Resource Description Graph Views for Configuring Linked Data Visualizations

Bettina Steger[1], Thomas Kurz[2], and Sebastian Schaffert[2]

[1] Fachhochschule Salzburg, Campus Urstein Süd 1, 5412 Puch/Salzburg, Austria
bsteger.mmt-m2011@fh-salzburg.ac.at
[2] Salzburg Research, Jakob-Haringer-Straße 5/II, 5020 Salzburg, Austria
firstname.lastname@salzburgresearch.at

Abstract. The Linked Data movement with the aims of publishing and interconnecting machine readable data has originated in the last decade. Although the set of (open) data sources is rapidly growing, the visualization of information in this 'Web of Data' is still at a very early stage, which is primary due to the strong learning curve of semantic technologies. This paper describes an approach to visualize data 'ready-to-go' by configuration that enables Web developers and designers to build useful applications on top of the 'Web of Data'. We provide a visualization tool as a JavaScript Library, which makes it simple to aggregate Linked Data and design templates. The tool provides a way to accomplish this purely on the client using existing Web technologies, like JavaScript MVC Frameworks with data binding and JSON-LD. Based on a usability test, an evaluation is carried out by potential users, such as Web developers and semantic Web experts.

Keywords: Linked Data, visualization, client-side, json-ld, data-binding

1 Introduction

After efficiently encouraging the publication and linking of open datasets in a standardized way, the Linked Data (LD) research community is now facing the problem of creating meaningful applications on top of the Linked Open Data (LOD) Cloud. As Heath discussed in [2], the aim of these cloud interfaces is to give 'things', in the broadest sense, a central role and treat them as first-class citizens in the Web. The graph structure of the LOD Cloud, one of the big potentials regarding dynamics and distribution, makes this task particularly challenging. In many cases, complex graphs (that include various resources spread on different datasets and using several schemas and ontologies) are required in order to create a meaningful picture of a 'thing'. The gap between the understanding of the complex structures and the creation of human understandable representation makes the creation of LD User Interfaces (UI) even more difficult. In our demo, we present *visuaLOD*[3], a browser-based library that allows to split the

[3] https://bitbucket.org/visualod/visualod.bitbucket.org

work flow of the creation of LD applications in a Semantic Web (SW) and a UI part. This enables SW experts and Web designers to work closely together. *visuaLOD* thereby turns complex graph structures into configurable object models, using well-known technologies such as JavaScript (JS) and JSON. The interfaces themselves are built employing Google's AngularJS Framework[4] to accomplish data binding to the view. With *visuaLOD* we try to reach a linear dependency between comfort in usage and application complexity.

In this paper we will refer to two different types of Web developers: Semantic Web experts are developers having had experience with SW and advanced technologies like RDF or SPARQL. In contrast, non-expert developers do not have any relation to SW, but do know how to build Web applications with e.g. PHP, Java, JavaScript. There are already different ways to create LD visualizations. The approach presented in this paper, however, also enables non-semantic-experts to work with Linked Open Data.

2 visuaLOD - a Linked Data View Builder

Different resources need different UIs, e.g. a person could be displayed with a profile page, whereas a place should be displayed with a map. Depending on type, properties and source, presentation and actions may differ. In our demo[5], we present a simple movie mashup application visualizing movie details and starring actors. Information about the actors, e.g. their birth places (*geonames*), can be explored by following links to other datasets. To provide a flexible visualization tool, we elaborated simple configuration and abstraction of complex data structures as basic requirements.

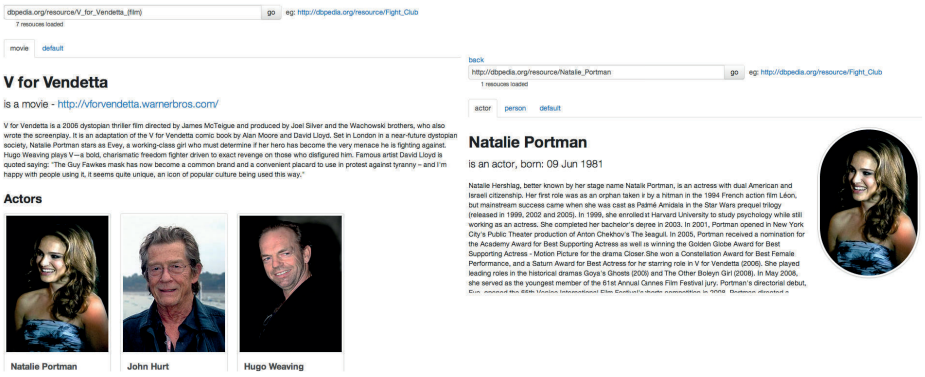

Fig. 1. Movie template is displayed because of resource's type. In addition, starring actors are fetched. Clicking on an actor's image changes the view to the right actor template. An actor's resource matches 3 different views: 'actor', 'person' and 'default'.

[4] http://angularjs.org
[5] http://visualod.bitbucket.org

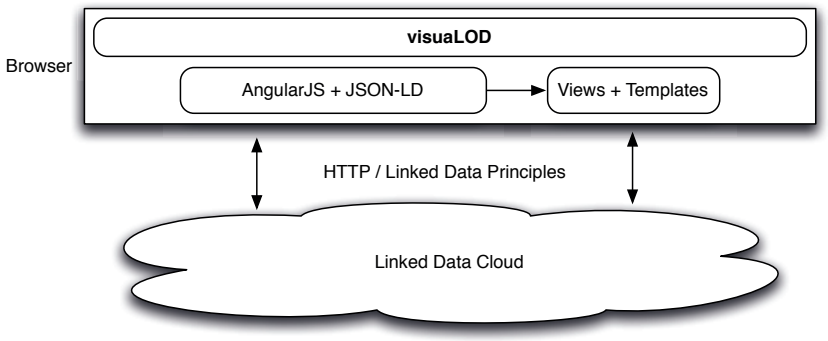

Fig. 2. Architecture: *visuaLOD* runs on the client, no need for server installation.

2.1 Resource Description Graph

Following the nature of Linked Data, the entry point of *visuaLOD* applications is a single LD resource. Taking this as a starting point, we follow dedicated links to fetch the part of the graph necessary for the information representation. We call the graph of resources and relations, needed to sufficiently visualize a 'thing' a 'Resource Description Graph' (RDG). RDGs are defined in so-called RDG views that include constraints (if a RDG is used for a specific resource), data mapping (how a graph is represented on client side) and a template (how a RDG is displayed).

The data mapping is managed with a JSON-LD Context. JSON-LD[6] is a lightweight LD serialization format based on JSON. Any RDF representation of a LD resource can be transferred into JSON-LD and vice versa. All defined properties in this **context** can be used in the template.

2.2 Work flow

The process, described in Figure 3, retrieves RDF data for the starting resource and maps it to the AngularJS model. It applies RDG views by validating their constraint part against the RDF data, fetches the additional resources and nests them into the model object. In AngularJS, the view is a projection of the model through the HTML template, so the templates are rendered in parallel.

3 Evaluation

A usability test was completed by six potential users: one SW expert and five Web developers with no prior knowledge of SW. Every user understood the given assignment and purpose of visuaLOD. The tool provides simple usage, but when

[6] http://json-ld.org/

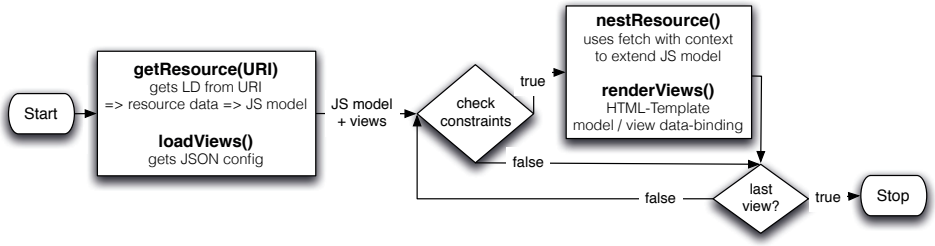

Fig. 3. Work flow

it comes to defining the JSON-LD Context the five non-semantic-Web-experts had difficulties. Feedback received from the SW expert suggests an easier usage to define the context. For example, visuaLOD could determine automatically if a property is a URI or a literal. The README[7] could be extended by screenshots and better examples on how to use visuaLOD.

4 Related Work

Fresnel[8] is a browser-independent presentation vocabulary for RDF. The main concept consists of two parts: *Lenses* and *Formats*. Lenses specify which properties of RDF resources are shown while formats indicate how to format content selected by lenses [4]. *Lenses and Formates* are defined in RDF, thus it is difficult to read and write a lens or format for non-experts.

LODSPeaKr[9] is a framework to create LD applications. It recommends to discover the data of the defined SPARQL endpoint, which means it is not suitable for all LD sources.

LESS [1] represents an approach for the visual presentation of LD resources and SPARQL query results. The process is based on the server side and uses a flexible, but proprietary templating language.

The KiWi project [3, 5] introduces perspective concepts allowing type-dependent visualization patterns. The weakness of the approach was mainly the strong coupling between back end and visualization. Nevertheless, the idea of KiWi perspectives lead to the visualization tool we presented in this demo.

All approaches introduced in the course of this chapter differ from the presented JS visualization tool: visuaLOD is purely browser-based. A server-side configuration is not required. Beyond JSON-LD works with any Linked Data server (e.g. RDF/XML can be converted to this serialization format).

[7] http://bitbucket.org/visualod/visualod.bitbucket.org
[8] http://www.w3.org/2005/04/fresnel-info/
[9] http://lodspeakr.org/

5 Conclusion and Further Work

This paper has sought to introduce a new approach to visualize Linked Data fully client-side. With *visuaLOD*, we presented a LD visualization tool that enables even non-semantic Web experts to build LD visualizations. To extend *visuaLOD* to a generic Linked Data Browser, we will provide a JS bookmarklet. It could be possible to allow the creation and storage of RDG views in an open accessible *view store*. In addition, future work could focus on the following aspects:

- Multi-language support: Detect the browsers language and show the end-users available texts in their language.
- Update: Not only read, but update LD resources using the advantages of data-binding and e.g. SPARQL update[10].
- View/Template builder: Create a UI for building views. View-Changes could automatically show how a visualization will look like.

References

1. S. Auer, R. Doehring, and S. Dietzold. LESS - template-based syndication and presentation of linked data. In *Proceedings of the 7th international conference on The Semantic Web: research and Applications - Volume Part II*, ESWC'10, pages 211–224, Berlin, Heidelberg, 2010. Springer-Verlag.
2. T. Heath. How Will We Interact with the Web of Data? *IEEE Internet Computing*, 12(5):88–91, Sept. 2008.
3. T. Kurz, S. Schaffert, T. Bürger, S. Stroka, and R. Sint. KiWi - A Platform for building Semantic Social Media Applications. In *Proceedings of the ISWC 2010 Posters and Demonstrations Track*, ISWC'10, pages 185–188, 2010.
4. E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel: a browser-independent presentation vocabulary for RDF. In *Proceedings of the 5th international conference on The Semantic Web*, ISWC'06, pages 158–171, Berlin, Heidelberg, 2006. Springer-Verlag.
5. S. Schaffert, J. Eder, S. Grünwald, T. Kurz, and M. Radulescu. Kiwi - a platform for semantic social software (demonstration). In *ESWC'09: The Semantic Web: Research and Applications, Proceedings of the 6th European Semantic Web Conference*, pages 888–892, Heraklion, Greece, June 2009.

[10] http://www.w3.org/TR/sparql11-update/