Describing Linked Data Platform Applications with the Hydra Core Vocabulary

Nandana Mihindukulasooriya and Raúl García-Castro

Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
{nmihindu,rgarcia}@fi.upm.es

Abstract. The Linked Data Platform (LDP) W3C Recommendation provides a standard protocol and a set of best practices for the development of read-write Linked Data applications based on HTTP access to Web resources that describe their state using the RDF data model. The Hydra Core Vocabulary is an initiative to define a lightweight vocabulary to describe hypermedia-driven Web APIs. By specifying concepts commonly used in Web APIs such as hypermedia controls with their explicit semantics, the Hydra Core Vocabulary enables creation of generic API clients. This paper discusses how LDP applications can benefit from the Hydra Core Vocabulary to describe their APIs.
Using Hydra, an LDP application can enable generic clients by describing the semantics of the expected and returned data. Having an API documentation will be a more efficient approach for most LDP applications than gathering information about affordences and restrictions in each HTTP interaction. Nevertheless, there are potential conflicts that have to be taken into consideration such as Hydra collections vs LDP containers or Hydra paging vs LDP paging.

1 Introduction

Linked Data principles refer to a set of best practices for publishing and interlinking structured data on the Web following the success of the World Wide Web [1]. Nevertheless, similar to the early days of the Web of documents, this data is mostly read-only static data which is bulk-updated periodically.

Read-write Linked Data applications can bring several benefits to areas such as Enterprise Application Integration (EAI) [2]. W3C Linked Data Platform (LDP) and Hydra are two standardization initiatives that provide foundation for building interoperable read-write Linked Data applications.

The LDP specification[1] provides a standard protocol and a set of best practices for the development of read-write Linked Data based on HTTP access to web resources that describe their state using the RDF data model. The standardization of this protocol represents a step forward in the Linked Data community as it lays the ground for the development of interoperable read-write Linked Data applications.

[1] http://www.w3.org/TR/ldp/

The Hydra Core Vocabulary[2] is developed by the W3C Hydra community group and it provides a lightweight vocabulary to create hypermedia-driven Web APIs. By specifying a number of concepts commonly used in Web APIs, it enables a server to advertise valid state transitions following REST best practices. This paper discusses how LDP applications can benefit from Hydra to describe their APIs.

2 Example

LDP applications use HTTP headers to advertise the information about their affordances. For instance, when a resource is retrieved using a GET operation, the LDP server provides an *"Allow"* header (See Fig 1), which lists the allowed HTTP operations. Further, it provides an *"Accept-Post"* header, and an *"Accept-Patch"* header that advertise the media types accepted by the respective operations.

However, when generic RDF media types such as '*text/turtle*' or '*application/ld+json*' are used, the clients cannot discover the information about the application-specific structural or value constraints. For example, an application may require clients to use a certain vocabulary and may have cardinality restrictions. The LDP working group deferred addressing this issue until the W3C RDF Data Shapes[3] working group provides a mechanism for defining such constraints. As of now, one alternative is to use custom media types (e.g., application/vnd.myapp-concept+json) but this approach does not scale as the client will have to be aware of large number of custom media types and also increases coupling between the server and the client. If the application restrictions are defined in media type semantics, a new media type has to be introduced whenever those restrictions evolve.

A better alternative is to compliment LDP applications with a Hydra API documentation (see an example[4]). This allows the application to advertise the restrictions on the input data and possible outcomes using *hydra:supportedClass*, *hydra:supportedOperation*, *hydra:supportedProperty* and other properties of *hydra:apiDocumentation*. In such cases, the API documentation can be advertised using both *hydra:apiDocumentation* and *ldp:constrainedBy* Link relations. Further, the Hydra Core Vocabulary can be extended to provide information that goes beyond the media type scope such as the headers supported and the value restrictions on the supported headers[5].

3 Potential conflicts

Even though the Hydra Core Vocabulary can be used to describe APIs of LDP applications, there are some potential conflicts that have to be taken into account

[2] `http://www.hydra-cg.com/spec/latest/core/`
[3] `http://www.w3.org/2014/data-shapes/charter`
[4] `http://nandana.github.io/ldp-hydra/api.json`
[5] `http://github.com/HydraCG/Specifications/issues/99`

```
HTTP/1.1 200 OK
Content-Type: application/ld+json;
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
Allow: OPTIONS,HEAD,GET,PUT,PATCH,DELETE
Accept-Patch: text/ldpatch
Content-Length: 250
ETag: W/'123456789'

{
  "@context": ...
  "@id": "http://example.org/product/a",
  "issueTracker": "http://example.org/products/a/bugs"
}
```

Listing 1.1. Fig 1. GET response (product LDPR)

because the LDP specification and the Hydra Core Vocabulary have overlapping features such as LDP Containers vs Hydra Collections and LDP Paging vs Hydra Paged Collections.

3.1 LDP Container vs hydra:Collection

Organizing a set of Linked Data resources into a collection is a common use case in many Linked Data applications. Both LDP and Hydra provide constructs to maintain enumerations of related resources. LDP Containers act both as enumerations of related resources and creation factories for new resources. There are three types of LDP containers depending on the different levels of flexibility needed by an application: Basic, Direct, and Indirect. LDP clients can use the Prefer header[6] to provide a hint to the server about the appropriate response to a client's needs. Hydra collections are more simple and list the resources in the collection using the *hydra:member* property. However, Hydra collections are flexible and can be extended while describing their behaviour using the API documentation. An LDP Direct Container can simultaneously exist as a Hydra collection by defining *hydra:member* as the membership property and its behaviour be described using a Hydra API documentation.

3.2 LDP Paging vs hydra:PagedCollection

Because some resources can get large, sometimes it becomes necessary to split such resources into pages when served to a client. Both the LDP specification and the Hydra Core Vocabulary provide constructs needed for paging. The LDP Paging specification[7] uses the Link headers to specify that a served representation is a page (e.g., Link: <http://www.w3.org/ns/ldp#Page>; rel="type"). Further, it

[6] http://www.ietf.org/rfc/rfc7240.txt
[7] http://www.w3.org/TR/ldp-paging/

provides links to the "canonical" resource (e.g., Link: <http://example.org/res>; rel="canonical"), and the next page. Optionally LDP servers may provide links to the first, last, and previous pages using similar Link headers.

Further, LDP Paging defines parameters that a client can use to provide hints to a server about the representations that it can handle (e.g., *max-triple-count*, *max-kbyte-count*, and *max-member-count*). When ordering is important, LDP Paging allows servers to specify an *ldp:pageSortCriteria* using *ldp:pageSortOrder*, *ldp:pageSortPredicate*, and optionally a *ldp:pageSortCollation* properties. In such cases, the LDP server ensures that all the members on any single page have the proper sort order with relation to all members on any next and previous pages. The ldp:pageSortCriteria reuses SPARQL[8] SELECTs ORDER BY clause to define ordering.

Similarly Hydra defines a *hydra:PagedCollection* which may include links to *firstPage, nextPage, previousPage,* and *lastPage*. Further, it may contain information about total items of the resource and the number of items included in a page. However, unlike LDP Paging, these information are included in the page content itself rather than in Link relation headers. Having these two different approaches makes it a bit cumbersome for the paging clients as they have to be aware and be able to handle both approaches.

4 Conclusion

This paper discusses the possibility of using the Hydra Core Vocabulary for describing LDP applications. The early attempts to describe LDP applications with a Hydra API documentation shows that it is possible even though there are some conflicts to take care of. While collections can coexist in both cases, the paging clients need to be aware of two different approaches. Thus, there is also an opportunity for the two W3C working groups to collaborate and learn from each other than implementing two distinct mechanisms for a similar purpose such as paging.

Acknowledgments: The authors are supported by the *4V* (TIN2013-46238-C4-2-R) project.

References

1. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis lectures on the semantic web: theory and technology **1**(1) (February 2011) 1–136
2. Mihindukulasooriya, N., García-Castro, R., Esteban Gutiérrez, M.: Linked Data Platform as a novel approach for Enterprise Application Integration. In: Proceedings of the 4th International Workshop on Consuming Linked Data (COLD2013), Sydney, Australia (October 2013)

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