REST and Linked Data: a match made for domain driven development?

Kevin R. Page
Oxford e-Research Centre
University of Oxford
United Kingdom

David C. De Roure
Oxford e-Research Centre
University of Oxford
United Kingdom

Kirk Martinez
Electronics and Computer Science
University of Southampton
United Kingdom

ABSTRACT

At a first glance there might appear to be an obvious alignment and overlap between the approaches prescribed by REST and Linked Data. On more detailed inspection divergences in scope and applicability present themselves, and for some aspects, incompatibility. In this paper we investigate these similarities and differences and suggest the coupling is worthy of a third look: in combination as a flexible environment in which the developer can focus on domain driven applications.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

Keywords

REST, Semantic Web, Linked Data, domain applications.

1. INTRODUCTION

The Linked Data movement has achieved considerable success constructing a semantic Web of Data [2]. While much initial semantic web research focussed on building a stack to enable reasoning and logic, the more recent Linked Data programme has attempted to reconnect the semantic web to its roots in the most successful distributed system ever constructed (or at the very least the latter half of its moniker!).

Moving on from an earlier assumption that URIs would do nothing more than uniquely identify Things, the key thrust of Linked Data has been the re-adoption of HTTP URIs for retrieval of resource representations. The approach can be summarised by the four Linked Data 'rules' [1]: use URIs as names for things; use HTTP URIs so that people can look up those names; when someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL); and include links to other URIs, so that they can discover more things.

A shallow keyword match over these principles would suggest a strong correlation with those underpinning REST [5], and yet rarely are the two mentioned together as complementary styles. Are they at cross purposes, completely orthogonal, or can experience from both approaches inform a more coherent framework for building distributed web services and applications?

Through participation in recent and current projects variously implementing RESTful and Linked Data APIs, and drawing from a longer history of Semantic Web and hypermedia research, we have collected our thoughts in an attempt to answer these questions. In section 2 we begin with an overview of similarities, while the following section is an exploration of differences. In section 4 we propose that these aspects should be taken together as a complementary whole, and used – as illustrated by our experiences in section 5 – to enable domain driven application development in a data-rich world.

Copyright is held by the author/owner(s).
WS-REST 2011 March 2011; Hyderabad, India
ACM 978-1-4503-0623-2/11/03.

2. COMMON INTERESTS

2.1 The primacy of resources

The key abstraction of information in REST is a resource [5]; similarly the URI is both the identifier for, and means by which relationships are expressed between, things in the Resource Description Framework (RDF) [7], which is the foundation of the Semantic Web stack. In both cases, the notion of an identifiable resource is fundamental to implementation; design and development of a system cannot progress without the assignment and association of resources.

Since Resource Oriented Architectures [11] and Linked Data are the most commonly encountered realisations of REST and the Semantic Web respectively, and since both are built upon *HTTP* and *HTTP* URIs, it is easy to recognise this as a common shared building block. It is therefore also relevant to note that neither REST as an architectural style nor RDF as originally conceived are monogamously wedded to HTTP.

2.2 Linking is not optional

The fourth Linked Data principle is to "*Include links to other URIs*" in the representation provided when a URI is dereferenced "*so that they can discover more things*" [1]. It is this inclusion of links to other HTTP URIs which, when dereferenced, provide further links to more HTTP URIs that sets Linked Data apart from earlier Semantic Web activity in its explicit encouragement of a dereferencable Web (and the trails of links through it).

"*Hypermedia as the engine of application state*" (HATEOAS) is a defining characteristic of the REST architectural

style [5]. State transitions in an application occur when moving from one resource to another (by retrieving or modifying) using the links provided in a representation.

A representation that supports linking is therefore a requirement for both approaches; neither would function as intended without the hyperstructures described above. While there is no specific mandated linked representation for REST implementations, Linked Data advocates "*using the standards*" which, in the case of RDF and SPARQL, both guarantee support for links to other resources.

2.3 Segregating semantics

Semantics about relationships between resources can be expressed by both approaches: in the Semantic Web they are described by ontologies written in RDFS and OWL, while RESTful implementations can encode semantics in link relations.

A common misapplication of both approaches is to assume semantics (or abuse implied semantics) encoded in a URI, when both REST and Linked Data explicitly expect clients to regard URIs as opaque strings when used for identification. In this way both follow the principle of separating identification from the semantics of interaction, description, and structure.

2.4 Adaptability

Both REST and the Semantic Web include facets in their design which allow the relationships between resources to be modified, should revision be required, without necessitating interface changes to the client.

Since state in a RESTful application is defined by navigation of the hyperstructure, if a server changes the links that are transferred to a client (via a representation) it also changes the possible state transitions the application can make. It does this without changing the mechanism by which the client performs the transition (the combination of HTTP and the representations for the specified media type).

As befits a distributed web system (where it is perhaps unlikely – and probably undesirable – for there to be 'one true ontology'), there is no constraint on the application of a single ontology to each resource on the Semantic Web. Assertions can be made using different ontologies, in different places, and at different times; ontologies (themselves expressed in RDF) can be extended and subsumed by other ontologies.

In both cases this adaptability can be seen as a benefit of self-description – a client has prior knowledge of the framework within which relationships are expressed, but there is no requirement of prior knowledge of the relationships themselves.

2.5 Applicability of Domain Driven Design

The Domain Driven Design [4] methodology espouses a focus on domain modelling throughout an iterative development process. This has particular resonance with the principles and practices outlined above in respect to both REST and Linked Data: the identification of resources and the links between them should naturally map to the domain (and business process) at hand [10], and the ability to iteratively modify the hyperstructure lends itself well to agile development.

In the experience of the authors, this methodology is key in developing a service that can be successfully used by domain application developers, and in turn domain users: the power of a successful data service is in encapsulating the complexity of a domain in a manner that allows its use to scale through simple usage. This simplicity must be deeply tied to the domain to allow natural and intuitive use by domain developers and users; an abstraction unfamiliar or unsuitable to them will have an effect opposite to that desired.

3. IMPEDIMENTS TO THE MATCH?

In this section we outline those areas where one might perceive differences between REST and Linked Data – although, as we summarise in the next section, we counter that these are rather vestiges of different demands and current practice rather than fundamental incompatibilities.

3.1 API vs. Model

In section 2 we explored the similarities between REST and Linked Data, principally centred on the notion of resources and the relationships between them. There is, however, a key difference in the *motivation* for resource identification:

- in RESTful systems, resources and their relationships are identified and exposed to enable a client to retrieve data and transition to other resources; in effect, they define an API to enable application operation and state transition. Linking is the mechanism to navigate the API; link relations encode semantics to enable this.
- in RDF and ontologies, resources are identified to encapsulate the an underlying data model. While Linked Data extends this idea so that sections of the model can be retrieved by dereferencing resources, linking in the returned representation is used to bind sections of the model rather than transition state.

By extension the adaptability and self-documentation described in section 2.4 applies to API interactions for REST, and the data model for the Semantic Web.

3.2 SPARQL

The third Linked Data rule cites not only RDF, but a sister standard which from a RESTful point of view is a troublesome relative: SPARQL.

SPARQL is the standard query interface for RDF; it is widely deployed as an interface to Linked Data services, and widely used by Linked Data applications. However most SPARQL endpoints are implemented – and used – in the RPC style. RESTful interfaces to SPARQL have been proposed [13] which expose resources that, when a representation is requested, trigger SPARQL queries. Consistent with the previous section, identification of these query resources is a matter of identifying the "*information units*" which comprise the service API.

Perhaps a more concerning implication is the relative popularity of SPARQL for application development, and particularly for combining Linked Data through SPARQL endpoints. In this scenario, whilst the data model benefits from the distributed nature of resources and linking, the application interaction does not: it eschews the benefits of RESTful operation.

3.3 Content negotiation

RESTful services use content negotiation to select a shared envelope that both the client and server can encode and decode the representation through (and the interface to the service is then dynamically carried via the representation as links). Typically a REST service will assume the resource being transferred in these representations can be considered a document; in the terminology of the following section, an 'information resource'.

Linked Data services, in implementing the "HTTP range issue 14" solution [12], add semantics to the content negotiation to distinguish between URIs that are non-information resources (identifiers for conceptual or real-world objects) and URIs that are information resources (documents) that describe the non-information resources. This is because assertions in the RDF graph are usually relationships that apply to the non-information resource, but Linked Data overloads URI usage so that it is also a mechanism for retrieving triples describing that resource (in a document, i.e. an information resource). (This is a change in behaviour from earlier use of HTTP URIs in RDF, when they were *not* expected to be dereferenced.)

One widely deployed technical solution is to issue a 303 redirect from the non-information resource URI to a content-negotiated information resource (which will have representations containing descriptive information about the non-information resource; at least one representation will be an RDF serialization). The HTTP redirect signals the transition from non-information resource to the client. The practical consequence of the redirect is, in our experience, a (variable but) measurable additional delay for each complete transfer of information between server and client [3]; there is added complexity when compared to a REST API in which everything is simply an information resource.

3.4 RESTful through and through?

While there is clearly alignment in approach, and overlap in parts of implementation, are deployed best practice Linked Data services RESTful? On two further counts, we believe they could be considered to fall short.

Firstly, because resources are identified primarily for the purposes of correctly modelling the data (section 3.1), less thought is applied to the Linked Data URIs that can be dereferenced and how an application might use them – and the links between them – for RESTful state transition. If an API has not been designed for HATEOAS, then perhaps it is understandable that Linked Data developers appear to prefer SPARQL; or that adoption of SPARQL reduces motivation to design an API with HATEOAS in mind.

Secondly, the majority of Linked Data sites are read-only: they publish data but few have the ability to modify it (i.e. PUT, POST or DELETE). This may, in part, be due to the political Open Data movement which is frequently hard to distinguish from the technical push for Linked Data. Proposals for a SPARQL Update are well progressed, but carry the expected RPC issues; and while a Uniform HTTP Protocol for Managing RDF Graphs has been proposed, it remains a mechanism to encode SPARQL commands that are applied to a whole graph store, rather than manipulation of specific resources exposed through a RESTful API.

4. A HARMONIOUS FUTURE TOGETHER IN THE DOMAIN

In the previous sections we outlined where RESTful and Linked Data approaches share a common method and where they diverge. We do not, however, believe the differences are irreconcilable: while worthy of note, issues surrounding SPARQL, 303 redirects, content negotiation and writeable resources could all be mitigated or indeed solved though modifications to implementation and convention.

On the point of API vs. Model, we regard this as a complementarity rather than a "difference", particularly when considered in the context of domain driven design.

The authors have a particular interest in enabling third generation e-Research [3], where we have found a common agreed model is key to enabling domain experts, researchers, and developers to collaborate with each other over increasingly vast quantities of information. Here it is important for a researcher (or developer) to be able to use clear domain models that separate concerns to enable the manipulation of the domain data: this is a task RDF has proven adept at. It is equally important for a developer to be able to quickly and simply access, modify, and publish domain data through a lightweight API for scalable and distributed services: which REST enables. If common models can be used for both the API design (the RESTful interactions with resources) and the modelling of resource relationships (the RDF and ontologies) then the focus of complexity in any application can be where it really matters: the domain driven design.

We propose this can be achieved through consideration of the following design principles:

1. That RESTfully serving Linked Data offers an opportunity to use a common domain model for expressing, and identifying, the resources exposed by the *API* as well as the data model and for linking resources (within a particular service, and between services).

2. That representations of Linked Data (RDF) use a self-describing semantic model beyond the relatively simple link semantics in most REST deployments. This presents an opportunity for more sophisticated description and navigation of links in representations, and through this the application of stronger semantics (with a common underlying model) for application state transitions.

3. Services that publish Linked Data resources should pay careful consideration to HATEOAS as a viable alternative SPARQL, and identify resources to enable RESTful use of the API.

4. RESTful methods should be developed for the write-enabled Web of Data.

5. FAMILY OUTINGS

In this section we briefly outline three implementation case studies which have informed our thinking.

As part of the SemSorGrid4Env project we have developed a Linked Sensor Observation Service prototype [8] which serves Observation models as RDF, HTML, OGC O&M GML and OGC WFS GML (through content negotiation and 303 redirects). While the former representations can be used for more traditional hypertext navigation of the dataset, the latter representation is suitable for loading into GIS mapping applications where it is rendered as a 'layer'.

The RDF representations utilises external ontologies for the measured (observed) properties, and sensor resources carry links to nearby (real-world) resources by location. The original implementation was bespoke to one particular dataset and there was little separation of concerns between the dataset, data model, and API. A second major version, currently under development, keeps these as distinct elements: this allows configuration of the service for any observation derived dataset and independent configuration of the API (both of which are configured in relation to the underlying Observation data model). As such it is a manifestation of the distinction between API and model introduced in section 3.1. We are using this configurable API to investigate how we can develop client applications that driven by HATEOAS and evaluate them in comparison to our existing SPARQL based clients.

The Structural Analysis of Large Amounts of Music Information (SALAMI) project will publish an unprecedented volume (23,000 hours) of publicly available computational music analysis data and a corresponding subset of ground truth. These results will be made available to the musicology research community through a Linked Data API that encourages the rapid and simplified development of domain applications to analyse and augment the dataset. Our application of domain driven design to identify models and APIs has been exercised in an initial, smaller scale, prototype for genre analysis of music [9], comprising several services including an audio repository (serving RDF and MP3 representations), linked data metadata services, and workflow enactment and result repositories. Applications have been developed that use the services both to create collections of music for input to the analysis, and to examine and combine the results in conjunction with external data services (e.g. BBC Music, DBpedia, Jamendo).

myExperiment is a social website for sharing scientific workflows [6] and is the largest public repository of its kind. Developed since 2007 using Ruby on Rails and Web 2.0 methods, a REST-like API was added early in the project and has been used to write numerous applications (e.g. integration within the Taverna workflow system, Google gadgets, Facebook applications, Android applications). A previously distinct Linked Data service (including SPARQL) has recently been integrated with the pre-dating REST API, providing a consistent interface for users of the REST and Linked Data access mechanisms; co-evolution of the two interfaces and their eventual recombination has been informative when considering the similarities and differences between them.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have described the similarities and differences between the REST and Linked Data service architectural styles. Based upon this analysis we have identified a 'best fit' approach that draws upon the strengths of each, and proposed a way forward to better serve the development of domain driven applications. We will be applying these lessons to our future developments in e-Research as we build upon the services described in the previous section, and hope this paper can aid the exchange of ideas and good practice between the REST and Linked Data communities.

Acknowledgements

This work was supported by the 'Structural Analysis of Large Amounts of Music Information (SALAMI)' project funded by the JISC Digitisation and e-Content programme as a part of the Digging into Data challenge, and by the IST STREP Programme of the Commission of the European Communities as project number FP7-223913 'SemSorGrid4Env: Semantic Sensor Grids for Rapid Application Development for Environmental Management'. We are also grateful to our colleagues in these, and the Networked Environment for Musical Analysis (NEMA) and myExperiment projects, for their efforts realising the deployments which have informed our work.

7. REFERENCES

[1] T. Berners-Lee. Linked Data, Design Issues. http://www.w3.org/DesignIssues/LinkedData.html, July 2006.

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *Special Issue on Linked Data, International Journal on Semantic Web and Information Systems*, 2009.

[3] D. De Roure, C. Goble, S. Aleksejevs, S. Bechhofer, J. Bhagat, D. Cruickshank, et al. The evolution of myexperiment. In *IEEE International Conference on eScience*, pages 153–160. IEEE Computer Society, 2010.

[4] E. Evans. *Domain-driven design: Tackling Complexity in the Heart of Software*. Longman, 2004.

[5] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, Information and Computer Science, University of California, Irvine, California, USA, 2000.

[6] C. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, et al. myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research*, 2010.

[7] G. Klyne and J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. *World Wide Web Consortium (W3C) recommendation, February*, 2004.

[8] K. R. Page, D. De Roure, K. Martinez, J. Sadler, and O. Kit. Linked sensor data: Restfully serving rdf and gml. In *Semantic Sensor Networks 2009 (SSN09)*, October 2009.

[9] K. R. Page, B. Fields, B. Nagel, G. O'Neill, D. De Roure, and T. Crawford. Semantics for music analysis through linked data: How country is my country? In *IEEE International Conference on eScience*, pages 41–48. IEEE Computer Society, 2010.

[10] Y. Raimond, T. Scott, S. Oliver, P. Sinclair, and M. Smethurst. Use of Semantic Web technologies on the BBC Web Sites. In D. Wood, editor, *Linking Enterprise Data*, pages 263–283. Springer, 2010.

[11] L. Richardson and S. Ruby. *RESTful Web Services*. O'Reilly & Associates, May 2007.

[12] L. Sauermann and R. Cyganiak. Cool URIs for the Semantic Web. W3C Semantic Web Education and Outreach Interest Group Note, Mar. 2008.

[13] E. Wilde and M. Hausenblas. RESTful SPARQL? You name it!: aligning SPARQL with REST and resource orientation. In *Proceedings of the 4th Workshop on Emerging Web Services Technology*, pages 39–43. ACM, 2009.