Jassa - A JavaScript suite for SPARQL-based faceted search

Claus Stadler, Patrick Westphal, Jens Lehmann

Universität Leipzig, Institut für Informatik, AKSW,
{cstadler|pwestphal|lehmann}@informatik.uni-leipzig.de
http://aksw.org

Abstract. The availability of SPARQL endpoints on the Web provides interesting opportunities for rapid web application development. However, sophisticated applications need components that can adopt to the data, yet, the efficient generic exploration and visualization of data contained in those endpoints is still challenging. In this paper, we present the "JAvascript Suite for Sparql Access" (Jassa) framework, which features a modular architecture for reusable components. The current highlights comprise APIs for RDF, SPARQL, data access and faceted browsing, together with corresponding user interface components based on AngularJS.

1 Introduction

The Linked Data initiative led to a paradigm shift, in which large amounts of structured data were made publicly available. With RDF, there is now a data model, which enables global identification and integration of resources as well as cross-dataset interlinking. On top of the RDF data model, SPARQL[1] became a standard language for accessing web databases. Yet, the development of Web applications for the exploration and visualization of SPARQL-accessible data on the Web still poses several challenges related to performance and design. In this paper, we present the Open Source JavaScript framework called *JAvaScript Suite for Sparql Access*.

The remainder of this paper is structured as follows. In Section 2 we outline the high-level structure of Jassa as well as its highlights. In Section 3 we briefly summarize related work. Finally, in Section 4 we conclude this paper.

2 The Jassa Architecture

Jassa is an umbrella term for a set of three related projects. A depiction of the architecture is shown in Figure 1. These projects are summarized as follows:

[1] http://www.w3.org/TR/sparql11-query/

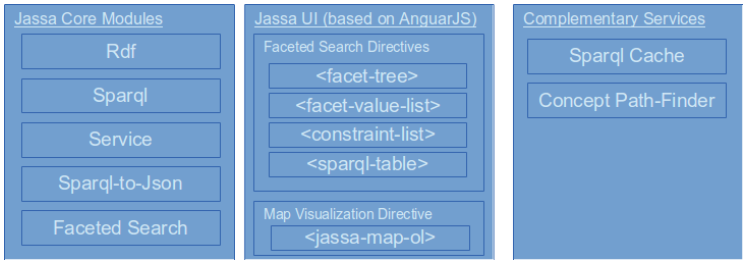


Fig. 1: The Jassa Architecture

- *Jassa Core*[2] is a project providing a layered set of APIs, ranging from APIs for the representation of RDF and SPARQL, an API for executing queries, a SPARQL-to-JSON mapper, and most prominently, the faceted search API.
- *Jassa UI*[3] is a project for user interface components based on Jassa Core and the AngularJS[4] framework.
- *Complementary Services for Jassa*[5] is a Java project that offers server side APIs that enhance Jassa, such as a SPARQL cache proxy. A more sophisticated service is the one for finding property paths connecting concepts.

In the following, we explain selected components of Jassa in a bottom-up fashion, starting from the RDF API and ending in the faceted search widgets.

2.1 The RDF and SPARQL APIs

The RDF module features the basic components for working with RDF data. The most important class is the *rdf.NodeFactory* used to create instances of *rdf.Node*, which in turn represent RDF terms. Although plain JSON could be used to represent RDF terms, we argue that such a 'strict' API is less prone to common errors and makes operations more succinct. The SPARQL module features classes for representing the syntactic constructs of SPARQL queries.

```
var s = rdf.NodeFactory.createVariable('s');
var t = new rdf.Triple(s, s, s);
```

Listing 1: Example of the RDF API

```
var query = new sparql.Query();
query.setResultStar(true);
query.setQueryPattern(new sparql.ElementTriplesBlock([t]));
query.setLimit(10)
console.log('As string: ' + query); // Output: Select * { ?s ?s ?s } Limit 10
```

Listing 2: Example of the SPARQL API

[2] https://github.com/GeoKnow/Jassa
[3] https://github.com/GeoKnow/Jassa-UI-Angular
[4] https://angularjs.org/
[5] https://github.com/AKSW/jena-sparql-api

Note that all mentioned 'namespaces', such as *rdf* and *sparql* actually reside in the *jassa* parent object.

2.2 The Sparql Service API

Web applications can run SPARQL queries against HTTP SPARQL endpoints by directly using an AJAX API, such as those provided by jQuery[6] or AngularJS. However, this simple approach has several drawbacks: the server may limit the sizes of result sets, leaving the client with unexpected incomplete results. Triple stores may have limited or even erroneous SPARQL implementations, forcing clients to phrasing their queries differently. In many cases, caching query responses is desired. On some occasions it can happen that an application launches a new query although the same query is already running. Jassa provides classes to *transparently* deal with all of these issues.

The main interface is *service.SparqlService* which defines methods for creating *service.QueryExecution* objects from a query string or query object. The *sparql.QueryExecution* class then offers the methods for retrieving the response: execSelect(), execConstruct(), exectDescribe() and exectAsk(). These methods yield promises of appropriate type, i.e. *sparql.ResultSet*, *rdf.Graph* and boolean.

The example below demonstrates the creation of a SPARQL service that transparently performs pagination, query transformations for some issues with Virtuoso[7] and caching of the individual pages.

```
var sparqlService = new service.SparqlServiceHttp(
    'http://dbpedia.org/sparql', ['http://dpbedia.org']);
sparqlService = new service.SparqlServiceVirtFix(sparqlService);
sparqlService = new service.SparqlServiceCache(sparqlService);
sparqlService = new service.SparqlServicePaginate(sparqlService, 1000)
var qe = sparqlService
            .createQueryExecution('Select * { ?s ?p ?o} Limit 10000');
qe.execSelect().then(function(rs) {
    while(rs.hasNext()) {
        var binding = rs.nextBinding();
        binding.get(rdf.NodeFactory.createVariable('s'));
    }
});
```

Listing 3: Usage example of the SparqlService API

We are currently experimenting with the integration of rdfstore-js's[8] parser into Jassa in order to add support for transparent conversions of query strings into query objects.

[6] `http://jquery.com/`

[7] For example, Virtuoso by default does not support queries having an outer-most OFFSET greater than 20000 in conjunction with an ORDER BY clause. Also, Virtuoso yields incorrect counts for queries having a LIMIT if there is a sub-query having a LIMIT as well. SparqlServiceVirtFix rephrases such queries in an attempt to mitigate these issues.

[8] `https://github.com/antoniogarrote/rdfstore-js`

2.3 The SPARQL-to-JSON mapper

Between SPARQL and JSON (or JavaScript objects in general) there is an impedance mismatch similar to that encountered in the object/relational world: SPARQL result sets are relations, however these are of little direct use in JavaScript applications without a transformation into *objects*. In general, object creation requires aggregation of data from multiple result set rows. The name Sponate is derived from the technologies by which it is inspired, namely SParql, jsON, and hibernATE.

Sponate uses *maps* to express the SPARQL-to-JSON mappings, and it supports initial query capabilities over the mapped objects using an interface similar to that of the JSON database MongoDB[9].

A usage example of Sponate is shown in the listing below.

```
var store = new sponate.StoreFacade(sparqlService, prefixMap);
store.addMap({
  template: [{
    id: '?s', name: '?l',
    partners: [{
      id: '?f', name: '?pl', amount: '?a',
    }]
  }],
  from: '?s a o:Project ; rdfs:label ?l ; o:funding ?f . ?f o:partner [ rdfs:
      label ?pl ] . ?f o:amount ?a' });
});
var criteria = {partners: {$elemMatch: {name : {$regex: 'university'}}}};
store.projects.find(criteria).limit(10).asList().done(function(arr) {
  // arr is an array of JavaScript objects according to the JSON template
})
```

2.4 The LookupService API

Given a set of URIs, Jassa makes retrieval of related information easy using the *LookupService* interface and its corresponding implementations. The only method on this interface is *Promise<Map> lookup(keys)*. The API is similar to that of the sparql service: Functionality is enhanced using consecutive wrapping. In this example, the basic lookup service is based on a sponate store.

```
var ls = new service.LookupServiceSponate(store.projects);
ls = new service.LookupServiceCache(ls);
// Limit to 20 keys per request - avoids large SPARQL queries
ls = new service.LookupServicePartition(ls, 20);
ls = new service.LookupServiceKeyFilter(ls, predicateForValidatingUris);
// Merge lookup requests that occur within a 50ms time window
ls = new service.LookupServiceTimeout(ls, 50);
ls.lookup([ /* rdf.Node objects for the lookup */ ]).then(function(map) {
  /* Use map.get(key) to retrieve the key's value */
});
```

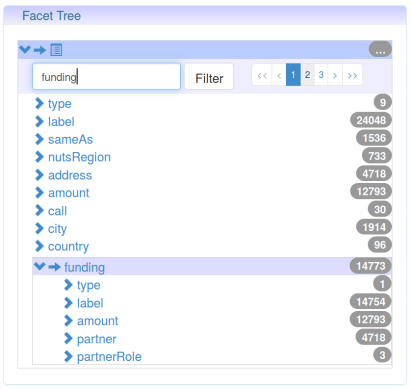
2.5 The Faceted Browsing Widgets

The widgets are provided as own AngularJS directives to be as easy to use as possible. Accordingly, to reuse our UI components only small pseudo-HTML

[9] http://docs.mongodb.org/manual/reference/operator/nav-query/

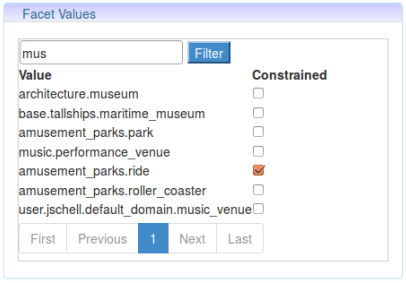
snippets have to be added to the overall HTML as presented in the following. Jassa comes with a highly configurable SPARQL-based faceted search API in the *facete* namespace, which supports the definition of custom constraint types as well as constraining sets of resources by indirectly (possibly inversely) related properties. Due to space limitations, we only demonstrate the usage of the faceted browsing widgets. Only the following code is needed to set up a generic faceted browser on a given SPARQL service object. Note that the widgets are synchronized via the state of the fctTreeConfig object; any modification to this object will automatically trigger an update of the widgets.

```
$scope.selectedPath =
  sparql.PathUtils.parse('http://www.w3.org/1999/02/22-rdf-syntax-ns#type');
$scope.fctTreeConfig = new facete.FacetTreeConfig();
```




```
<facet-tree
  sparql-service="sparqlService"
  facet-tree-config="fctTreeConfig"
  select="selectFacet(path)"
></facet-tree>
```

(a) Facet Tree




```
<facet-value-list
  sparql-service="sparqlService"
  facet-tree-config="fctTreeConfig"
  path="selectedPath"
></facet-value-list>
```

(b) Facet Value List

3 Related Work

A generic JavaScript-based RDF data browser called Tabulator [3] was developed under the umbrella of the World Wide Web Consortium. Besides the tree based traversing the tool also provides a map and a calendar view. Another library providing RDF access in JavaScript is RDFQuery[10], which provides an API for manipulation and querying of RDF data within JavaScript as well as the extraction of RDF data from Web content. However, neither of these projects seem to offer the powerful abstractions and implementations provided by Jassa. As for RDF JavaScript APIs, there are recent efforts in re-continuing work on

[10] http://code.google.com/p/rdfquery/

a specification on RDF interfaces in JavaScript[11]. In regard to faceted browsing of RDF datasets, there are for instance the Sparklis browser[12] and the Pelorus faceted navigation tool[13]. Jassa however features support for nested and inverse properties and offers reusable components. Very recent development efforts which provide similar features are [2] and [1].

4 Conclusions and Future Work

In this software description, we explained the components of the *JAvascript Suite for Sparql Access* (Jassa). Its foundation is built on a *core* library, which includes an RDF, SPARQL API, advanced service abstractions (e.g. transparent caching, query transformation and pagination) and a faceted search module. The *jassa-ui* modules introduce user interface components for faceted search and map display. Thanks to AngularJS, these widgets can be embedded as ordinary HTML elements in websites. Overall, Jassa simplifies Semantic Web application development via light-weight but powerful and efficient APIs. In the future, Sponate's feature set will be extended, such as with support for references between maps. We are also working on new facet retrieval strategies which may result in performance improvements. Examples of applications built using Jassa include the generic SPARQL browser *Facete2*[14] and the DBpedia Linked Data Viewer[15].

Acknowledgment

This work was supported by grants from the EU's 7th Framework Programme provided for the projects LOD2 (GA no. 257943) and GeoKnow (GA no. 318159).

References

1. M. Arenas, B. Cuenca Grau, E. Kharlamov, Š. Marciuška, D. Zheleznyakov, and E. Jimenez-Ruiz. Semfacet: Semantic faceted search over yago. In C.-W. Chung, A. Z. Broder, K. Shim, and T. Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, pages 123–126, New York, NY, USA, 2014. ACM Press.
2. H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. Easy access to the freebase dataset. In C.-W. Chung, A. Z. Broder, K. Shim, and T. Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, pages 95–98, New York, NY, USA, 2014. ACM Press.
3. T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *The 3rd International Semantic Web User Interaction Workshop*, 2006.

[11] `http://www.w3.org/TR/rdf-interfaces/`
[12] `http://www.irisa.fr/LIS/ferre/sparklis/osparklis.html`
[13] `http://clarkparsia.com/pelorus/`
[14] `http://facete.aksw.org`
[15] `http://ldv.dbpedia.org/#/ontology/Actor`