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 various abstractions and utilities for the creation and transformation of SPARQL queries, as well as the processing of corresponding result sets. The current highlights comprise modules for RDF, SPARQL, data access, conversion of result sets to objects 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, a data model was introduced, which enables global identification and integration of resources as well as cross-dataset interlinking. On top of RDF, 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 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* (Jassa). Jassa is motivated by the goal of creating the generic widgets for faceted browsing of RDF data depicted in Figure 2. The library is designed to tackle many of the challenges encountered during the development process.

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/

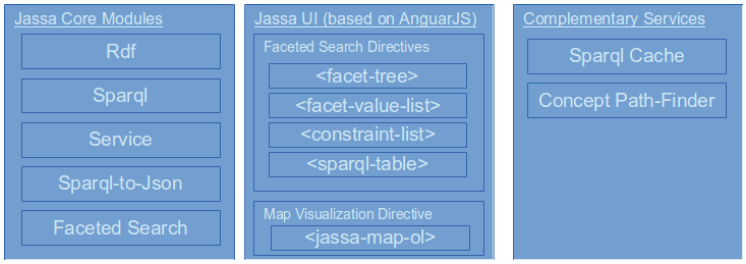


Fig. 1: The Jassa Architecture

- *Jassa Core*[2] is a project that provides a layered set of modules for: the representation of RDF and SPARQL, the execution of queries, SPARQL-to-JSON mapping, and most prominently, faceted search.
- *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. Another service is capable of finding property paths connecting two sets of resources.

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

2.1 The RDF, VOCAB and SPARQL modules

The *rdf* module features the basic components for working with RDF data. The most important class is *rdf.NodeFactory*, from which *rdf.Node* objects can be created that represent RDF terms. The *vocab* module provides predefined node objects for the commonly used xsd, rdf, rdfs and owl vocabularies. The *sparql* module features classes for representing the syntactic constructs of SPARQL queries.

```
1  var s = rdf.NodeFactory.createVar('s');
2  var o = rdf.NodeFactory.createUri('http://dbpedia.org/ontology/Airport');
3  var t = new rdf.Triple(s, vocab.rdf.type, o);
4
5  var query = new sparql.Query();
6  query.setResultStar(true);
7  query.setQueryPattern(new sparql.ElementTriplesBlock([t]));
8  query.setLimit(10)
9  console.log('As string: ' + query);
10 // Output: Select * { ?s a <http://dbpedia.org/ontology/Airport> } Limit 10
```

Listing 1: Example for forming a query

Note, that all mentioned namespaces reside in the global *jassa* object. As can be seen from Listing 1, we decided to follow designs of the well-known Apache Jena

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

project[6], with the intention of providing a consistent development experience when combining these frameworks in a project.

2.2 The Sparql Service API

Web applications could run SPARQL queries against HTTP SPARQL endpoints by directly using an AJAX API. However, this simple approach has several drawbacks: The server may limit the sizes of result sets. Triple stores may have limited or even erroneous SPARQL implementations, forcing clients to phrase 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 a decorator-based approach to *transparently* deal with these issues. The main interface is *service.SparqlService*, which defines methods for creating *service.QueryExecution* objects from a query string[7] or query object. Usually, decorators that apply query transformations will first parse the string before passing the transformation result as an object to the delegate. The *sparql.QueryExecution* class offers methods for retrieving the response: execSelect(), execConstruct(), exectDescribe() and exectAsk(), which yield promises of appropriate type, i.e. *sparql.ResultSet*, *rdf.Graph* and boolean.

Listing 2 demonstrates the creation of a SPARQL service that performs pagination[8], query transformations for some issues with Virtuoso[9] 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)

// Note: This accompanying fluent API is offered for convenience:
sparqlService = service.SparqlServiceBuilder
    .http('http://dbpedia.org/sparql', ['http://dpbedia.org'])
    .virtFix().cache().paginate(1000).create();

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.createVar('s'));
    }
});
```

Listing 2: Usage example of the SparqlService API

[6] `http://jena.apache.org/`

[7] We are currently experimenting with the integration of the SPARQL.js parser: `https://github.com/RubenVerborgh/SPARQL.js`

[8] The default implementation assumes that limit and offset do not influence the result set order.

[9] For example, some versions of Virtuoso do not support queries having an outer-most OFFSET greater than 20000 in conjunction with an ORDER BY clause. Please refer to the Jassa github page for implementation details.

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. 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[10]. A usage example of Sponate is shown in Listing 3.

```
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
})
```

Listing 3: Example of a Sponate mapping

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 decorators, as shown in Listing 4, where the basic lookup service is based on a Sponate store.

```
var ls = new service.LookupServiceSponate(store.projects);
ls = new service.LookupServiceCache(ls);
// Execute the original lookup request by performing lookups with
// partitions of at most 20 keys - avoids large SPARQL queries
ls = new service.LookupServicePartition(ls, 20);
// Validate each key by a filter predicate before doing the actual request
ls = new service.LookupServiceKeyFilter(ls, predicateForValidatingUris);
// Merge all lookup requests within a 50ms time window into a single request
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 */
});
```

Listing 4: Decoration of a LookupService

2.5 The Faceted Browsing Widgets

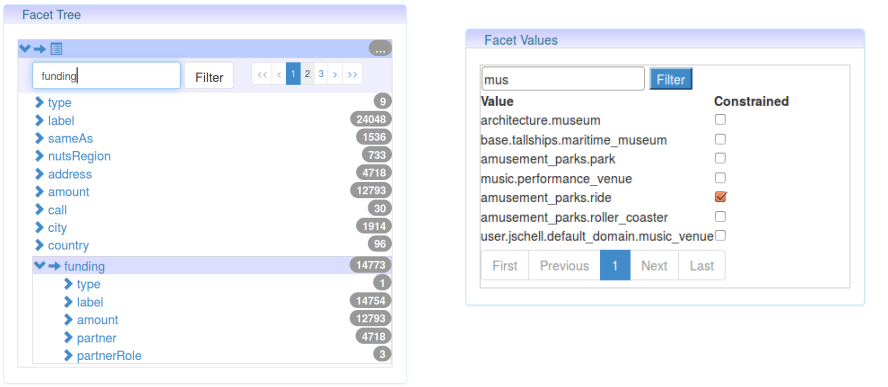
Jassa comes with a powerful SPARQL-based faceted search module in the *facete* namespace, which supports the definition of custom constraint types as well as

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

constraining sets of resources by indirectly (possibly inversely) related properties. Due to space limitations, we only demonstrate the usage of the faceted browsing widgets, shown in Figure 2. These widgets are implemented as AngularJS directives and can thus be embedded into AngularJS applications using the corresponding HTML snippets. The HTML attribute values refer to JavaScript objects, of which a basic setup is shown in Listing 5. Note that the widgets are synchronized by AngularJS on the state of the *fctTreeConfig* object; any change will automatically trigger an update of the widgets.

```
$scope.fctTreeConfig = new facete.FacetTreeConfig();
$scope.selectedPath = null; // Start with no selection
$scope.selectFacet = function(path) { $scope.selectedPath = path; }
```

Listing 5: Basic setup to make the facet-value-list show the values for a selected facet




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

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

(a) Facet Tree **(b)** Facet Value List

Fig. 2: Widgets for Faceted Browsing & the HTML (AngularJS-based) to create them

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[11], 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

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

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 a specification on RDF interfaces in JavaScript[12]. In regard to faceted browsing of RDF datasets, there are for instance the Sparklis browser[13] and the Pelorus faceted navigation tool[14]. 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, pagination and page expansion) and a faceted search module. The *jassa-ui* modules introduce user interface components for faceted browsing 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 with the aim of improving overall performance. Examples of applications built using Jassa include the generic SPARQL browser *Facete2*[15] and the *Linked Data Presentation Framework* [4].

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 *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 *23rd 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 Intl. Semantic Web User Interaction Workshop*, 2006.
4. D. Lukovnikov, C. Stadler, and J. Lehmann. Ld viewer - linked data presentation framework. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 124–131, New York, NY, USA, 2014. ACM.

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

