LDP4j: A framework for the development of interoperable read-write Linked Data applications

Miguel Esteban-Gutiérrez, Nandana Mihindukulasooriya, and Raúl García-Castro

Center for Open Middleware
Ontology Engineering Group, Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid, Spain
{mesteban,nmihindu,rgarcia}@fi.upm.es

Abstract. Enterprises are increasingly using a wide range of heterogeneous information systems for executing and governing their business activities. Even if the adoption of service orientation has improved loose coupling and reusability, applications are still isolated data silos whose integration requires complex transformations and mediations.
However, by leveraging Linked Data principles those data silos can now be seamlessly integrated, and this opens the door to new data-driven approaches for Enterprise Application Integration (EAI). In this paper we present LDP4j, an open souce Java-based framework for the development of interoperable read-write Linked Data applications, based on the W3C Linked Data Platform (LDP) specification.

1 Introduction

Nowadays every organization uses several information systems to manage their information and integrating those applications is a key requirement for efficiently exciting the business processes of organizations. Enterprise Application Integration (EAI) techniques, which propose solutions to this problem, have evolved over time from ad hoc one-to-one integrations to approaches such as Service-Oriented Architectures (SOA) and Enterprise Service Buses (ESBs), using either SOAP-based or RESTful web services. Application integration using read-write Linked Data is a novel approach that is getting traction in the industry[1].

Application connection (interfaces) and data integration are the two main problems in application and data integration; there are three main challenges: *syntactic heterogeneity*, *structural heterogeneity*, and *semantic heterogeneity* [1]. The usage of standard data exchange formats such as XML or JSON solves the syntactic heterogeneity problem in the current EAI techniques and the structural heterogeneity problem is solved by complex schema transformations. However,

[1] https://jazz.net/story/about/about-jazz-platform.jsp

because the explicit semantics of data is not clearly expressed, traditional approaches struggle with semantic heterogeneity.

Nevertheless, Semantic Web technologies provide better solutions to the data integration problem. The Linked Data principles help creating a global data space [2] with typed links between data from different sources [3], hence breaking isolated data silos. RDF provides a simple and flexible data model that is well-suited for data integration and the conceptualization of domain models can be expressed in terms of RDF Schema and OWL ontologies. Machine-readable structured data with explicit formal semantics that are expressed using standards makes merging, integrating, processing, and analyzing data possible without needing out-of-band knowledge or proprietary tools. Links to related entities in data make it possible to start from a piece of data and traverse through different sources with a follow-your-nose approach[2] in order to discover more entities and get context information.

The Linked Data Platform (LDP) specification[3] provides a standard protocol for read-write Linked Data and a set of best practices, based on HTTP access to web resources that describe their state using the RDF data model. The standarization 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.

As a consequence of this, LDP provides a base for application integration using read-write Linked Data; however, this approach requires having tools and libraries that provide support for developing read-write Linked Data applications that support the LDP protocol.

At the time of this writing, support for this specification is being included in existing Semantic Web commercial tools as well as in green-field and brown-field open source projects[4]. Unfortunately, in these cases LDP support is provided as a remote data access mechanism, not an application integeration mechanism.

The LDP4j framework is one effort to fill this gap by providing a library and a set of tools that facilitate the development for read-write Linked Data applications so that they can follow this novel approach of application integration using Linked Data.

This paper presents the LDP4j framework, discusses lessons learned, challenges, and future work. The paper is organized as follows. Section 2 discusses the requirements beyond LDP for application integration scenarios. Section 3 introduces the LDP4j framework. Finally, Section 4 draws some conclusions.

2 Interoperable read-write Linked Data Applications: Beyond the Linked Data Platform

Applications that support the LDP protocol can expose all or part of their data using one or more vocabularies and can consume Linked Data from other ap-

[2] `http://patterns.dataincubator.org/book/follow-your-nose.html`

[3] `http://www.w3.org/TR/ldp/`

[4] See `https://www.w3.org/wiki/LDP_Implementations` for a list of LDP implementations.

plications by following links and traversing through data. This opens the door to a novel approach for integrating applications [4]. However, one of the lessons learned while developing LDP4j was that as a middleware provider LDP support is not enough to get adopted as a viable approach in industry. In order to integrate heterogeneous applications using read-write Linked Data in a production environment, several quality requirements have to be fulfilled and this section discusses some of these requirements[5].

Most enterprise applications have security requirements that generally include authentication, authorization, accounting, integrity, confidentiality, and non-repudiation. The LDP protocol must not just be integrated with current security solutions for web applications (i.e., WebID [5]) but also explore the specificities that stem from the way in which the data is made available.

Furthermore, applications need support for business transactions [6] to ensure consistency. Depending on the level of consistency required, strong ACID properties [7] or other alternatives such as BASE [8] must be guaranteed. Several RESTful transaction models have been proposed for web applications in the last few years with some limitations [9] and a transaction model for LDP should be built using them as the base.

Finally, data validation is a vital step for ensuring the quality of data in applications and expressive schema languages and related tools are essential for effective data validation. In constrant with relational databases and XML, RDF is built upon the Open World Assumption and the Non-unique Name Assumption, and this makes data validation challenging, as modelling languages (RDF Schema and OWL) are more more suited for inferring than for validation. Thus, there is a need for new standards and tooling support[6].

3 LDP4j

LDP4j[7] is a Java-based framework for the development of interoperable read-write Linked Data applications based on the LDP specification. This framework provides the components required by clients and servers for handling their communication, hidding the complexity of the protocol details from application developers and letting them focus on implementing their application-specific business logic (see section 3.1). In addition, the framework plans extensions to the LDP specification by providing additional features aimed at enhancing the interoperability between LDP-based read-write Linked Data applications (see section 3.2). The LDP adapter for the Bugzilla [10] bug tracker and morph-LDP [11] are two example applications built using the LDP4j framework.

3.1 LDP Support

In order to facilitate the development of LDP-based applications, LDP4j provides different components that developers may use for the development of such

[5] For an extended discussion of the requirements please refer to [4].

[6] http://www.w3.org/2014/data-shapes/charter

[7] http://www.ldp4j.org/

applications; in particular: an extensible LDP Server and an extensible LDP Client. The following sections present both components.

LDP4j Server Component. The purpose of the LDP4j Server Component (or *LDP Server* for short) is to provide the means for publishing application-specific LDP containers and resources, abstracting developers from the particularities of the LDP protocol and letting them focus on the particular business logic behind the containers and resources themselves.

The *LDP Server API* is divided into: (1) the **server API**, which defines entities related to different aspects of the LDP protocol that are of interest for the LDP Server (i.e., resources, content, and format); (2) the **server SPI**, which defines APIs that provide extensions to the types supported by the *server API*; (3) the **server frontend**, which provides developers with a frontend that allows the interaction with the different services provided by the LDP Server, if required; and (4) the **developer API**, which defines the low-level interfaces and annotations that have to be used to create server-side LDP applications using LDP4j. This latter part of the API provides and extensibility layer that enables the developer to specify the behaviour of the LDP4j Server when dealing with LDP containers and resources.

In order to develop *basic* LDP containers, developers must implement the *org.ldp4j.server.core.ILinkedDataPlatformContainer* interface that provides the means for handling the creation of LDP resources given an RDF serialization as well as for retrieving a RDF summary of contents of one or more of the resources managed by the container. Each container must be identified by an application unique identifier. In addition, every created resource must be identified by a container unique identifier.

On the other hand, the development of LDP resources requires implementing the *org.ldp4j.server.core.ILinkedDataPlatformResourceHandler*, which provides the means for retrieving the contents of existing resources and well as for updating their contents given an RDF serialization. Resource handlers must be identified by an application unique identifier, which should match that of the container used for the creation of the resources handled by the handler.

If the application supports resource deletion, developers will have to extend their resource handler classes in one of two ways: implementing also the *org.ldp4j.server.core.Deletable* interface, or adding a method with the same signature as the *delete* method defined in the latter interface, annotated with *org.ldp4j.server.core.Delete*.

LDP4j Client Component. While the LDP4j Server Component is meant to help developers in publishing contents via the LDP protocol, the purpose of the LDP4j Client Component is to let developers build applications capable of consuming those contents by exploiting the LDP protocol. Again, the LDP4j Client Component hides the specificities of the LDP protocol to the developer so that he only has to decide how to use those contents.

The *LDP Client API* provides the frontend to the LDP Client that developers have to use for consuming LDP contents. The frontend consists of a facade that allows creating **proxies** to containers and resources given their URLs. For the time being, these proxies provide the client-side functionalities that match those offered by the current version of the LDP Server API.

3.2 Extensions to LDP

Supporting the whole LDP specification is the first step in the development roadmap. However, in order to fully realize the vision of interoperable read-write Linked Data applications it is necessary to make explicit any application-specific domain knowledge required for interacting with the particular application in a sensible way. Further LDP4j work is geared towards the development of a set of extensions for enriching the LDP protocol along this line:

- **Vocabulary support**. Enable the publication and discovery of the vocabularies used by a read-write Linked Data application together with the restrictions that apply. As a side effect, this would relief application developers from the burden of input data validation as this could be transparently done by middleware.
- **Co-reference support**. Provide middleware services for dealing with the *coreference problem*[8], following the lines already identified in [12].
- **Transaction support**. Provide middleware services for providing transaction support for Linked Data applications using a RESTful transaction model.

Finally, support for other enterprise requirements aimed at facilitating the adoption and uptake of the framework are planned, in particular the provision of Linked Data-specific access control, authorization, and accounting mechanisms.

4 Conclusions

The usage of Linked Data as a means for integrating applications has several advantages over traditional EAI approaches. However, using Linked Data for this purpose requires having mechanisms not just for reading, but for writing data. The LDP specification defines a protocol for read-write Linked Data applications and its standardization represents a big step towards the industrial adoption of Linked Data technologies.

However, the interoperability of applications supporting the LDP protocol is limited to domain-independent concerns (i.e., exchange formats, communication patterns, failure signaling, etc.). Thus, their integration requires out-of-band domain knowledge in order to use them properly while interacting with them

[8] When information about a certain entity —which is identified in different manners— is spread across different sources.

via the LDP protocol. As a result, LDP is not enough for Linked Data-based EAI: it is necessary to put domain-knowledge into the game.

In this paper we have presented LDP4j, a framework for the development of read-write domain-aware Linked Data applications. The framework implements and extends the LDP protocol with features for exposing the domain knowledge that dictates how to interact with the application. This framework represents the next big step for realizing the vision of Linked Data-based Enterprise Application Integration.

References

1. Gagnon, M.: Ontology-based integration of data sources. In: Proceedings of 10th International Conference on 10th International Conference on Information Fusion (FUSION2007), IEEE (2007) 1–8
2. 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) (2011) 1–136
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS) **5**(3) (2009) 1–22
4. 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 (Oct 2013)
5. Story, H., Harbulot, B., Jacobi, I., Jones, M.: FOAF+SSL: RESTful Authentication for the Social Web. In: Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009). (2009)
6. Papazoglou, M.P.: Web Services and Business Transactions. World Wide Web: Internet and Web Information Systems **6**(1) (2003) 49–91
7. Gray, J., Reuter, A.: Transaction Processing: Concepts and Techniques. first edn. Morgan Kaufmann, California, USA (Sep 1992)
8. Pritchett, D.: BASE: An Acid Alternative. Queue **6**(3) (May 2008) 48–55
9. Mihindukulasooriya, N., Esteban-Gutiérrez, M., García-Castro, R.: Seven challenges for RESTful transaction models. In: Proceedings of the companion publication of the 23rd international conference on World wide web, Seoul, South Korea (Apr 2014) 949–952
10. Mihindukulasooriya, N., Esteban-Gutiérrez, M., García-Castro, R.: A Linked Data Platform adapter for the Bugzilla issue tracker. In: Demo at the 13th International Semantic Web Conference (ISWC2014), Riva del Garda, Italy (Oct 2014)
11. Mihindukulasooriya, N., Priyatna, F., Corcho, O., Garcıa-Castro, R., Esteban-Gutiérrez, M.: morph-LDP: An R2RML-based Linked Data Platform implementation. In: Demo at the 11th Extended Semantic Web Conference (ESWC2014), Crete, Greece (May 2014)
12. Esteban-Gutiérrez, M., García-Castro, R., Mihindukulasooriya, N.: A Coreference Service for Enterprise Application Integration using Linked Data. In: 7th International Workshop on Applications of Semantic Technologies (AST 2013), Koblenz, Germany (Sep 2013)