Lorem ipsum dolor?[✤]

Lorem Ipsum Reserch
Santa Claus[†]

Abstract

This document describes a specific software-architectural pattern as a means of providing solutions to recurring application development demands in the context of the 'Web of Data'. As such a pattern it can be described in the abstract, but it is felt that it has most utility in concrete scenarios involving the Web browser in conjunction with SPARQL servers and the Web at large.

The infrastructure components of a 'SPARQL Diamond' are a SPARQL server (database, query engine and HTTP server), a Web browser (a HTTP client with processing/rendering capabilities). A text templating engine intermediates custom queries from the user interface and the server endpoint (request). Templating is again applied between the server (response) to deliver appropriate rendering of the results.

This approach could potentially be used to create content-delivery sites with the same general functionality as typical database-backed systems. However it is felt that it particularly lends itself to rapid prototyping and the enhancement of existing pages with 3rd party information.

The intention here is to demonstrate that by exploiting widely-deployed systems together with a small number of utility libraries to implement the SPARQL Diamonds pattern, data-driven applications can be created with minimal effort.

Keywords: SPARQL, RDF, Web, software pattern.

✤ To insert more footnotes symbols such as ✤ you have to go on the top menu:
Insert / Footnote and Endnote / Footnote or endnote and select the desired character

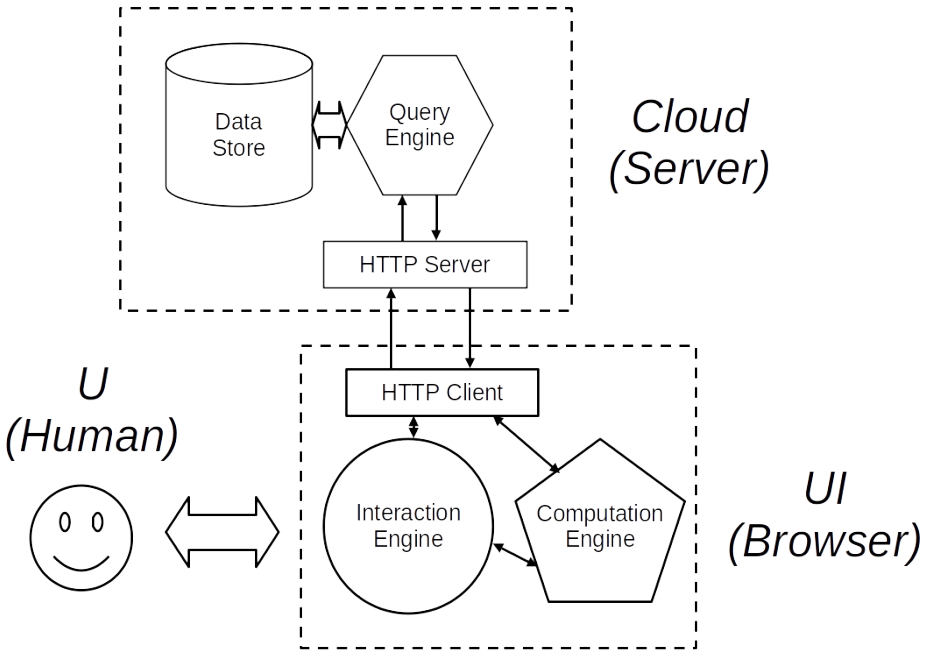
† Santa is a mythical figure with historical origins who, in many Western cultures, brings gifts to the homes of well-behaved, "good" children on Christmas Eve (24 December) and the early morning hours of Christmas Day (25 December).

Introduction

The technique has no doubt been used elsewhere before,

Operating Environment

The environment of SPARQL Diamonds is shown in Fig. 1.



This is very similar to a traditional database-backed Web site, the key difference being that instead of the query construction and presentation being carried out server-side, in this context with a SPARQL Server, all that work is carried out at the client. Although the block labelled *Cloud (Server)* shows typical internal subcomponents, the details of implementation are irrelevant. All that matters is some degree of support for the SPARQL Protocol [https://www.w3.org/TR/sparql11-protocol/].

As a Web-based system, SPARQL Diamonds are clearly dependent on the HTTP protocol. More specifically, they are inclined towards the REST [https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm] architectural style. Strictly speaking, SPARQL servers fall outside of this style, notably due to the use of the SPARQL Query Language inside the query part of URLs rather than REST's notion of potentially opaque identifiers that act purely as names for given resources. But in practice this can be viewed as an

implementation detail - the combination of endpoint URL with query could be interpreted as merely an arbitrary string identifying a resource, or even aliased from a more human-readable URL.

Client-side, the configuration described here will be that of a typical Web browser. But this doesn't preclude the use of the Diamonds pattern in a different environment such as a dedicated App for a mobile device. The only requirements are a HTTP client, a processing component to manipulate data and control its flow and some means of interacting with the end user.

The Diamond

Templating Engine

Text search-and-replace is a technique

Alternate Approaches

The SPARQL Diamond is obviously only one of a multitude of possibilities for interfacing with an online data store. One configuration deserves special mention, that of maintaining an RDF model at client. Several RDF API libraries are available [https://www.w3.org/community/rdfjs/wiki/Comparison_of_RDFJS_libraries]. This approach offers considerable advantages in terms of flexibility, for example with a SPARQL CONSTRUCT request, the resultant RDF can be merged directly into the local model. However this flexibility is typically at the cost of increased programming complexity.

Another set of solutions are possible by employing a server-side system between the SPARQL server (or other RDF store) and HTTP interface, with the browser being solely for user interface.

// footnote In fact the 'Diamonds' approach described here originated in an experimental server-side content management system [https://github.com/danja/seki]. After the 'lightbulb moment' of realising the processing could be shifted to the browser, a simple Wiki was built in this fashion [https://github.com/danja/foowiki], with a significant reduction in infrastructure requirements and considerably reduced development and deployment time.

how do other people do it?

Pros and Cons

The benefits of SPARQL Diamonds largely stem from the decoupling of the query and presentation phases of development from the operational wiring. A (very loose) analogy can be made with the Model-View-Controller of object-oriented programming. The model is implied by the query template (and instantiated by the results). The view is defined by the HTML template. The controller is determined by a handful of functions within the page Javascript (calling background libraries) which typically will require little or no change across application scenarios.

- Simplicity : the developer can focus on formulating queries and result presentation
- Largely declarative : the data and its presentation are primarily determined by the templates
- Relatively fast and lightweight in operation
- Compatible with well-known techniques and standard libraries (eg. jQuery)
-

RDF-unaware : the data structure operated on by the client is that of JSON rather than RDF meaning that graph manipulation techniques aren't available

Most of the time is spent navigating the data

References

error handling

security

Conclusions

SinLorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus sed leo ligula. Nam euismod libero in est commodo, non egestas libero sodales. In hac habitasse platea dictumst. Vestibulum ex enim, tincidunt et mollis at, pretium ac justo. Suspendisse pretium augue in lacus pulvinar ultrices. Fusce porta ipsum commodo fermentum aliquam. Mauris molestie urna nec dolor tempus, quis varius turpis semper. Integer pharetra mollis neque a venenatis.

First issue	Whatever and whatever;
Second issue	Super cool thing;
Third issue	Lesser cool but more useful;

Mauris ornare metus et ipsum scelerisque porttitor. Vestibulum id condimentum justo, quis accumsan dolor. Vestibulum tempus neque id nibh vulputate, ullamcorper eleifend purus semper. Duis velit risus, suscipit eu neque vitae, condimentum dignissim eros. Donec in odio eu arcu aliquam commodo. Donec condimentum consectetur placerat. Nulla facilisi. Aenean varius, diam non mattis hendrerit, diam tellus volutpat lacus, nec accumsan nisl nunc sed lectus. [1]

1 This work is made available under CC0 1.0 Universal (CC0 1.0) Public Domain Dedication, still original author is Michele Marrali, from Studio Storti Srl if you wanna give me credit I do not mind.