A Linked Data Platform adapter for the Bugzilla issue tracker

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

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

Abstract. The W3C Linked Data Platform (LDP) specification defines a standard HTTP-based protocol for read/write Linked Data and provides the basis for application integration using Linked Data. This paper presents an LDP adapter for the Bugzilla issue tracker and demonstrates how to use the LDP protocol to expose a traditional application as a read/write Linked Data application. This approach provides a flexible LDP adoption strategy with minimal changes to existing applications.

1 Introduction

The W3C Linked Data Platform (LDP) is an initiative to produce a standard protocol and a set of best practices for the development of read/write Linked Data applications [1]. The LDP protocol provides the basis for a novel paradigm of application integration using Linked Data[1] in which each application exposes its data as a set of Linked Data resources and the application state is driven following the REST design principles [2].

Some advantages of this approach over traditional SOAP-based web services include: (a) global identifiers for data that can be accessed using the Web infrastructure and typed links between data from different applications [3]; (b) the graph-based RDF data model that allows consuming and merging data from different sources without having to do complex structural transformations; and (c) explicit semantics of data expressed in RDF Schema or OWL ontologies which can be aligned and mapped to data models of other applications using techniques such as ontology matching.

This approach is more suitable when the integration is data-intensive and the traceability links between different applications are important. The Application Lifecycle Management (ALM) domain, in which heterogeneous tools are used in different phases of the software development lifecycle, provides a good use case for this approach. The ALM iStack project[2] has developed a prototype for

[1] `http://www.w3.org/DesignIssues/LinkedData.html`
[2] `https://sites.google.com/a/centeropenmiddleware.com/alm-istack/`

integrating ALM tools by using the LDP protocol and this paper presents an LDP adapter developed to LDP-enable the Bugzilla[3] issue tracker.

2 An LDP adapter for Bugzilla

The three main alternatives for LDP-enabling an application are: (a) native support built into the application; (b) an application plugin; and (c) an LDP adapter. Providing native support requires modification to the application and not all applications allow extensions through plugins. As we have seen in the early stages of web services [4], adapters provide a more flexible mechanism to gradually adopting a technology while using the existing tools with minimum changes, and we have leveraged this approach.

An application is defined in terms of its data model and business logic. An LDP-enabled application exposes the data as Linked Data and allows to drive its business logic following the REST design principles. Thus to LDP-enable an application, its data model should be expressed in RDF by mapping it to a new ontology or by reusing existing vocabularies. In the Bugzilla adapter, the Bugzilla native data model is mapped to the ALM iStack ontology[4]. The adapter exposes the Bugzilla data as LDP resources by transforming the data between the ALM iStack ontology and the Bugzilla native model so that LDP clients can consume RDF data from Bugzilla as if it was a native LDP application.

The Bugzilla LDP adapter, which is a JavaEE web application, consists of three main layers: (a) *LDP layer*, (b) *transformation layer*, and (c) *application gateway layer*, as illustrated in Figure 1.

The **LDP layer** handles the LDP communications and exposes the Bugzilla data as LDP resources. This layer is built using the LDP4j framework[5] which provides a middleware for the development of read/write Linked Data applications [5]. The concepts such as bugs, products, product versions, and users are mapped to LDP containers which list these entities and allow creating new entities. Each entity such as a bug, a product, or a user is mapped to an LDP resource with its own URI that can be used by clients to access them.

The **transformation layer** handles data validation and transformation. This includes extracting information from RDF data, validating them based on application restrictions, and mapping them to the Bugzilla model. The ALM iStack ontology is generic so that it can be used with other issue trackers (*e.g.*, JIRA[6], Redmine[7]); thus there is an impedance mismatch between the ontology and the Bugzilla native model which is managed by the adapter.

The **application gateway layer** handles the communication with the Bugzilla instance using its XML-RPC remote interface. Because the Bugzilla bug tracker is also accessed using its web UI, the adapter synchronizes with the

[3] `http://www.bugzilla.org/`
[4] `http://delicias.dia.fi.upm.es/ontologies/alm-istack.owl`
[5] `http://www.ldp4j.org/`
[6] `https://www.atlassian.com/software/jira`
[7] `http://www.redmine.org/`

Bugzilla instance based on user-defined policies. In addition there are several cross-cutting services such as configuration management, consistency, security, and synchronization which are utilized by multiple layers.

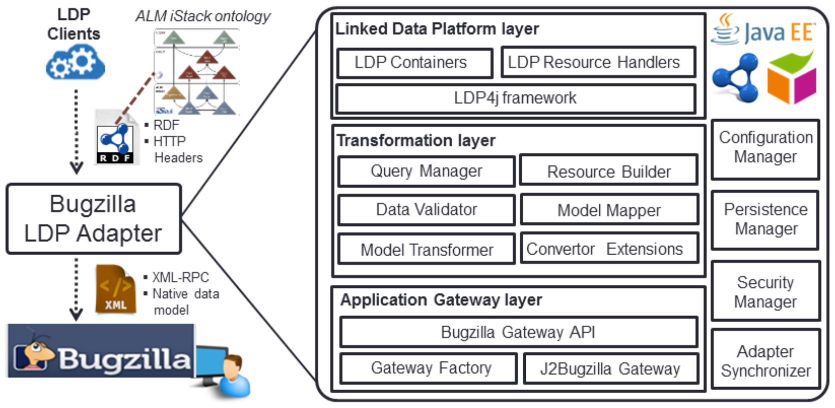
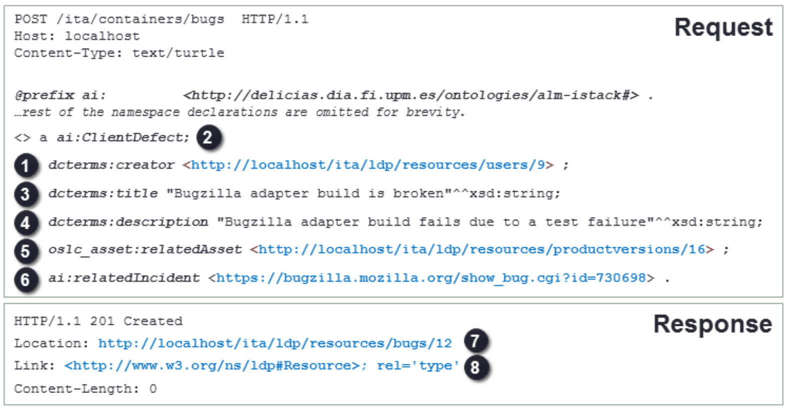


Fig. 1. High-level architecture of the Bugzilla adapter

3 Demonstration

This demonstration shows how LDP clients can use the adapter to access the Bugzilla bug tracker and to perform tasks such as discovering bugs reported against a product, modifying the status or the other properties of the bug, or creating new bugs (e.g., Fig. 2 shows a creation request and response).



```
POST /ita/containers/bugs  HTTP/1.1
Host: localhost
Content-Type: text/turtle

@prefix ai:         <http://delicias.dia.fi.upm.es/ontologies/alm-istack#> .
…rest of the namespace declarations are omitted for brevity.
<> a ai:ClientDefect; 2
1 dcterms:creator <http://localhost/ita/ldp/resources/users/9> ;
3 dcterms:title "Bugzilla adapter build is broken"^^xsd:string;
4 dcterms:description "Bugzilla adapter build fails due to a test failure"^^xsd:string;
5 oslc_asset:relatedAsset <http://localhost/ita/ldp/resources/productversions/16> ;
6 ai:relatedIncident <https://bugzilla.mozilla.org/show_bug.cgi?id=730698> .
```

```
HTTP/1.1 201 Created
Location: http://localhost/ita/ldp/resources/bugs/12 7
Link: <http://www.w3.org/ns/ldp#Resource>; rel='type' 8
Content-Length: 0
```

Fig. 2. Creation of a new bug using the Bugzilla LDP adapter

For example, a continuous integration server in an integrated ALM setting encounters a build failure. Thus, the "*integration server agent*" (1) wants to report a defect (2) titled "*Bugzilla adapter build is broken*" (3) with description "*Bugzilla adapter build fails due to a test failure*" (4) for the "*version 1.0* *of the Bugzilla Adapter*” product (5) that is related to the “*issue 730698*” in “*https://bugzilla.mozilla.org/*” (6). The LDP client converts this message to an LDP request according to the ALM iStack ontology as shown in Figure 2.

Once this request is received by the adapter, it extracts the necessary information, transforms it into the Bugzilla model using a mapping between the ontology and Bugzilla models, and creates a bug in the Bugzilla instance using its remote XML-RPC interface. Once created, the Bugzilla instance returns the identifier for the bug inside Bugzilla. Then, the adapter generates an URI for the bug and manages the mapping between the identifier given by the Bugzilla and the URI. Any information that does not fit into the Bugzilla model such as links to external applications is maintained in the adapter. Finally, the adapter returns the URI using the Location header (7) and lets the client know it is an LDP resource using the “type” link relation (8) according to the LDP protocol.

The LDP client or other external applications can access and link to the bug using the URI returned by the adapter. In addition, clients can modify the bug using the PUT operation with modified RDF data which then will be propagated to the Bugzilla instance following a similar process.

4 Conclusion

In this paper, we presented the Bugzilla LDP adapter and provided an overview of how to build adapters for LDP-enabling existing applications in order to use them as read/write Linked Data applications. With minimal changes to the existing application, the Bugzilla LDP adapter enables semantic integration of the Bugzilla tool with other LDP-enabled applications and makes possible to have typed links between application data.

Acknowledgments: The authors are partially supported by the ALM iStack project of the Center for Open Middleware.

References

1. Speicher, S., Arwe, J., Malhotra, A.: Linked Data Platform 1.0 (June 2014) W3C Candidate Recommendation, `http://www.w3.org/TR/ldp/`.
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 (Oct 2013)
3. 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
4. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M., Toumani, F.: Developing adapters for web services integration. In: Advanced Information Systems Engineering, Springer (2005) 415–429
5. Esteban-Gutiérrez, M., Mihindukulasooriya, N., García-Castro, R.: LDP4j: A framework for the development of interoperable read-write Linked Data applications. In: Proceedings of the 1st ISWC Developers Workshop, Riva del Garda, Italy (Oct 2014)