Towards semantic resource mashups

Lionel Médini, Pierre-Antoine Champin, Michaël Mrissa, Amélie Cordier

Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, F-69622, France
{firstname.lastname} -at- liris.cnrs.fr

Abstract. This paper presents an architectural framework to build semantic resource mashups that query, process and integrate data from business objects, Linked Open Data sources and Web APIs, as semantic resources. Such resources can be queried locally or distantly in an HTTP-like manner and exchange representations as RDF graphs. We illustrate this work with an application that mashes up sensor and actuator data for the Web of Things.

Categories and Subject Descriptors
E.1.2 [**DATA**]: Data structures – *Distributed data structures.*

General Terms
Design, Standardization.

Keywords
RDF, REST, RDF-REST, Backbone.js, framework, semantic mashup.

1 Introduction

The Web of services forms a graph in which nodes can produce and consume data. Each node can be regarded as a mashup that queries other nodes, processes the data locally and serves back enriched data. Semantic mashups are Web applications that rely on shared vocabularies to understand and enrich RDF graphs, such as DataConf [2], built on the Backbone.js[1] framework. Each node may also query data through one or several REST resources. The RDF-REST framework [1] exposes internal business objects as resources with a uniform HTTP-based interface and RDF representations. Semantic mashups can benefit from the RDF-REST framework by handling local and external resources using a common, HTTP-based semantic interface. Conversely, RDF-REST allows building applications that query distant resources, but does not provide a routing engine, nor perform resource aggregation.

[1] http://backbonejs.org/

In this paper, we propose an architecture integrating these two frameworks, based on the notion of *Semantic Resource Mashup* (SRM). SRMs are applications that can retrieve, transform and match graphs representing resources from different endpoints into a common graph representing an enriched resource. This paper is organized as follows: it presents the architecture of our framework and illustrates its use in a Web of Things application. We then conclude and draw perspectives of this work.

2 SRM Framework

We herein propose to combine RDF-REST and the DataConf mashup engine into an SRM framework. The framework aims at building applications that mash up **SemanticResource** objects that extend Backbone **Model** objects so that they conform to the RDF-REST resource interface. The architecture of an SRM (see **Fig. 1**) relies on the following domain-specific elements: i) a Backbone **router**[2] that provides to the SRM the ability to identify the routes corresponding to each type of known resource: depending on the hash part of the current URL, the router identifies the resource brokers (see below) that can provide descriptions or enrich the requested resource; ii) a **resource ontology** that describes the RDF types of all the resources that the application can handle in the Hydra vocabulary [3], as well as the existing object properties among these types; iii) **Resource brokers** that represent data sources that can be queried locally or over the Web to enrich the requested resource graphs. Each resource broker possesses a **ResourceBrokerDescription** (in the Web Intents[3] sense) indicating the resource types it can enrich, based on the resource ontology classes and properties. The router uses these descriptions to select the appropriate brokers for a given route; iv) implementations of the local semantic resources that represent the application business objects: these resources must fit the RDF-REST scheme (i.e. be queried using HTTP methods and exchange RDF graphs with their clients); v) a set of RDF-REST parsers and serializers that perform the inter-resource communication scheme of the RDF-REST framework: they are used by the SRM to transform the data structures managed by the resource brokers into RDF graphs and conversely "feed" the brokers with data corresponding to RDF resource representations; vi) a **SemanticResourceAggregator** that finally performs the mashup operation. Currently, it relies on a set of aggregation rules to provide the enriched resource representation: it links the RDF graph of the requested resource with the origin of each graph retrieved by the brokers, according to the properties stated in the rules (most commonly, an **owl:sameAs** property).

Each time an SRM application is queried about a particular **SemanticResource** instance using a particular HTTP verb, it uses its own router to determine the appropriate **Route** based on the resource type and available resource broker descriptions, using the hash part of the URI (as does Backbone's router). It then identifies the resource brokers that can enrich this resource and invokes them through the RDF-REST engine, regardless of the fact that the resource broker addresses a local or distant source. The

[2] In this work, we thus assume SRMs to be single-page, client-side applications.
[3] https://dvcs.w3.org/hg/web-intents/raw-file/tip/spec/Overview.html

resource and the resource brokers exchange RDF graphs through RDF-REST parsers and serializers. Each graph is mapped to the existing one using the local semantic resource aggregator that produces the complete graph of the enriched resource.

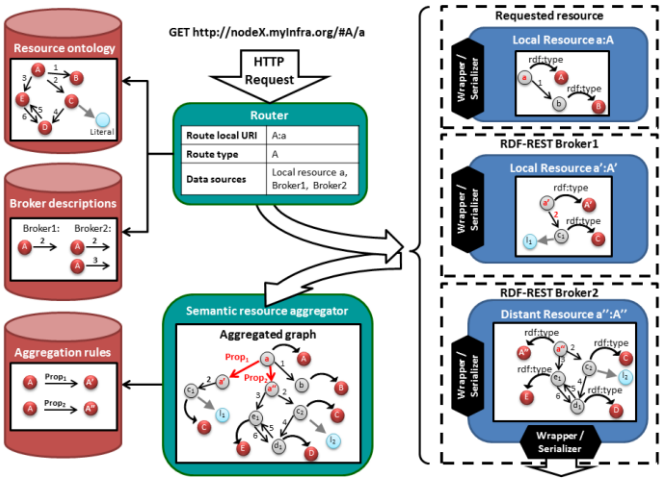

Fig. 1. Logical view of a Semantic Resource Mashup architecture.

3 Application example

We illustrate the applicability of our work in the domain of the Web of Things, using the concept of "avatar". Avatars are software artifacts that represent physical objects on the Web, expose object high-level behaviors (aka "functionalities") as REST resources and collaborate with other avatars [4]. When an avatar is requested for a functionality, it can use local object resources or functionalities exposed by other avatars. Let a user require temperature regulation in a smart house that includes a temperature sensor, a heater and an air conditioner. The avatar of the sensor exposes a functionality called **TemperatureRegulator** as a resource. When requested, it uses the following resources: **TemperatureSensor** (local), **TemperatureHeater** (distant) and **TemperatureDecreaser** (distant).

The avatar of the temperature sensor receives a POST request containing a graph that states that the temperature should be regulated to 22°C. The router identifies that the resource broker that handles the local **TemperatureRegulator** resource can handle this graph. It transmits the graph to the resource and its internal logics determine that it first needs to know the actual temperature. It sends a GET query to the local **TemperatureSensor** resource, which returns a graph informing that the temperature is 20°C. It then determines that the room should be heated and sends a PUT request to start the heater, which is handled by the avatar of the heater using the same architecture and returns a graph stating that the heater is on. The **TemperatureRegulator** resource then

returns to its client a graph stating that the thermoregulation is active, using the **TemperatureSensor** and **TemperatureHeater** resources, while internally continuing to regularly send requests to the sensor and heater resources. This work is funded by the French National Agency for Research (ANR), under the reference <ANR-13-INFR-012>.

4 Conclusion

In this paper, we present an architecture for Semantic Resource Mashups that relies on the RDF-REST and Backbone.js frameworks to process distributed RESTful resources. It mainly relies on: i) semantic resources that exchange RDF representations of business objects and expose uniform HTTP-compliant interfaces, ii) a resource ontology that describes the domain resource types and relations and iii) an RDF-based engine for routing and aggregating data flows. We illustrate our approach with a temperature regulation example for the Web of Things.

SRMs can be used to query RESTful endpoints such as RDF stores or to build a network of collaborative nodes. They also enable adaptive resource deployment on the server or client side without changing the application-level interactions between objects. Future work includes enabling SRMs to create, update, delete locally enriched resources, while being able to propagate these changes to appropriate endpoints. Another perspective is to replace the currently static broker descriptions and aggregation rules with dynamic discovery infrastructures such as RESTdesc [4], so that mashup applications can discover new data sources at runtime.

5 References

[1] Champin P-A. RDF-REST: A Unifying Framework for Web APIs and Linked Data. In Services and Applications over Linked APIs and Data (SALAD), workshop at ESWC, Montpellier (FR). pp. 10-19. CEUR 1056. (2013)

[2] Médini L., Bâcle F., Le Peutrec F. & Durant de la Pastellière B. DataConf and Its Linked Open Data Ecosystem: Produce, Link and Consume Scientific Conference Metadata. In LinkedUp Veni Competition on Linked and Open Data for Education, at Open Knowledge Conference 2013. M. d'Aquin, S. Dietze, H. Drachsler, M. Guy, E. Herder eds. Geneva. pp. 3-10. CEUR 1124. ISSN 1613-0076. (2014).

[3] Lanthaler, M., & Gütl, C. (2013). Hydra: A Vocabulary for Hypermedia-Driven Web APIs. In Proceedings of the 6th Workshop on Linked Data on the Web (LDOW2013) at the 22nd International World Wide Web Conf. (WWW2013).

[4] Mrissa M., Médini L., Jamont JP. Semantic Discovery and Invocation of Functionalities for the Web of Things. In IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, Parma, June 23-25 2014 (to be published).

[5] Verborgh, R., Steiner, T., Van Deursen, D., Coppens, S., Vallés, J. G., & Van de Walle, R. (2012, April). Functional descriptions as the bridge between hypermedia APIs and the Semantic Web. In Proceedings of the Third International Workshop on RESTful Design (pp. 33-40). ACM.