

Semantic Web & Javascript Disposition

Arne Hassel
Department of Informatics
University of Oslo
arnehass@ifi.uio.no

Fall 2011

Introduction

This document is the disposition to my master thesis. The thesis will be based on a framework that I'll develop alongside writing the thesis. Thus I've decided to first describe the framework and its functionalities. These aspects will then be tied to the thesis. I conclude with what I hope to learn and how these accomplishments will contribute to academia.

Contents

1	The framework	2
1.1	Linking/fetching	2
1.2	Deserializing	2
1.3	The engine	2
1.4	Simple Protocol and RDF Query Language (SPARQL)	3
1.5	Application Programming Interface (API)	3
1.5.1	Documentation	3
1.5.2	Debugging	3
1.6	Storing	4
1.6.1	Working with other sites	4
2	The thesis	4
2.1	Background	4
2.2	Problem Description and Requirements	4

2.3	Tools	4
2.4	Linking/Fetching	5
2.5	Deserializing	5
2.6	The engine	5
2.7	SPARQL	5
2.8	The API	5
2.9	Discussion	5
3	Conclusion	5

1 The framework

As I've discussed in my essay, the goal for the framework is to enable developers to work with Semantic Web (SW). The framework will be based on the programming language Javascript (JS), and will first and foremost serve development using that language.

1.1 Linking/fetching

In order to have data that can be processed, that data must be fetched. This part of the framework will ensure that developers can get the data, whether this is transcribed on a local file, on a remote server or fetched through a Linked Open Data (LOD)-point.

1.2 Deserializing

The data that is fetched, whether it is serialized as RDF/XML, N3, Turtle, or N-triples, needs to be interpreted to a Resource Description Framework (RDF)-like format that can be processed by JS. Some work has already been done here, both with libraries (e.g. `rdfQuery` [6] and `js3` [4], which both were discussed in the essay) and with standards (e.g. RDF Interfaces 1.0 [5]).

1.3 The engine

The framework must be able to process the data, and for this an engine is perfectly suited. Lot of work in SW have already been invested in engines, i.e. reasoners, and I can probably find good pointers in existing work.

Considering the flora of entailment regimes, the framework will only serve a simple entailment at its core. Hopefully it is possible to create a plugin-interface, that enables other developers to contribute to more advanced entailments regimes.

There is also the possibility that the framework can outsource all of this work to existing frameworks, i.e. there's no need to develop an engine as part of this thesis.

1.4 SPARQL

To query the data it is natural to use SPARQL at its core. The framework will implement the recommended version at first [3], but it would be nice if it also supported the working draft [1] as well.

As with the engine, it's probably possible to outsource this part to existing frameworks.

1.5 API

When the data is "prepared", i.e. fetched and deserialized, it is ready to be processed by the developer. An API must be developed that serves a level of abstraction that connects the developer with the core functionality.

Although not as important as the API itself (when considering the thesis), two other aspects should be noted, namely documentation and debugging.

1.5.1 Documentation

Any good API offers a good documentation. Although the methods may seem intuitive and expressive for me and my supervisors, a documentation should be provided.

1.5.2 Debugging

Considering the difficulties there can be when working with libraries that are yet unfamiliar, it is important to support good tools for debugging. I believe it is outside the scope of this thesis to create any advanced debugging-tools, but at least I can supply some good outputs, that enables developers to debug nicely with existing debugging-tools, such as Firebug for Mozilla Firefox, or the Developer Tools in Google Chrome.

1.6 Storing

Although not initially a part of the framework, developers may be interested to store the data processed in their application. The advantage by doing this with JS could be fast retrieval of stored data, since this can be stored in a format easily processed by JS.

1.6.1 Working with other sites

It would be interesting to see if the framework could work with other services, e.g. by enabling the use of the SPARQL 1.1 Graph Store HTTP Protocol [2]. This approach could enable developers to store data.

2 The thesis

It is important in a thesis that the user is presented with the necessary information required to understand the content provided. With that said, I will assume that the readers have a basic understanding of Information and communication technologies (ICT), i.e. what should be required of a bachelor student in the fields of computer studies.

In addition to the sections that I'll now present, the thesis will also contain a section on acknowledgments, an abstract, an introduction, and a conclusion.

2.1 Background

This section will present the underlying factors behind the thesis, such as SW and JS.

2.2 Problem Description and Requirements

This section will describe the problem description of the thesis, as well as the requirements it will solve during the succeeding chapters. This part will probably look somewhat like the part I've written on the framework in this disposition.

2.3 Tools

If another framework is included to support the framework, or if a specific tool with unique functionality have been used, this is the chapter that would describe them.

2.4 Linking/Fetching

This chapter will describe the choices I've made considering the part of the framework that enables the developers to fetch data.

2.5 Deserializing

This chapter will describe how the framework deserializes the fetched data, as well as the format to which it was deserialized. It can be interesting to see which alternatives there are to this format, and I can describe why one is favored before another.

2.6 The engine

This chapter will describe how the engine works, and how it's connected to the rest of the framework.

2.7 SPARQL

This chapter will describe how the framework enables querying with SPARQL.

2.8 The API

I think most of the interesting work will be done in developing the API. Why is one approach more favorable than the other? Is a strict approach more favorable than to a loose one, e.g. can "to much" freedom make it difficult for beginners to use the tool?

I think data can be collected for this part by mocking up some APIs with some basic functionality, and do some questionnaires/research with users.

2.9 Discussion

This part will be used to discuss the framework, and what that's been discovered in the process of its' development.

3 Conclusion

I think this work will offer some interesting contributions. First of all, it's gonna be interesting to see how much I can build on existing work, and how difficult this is to setup for a developer that wants to do the same. Can I build an API on existing tools, or do I have to reinvent the wheel in JS? How do the tools cooperate? Are they effective? Will it be troublesome to connect them together, and in the end offer an API that easily offers what developers need? I believe this work will reveal some pitfalls, which can be useful in themselves, or even in an abstract form.

References

- [1] S "Harris and A" Seaborne. Sparql 1.1 query language. <http://www.w3.org/TR/sparql11-query/>, May 2011.
- [2] C Ogbuji. Sparql 1.1 graph store http protocol. <http://www.w3.org/TR/sparql11-http-rdf-update/>, May 2011.
- [3] E "Prud'hommeaux and A" Seaborne. Sparql query language for rdf. <http://www.w3.org/TR/rdf-sparql-query/>, January 2008.
- [4] N Rixham. webr3/js3 - github. <https://github.com/webr3/js3>, November 2010.
- [5] N Rixham, M Sporny, M Birbeck, I Herman, and B Adrian. Rdf interfaces 1.0. <http://www.w3.org/TR/2011/WD-rdf-interfaces-20110510/>, May 2011.
- [6] J Tennison. rdfquery - rdf processing in your browser - google project hosting. <http://code.google.com/p/rdfquery/>, June 2011.