Journal of Visual Languages and Computing 25 (2014) 738–744

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Short Paper

SWOWS and dynamic queries to build browsing applications on linked data ☆

Paolo Bottoni *, Miguel Ceriani *

Sapienza, University of Rome, Department of Computer Science, Italy

ARTICLE INFO

Article history:
Received 26 September 2014
Received in revised form
21 October 2014
Accepted 24 October 2014
Available online 31 October 2014

Keywords:
Linked data
Semantic Web
Declarative programming
Visual programming languages
Visual exploration
SPARQL
Semantic information retrieval
Interactive applications

ABSTRACT

The linked data initiative is pushing dataset maintainers to publish data online in a highly reusable way through a set of open standards, such as RDF and SPARQL. The variety and amount of structured data available on the Web are increasing, but their consumption is still quite limited. In particular, applications that are used to explore linked data are usually generic linked data browsers or applications with hard-coded logic tailored for specific needs. SWOWS is a platform for declarative specification of applications consuming linked data. In this paper, we describe the use of the platform for creating browsing applications tailored to specific contexts, and show how the declarative paradigm supports the development of flexible applications. To this end, the platform has been extended to support the dynamic generation of SPARQL queries. An example of a linked data browser created with the platform is given.

1. Introduction

Over the past few years the Web has been evolving from an interlinked set of documents to an interlinked Web of data and services. The structured data available online are increasing both in quantity and diversity [7], thanks in part to the comprehensive data model proposed by the World Wide Web Consortium (W3C): the Resource Description Framework (RDF) [13].

A key advantage of the linked data model is its support for serendipitous exploration and reuse of existing data. Potentially, any expert of a specific domain can build a fully customised visualisation from a set of linked data sources. In practice, building such visualisations currently requires advanced programming skills and involves a number of system choices that constrain the effective reuse of a visualisation. To mitigate this problem, we have developed the Semantic Web Open datafloW System (SWOWS) [10,9], a platform that allows declarative construction of interactive linked data applications. Applications are built from a set of operators (based on SPARQL, the standard query language over the RDF model [19]) adopting the pipeline metaphor.

This paper will describe the use of SWOWS to build applications to browse specific sets of linked data. The purpose is to show how such applications can be built purely on Web and Semantic Web standard technologies and how the declarative approach can help in organising flexible visualisations.

The SWOWS platform has been extended to allow manipulation of SPARQL queries as data. A developer using SWOWS can write SPARQL queries that take other SPARQL queries as input and produce new queries as output. The queries written with SWOWS can thus play a role similar to the role *higher-order functions* have in other programming languages [25]. The use of this functionality (together with other extensions to the original SWOWS platform) is showcased in the paper through an example application.

☆ This paper has been recommended for acceptance by S.-K. Chang.

* Corresponding authors.
E-mail addresses: bottoni@di.uniroma1.it (P. Bottoni), ceriani@di.uniroma1.it (M. Ceriani).

http://dx.doi.org/10.1016/j.jvlc.2014.10.027
1045-926X/

Section 2 of this paper introduces the technological background. Section 3 discusses related work and Section 4 specifically describes SWOWS. New features are shown in Sections 5 and 6, where dynamic query generation is illustrated. An example application is described in Section 7. Section 8 discusses conclusions and future work.

2. Technological background

The relational model is widely used to represent virtually any kind of structured information. The *Resource Description Framework (RDF)* [13] generalises the relational model to the universe of structured data in the World Wide Web, better known as the Semantic Web [5]. In the RDF model, knowledge is represented via *RDF statements* about *resources* – where a resource is an abstraction of any piece of information about some domain. An RDF statement is represented by an *RDF triple*, composed of *subject* (a resource), *predicate* (specified by a resource as well), and *object* (a resource or a literal, i.e. a value from a basic type). An *RDF graph* is therefore a set of RDF triples. Resources are uniquely identified by a Uniform Resource Identifier (URI) [4], or by a local (to the RDF graph) identifier if they do not have a meaning outside of the local context (in which case they are called *blank nodes*). The resources used to specify predicates are called *properties*. A resource may have one or more *types*, specified by the predefined property `rdf:type`. An *RDF dataset* is a set of graphs, each associated with a different name (a URI), along with a default graph without a name. We use RDF throughout the framework to represent any kind of information, as well as its transformations. In RDF, *prefixes* can be used in place of the initial part of a URI, which represent specific namespaces for vocabularies or sets of resources.

We extensively use *SPARQL*[1] [19], the standard query language for RDF datasets. SPARQL has relational algebra semantics, analogous to those of traditional relational languages such as Structured Query Language (SQL). The SPARQL CONSTRUCT, one of the SPARQL query forms, takes an RDF dataset as input and produces an RDF graph. While the SPARQL Query Language is "read-only", the SPARQL Update Language [29] defines a way to perform updates on a *Graph Store* – the "modifiable" version of an RDF dataset. A SPARQL Update *request* is composed of a number of *operations*. The current version of the standard is SPARQL 1.1, but much of the existing work refers to the previous version, SPARQL 1.0 [28]. The SPARQL 1.1 algebra offers an expanded set of operators, effectively allowing the expression of queries that were not expressible before.

The ubiquity of Web browsers and Web document formats across a range of platforms and devices drives developers to build applications on the Web and its standards. Requirements for browsers have dramatically changed from the first days of the Web. Now a browser is an interface to an ever-growing set of client capabilities, exemplified by the *Rich Web Client* activity at W3C.[2] All modern browsers support the Scalable Vector Graphics (SVG) standard [14], a language representing mixed vector and raster content, based on Extensible Markup Language (XML) [11]. Together with the long established Document Object Model (DOM) events [26] and ECMAScript [1] support, SVG allows the realisation of complete interactive visualisation applications. Indeed, ECMAScript libraries for interactive data visualisation are proliferating, from standard visualisations [17,3,8] to specialised ones for specific domains [32], especially leveraging the SVG technology.

3. Related work

Several languages were proposed to define *SPARQL views* in a way analogous to SQL views. A SPARQL view is a graph intensionally defined by a SPARQL CONSTRUCT query; the input dataset can be composed of "real" graphs (i.e. extensionally defined) and views. *RVL* [23] is an early effort, using an imperative language for defining views based on an independently defined query language (RQL [20]). *vSPARQL* [31] extends the SPARQL 1.0 grammar by allowing named views defined with CONSTRUCT queries. Schenk and Staab, working on *Networked Graphs* [30], propose an RDF-based syntax to define views, also based on SPARQL 1.0 CONSTRUCT queries. Although powerful enough to define read-only applications (possibly together with visualisation tools described below), networks of views do not easily model interactive applications. In particular, they face the problem of how to represent events and time-dependent information, including the application state.

Two *pipeline languages* have been proposed to define RDF transformations: DERI Pipes [22] and SPARQLMotion [21]. These languages offer a set of basic operators on RDF graphs to build the pipelines and they are both endowed with a graphical environment to create the pipelines using the available operators (free in the case of DERI Pipes, in a commercial software for SPARQLMotion [12]). In SPARQLMotion, both pipelines and queries are represented in RDF (for queries using the SPIN-SPARQL [16] syntax).

Visualbox [18] and Callimachus [2] have been proposed for *linked data visualisation*. In their two-step model/view approach, SPARQL queries select data and then a template language generates the (XML-based) visualisation. SPARQL Web Pages (SWP) [21] is an RDF-based framework (to be used with SPARQLMotion or on its own) used to describe and render HTML+SVG visualisations of linked data. HTML and SVG are mapped to two vocabularies and, together with the UISPIN Core Vocabulary, allow the association of an RDF resource with the description of its visualisation. The description may also be statically associated with a class of resources with each specific resource mapping defined by a SPARQL query. In all these proposals, the execution model is defined for a single HTTP request, as with typical application server technologies like Java Servlet or PHP. Persistence and logical relations between requests and client states must be managed explicitly (e.g. saving/loading session data and encoding parameters in requests).[3]

[1] Originally a recursive acronym SPARQL Protocol and RDF Query Language, the extended form has then been dropped from W3C documents.

[2] http://www.w3.org/2006/rwc/Activity.html

[3] Both Callimachus and SWP offer some aid for building interactive applications via special functions and syntaxes, but the execution model remains request oriented.