARQL results without imposing custom configuration rules but using standard SPARQL and Velocity data structures. It has been adopted by TheyBuyForYou (TBFY) project [6] to facilitate access to its public procurement KG[2] with more than 150 million triples.

R4R has to be seen as a fully operational first step of building REST API over KG based on templates to create resources, SPARQL queries to retrieve data and directories to define resource paths. We plan to extend R4R in the future to enable write operations (POST, PUT, DELETE) to fully interact with the KG data and implement additional content negotiation capabilities and formats (JSON-LD, Turtle, HTML).

Acknowledgments

Work supported by *KnowledgeSpaces*, PID2020-118274RB-I00.

References

1. Daga, E., Panziera, L., Pedrinaci, C.: A BASILar approach for building web APIs on top of SPARQL endpoints. In: CEUR Workshop Proceedings. vol. 1359, pp. 22–32 (2015)
2. Daquino, M., Heibi, I., Peroni, S., Shotton, D.: Creating Restful APIs over SPARQL endpoints with RAMOSE. arXiv preprint arXiv:2007.16079 (2020)
3. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J.E.L., Kirrane, S., Neumaier, S., Polleres, A., Navigli, R., Ngomo, A.C.N., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge Graphs (2021)
4. Lisena, P., Meroño-Peñuela, A., Kuhn, T., Troncy, R.: Easy Web API Development with SPARQL Transformer. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) The Semantic Web – ISWC 2019. pp. 454–470. Springer International Publishing, Cham (2019)
5. Meroño-Peñuela, A., Hoekstra, R.: grlc makes GitHub taste like linked data APIs. In: European Semantic Web Conference. pp. 342–353. Springer (2016)
6. Soylu, A., Corcho, O., Elvesæter, B., Badenes-Olmedo, C., Martínez, F.Y., Kovacic, M., Posinkovic, M., Makgill, I., Taggart, C., Simperl, E., Lech, T.C., Roman, D.: Enhancing public procurement in the european union through constructing and exploiting an integrated knowledge graph. In: The Semantic Web – ISWC 2020. pp. 430–446. Springer International Publishing (2020)
7. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017). https://doi.org/10.1109/TKDE.2017.2754499

[2] https://github.com/TBFY/knowledge-graph-API