Implementing Linked Widgets: Lessons Learned for Linked Data Developers

Tuan-Dat Trinh, Peter Wetz, Ba-Lam Do, Amin Anjomshoaa, Elmar Kiesling, and A Min Tjoa

Vienna University of Technology, Vienna, Austria
{tuan.trinh,peter.wetz,ba.do,amin.anjomshoaa, elmar.kiesling,a.tjoa}@tuwien.ac.at

Abstract. Seven years after Linked Data has been introduced as a concept to publish data on the web, an abundant cloud of Linked Open Data (LOD) built upon standard web technologies has emerged. To facilitate and encourage widespread use of that data, a critical step is now to streamline the process for creating applications on top of LOD. This paper discusses lessons learned while developing an open standards-based platform that aims to achieve that by means of Linked Widgets. Whereas resources are already connected in the LOD cloud, Linked Widgets in a similar vein aim to alleviate LOD application development in an open and interlinked fashion. Through reuse, we aim to foster both users' and developers' productivity and creativity.

1 Introduction

Due to substantial efforts by developers, researchers, and practitioners, the LOD cloud has grown to approximately 1000 working datasets and 60 billion triples just seven years after researchers introduced the Linked Data [2] concept.[1] This approach for publishing data rests upon standard web technologies such as HTTP, RDF and URIs. Today, the LOD cloud includes data from various domains including media, geography, government, and publications.

The impressive growth of the LOD cloud may suggest that the Semantic Web vision is becoming a reality. However, although computers can process semantically annotated information from these RDF datasets, end users, who have limited experience with Semantic Web technologies, still cannot easily access or instruct computers to collect and process data. Therefore, users still rely on developers to create relevant and interesting LOD applications for them. So far, the number of domain-independent LOD applications targeted at end users has been limited. Those that do exist are typically proprietary and developed by a closed group of developers; as a consequence, sharing and reuse of applications is restricted and users can access only a small part of the LOD cloud. In order to overcome this major roadblock to LOD adoption, novice users need means

[1] `http://stats.lod2.eu`

to overcome the technological barriers that keep them from exploring the LOD cloud in a flexible and effective manner.

To address this challenge, we have developed an open mashup platform[2] as a bridge to tie together three stakeholders, i.e., data publishers, developers, and end users. We provide further details about this platform in [3,4]. The key idea of the platform is called Linked Widgets – a type of web applications on top of Linked Data – implemented by developers to enable end users to obtain, process, or visualize data from the LOD cloud and other open data sources. A Linked Widget has input and output models, which are semantically enriched through an annotator and published as Linked Data accessible via a SPARQL endpoint. When connected, each widget can receive data and return its processed output data to another widget. Users without a skillset in Semantic Web technologies can readily combine Linked Widgets in different ways to compose LOD applications.

Fig. 1 illustrates an example of such LOD applications. It detects and presents beautiful spots and corresponding images on a map based on a touristic textual description. By connecting *Text Annotator* – which detects a list of DBpedia [1] locations from text – and *Image Search* – which enriches all types of geographic objects with Flickr[3] images, the location resources from DBpedia are enriched with images from Flickr. Similarly, users can connect DBpedia resources to EventMedia[4] resources by adding the *Music Event Search* between the *Text Annotator* and the *Image Search*, as illustrated in Fig. 2.

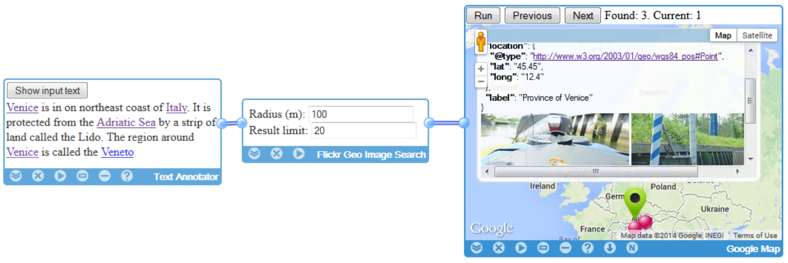
This paper discusses three challenging implementation considerations that we faced while developing the linked widget platform. The remainder of this paper is organized as follows. In Section 2, we introduce crucial architectural features that LOD applications should follow. Section 3 proposes frameworks for interactive LOD web applications. Section 4 is our suggestion for the data format of lightweight LOD applications. Finally, the paper concludes in Section 5.

2 Architectural Design Considerations

When defining the architecture for our mashup platform, we set out to follow three essential design principles, (i) *openness*, (ii)*connectedness*, and (iii) *reusability*. These are also the principles that we propose LOD applications should follow.

First, LOD applications cannot tap their full potential, if they are not open. They should follow an open architecture that enables arbitrary developers and end users to contribute and share their work with the LOD community. An example for the benefits of openness is LinkedGeoData,[5] which uses information collected by the OpenStreetMap[6] project and makes it available as an RDF

[2] `http://linkedwidgets.org`
[3] `https://flickr.com`
[4] `http://eventmedia.eurecom.fr`
[5] `http://linkedgeodata.org/About`
[6] `http://www.openstreetmap.org/`



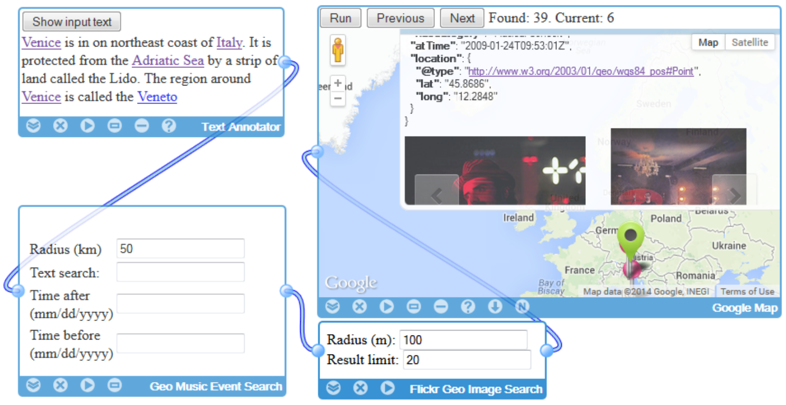
Venice is in on northeast coast of Italy. It is protected from the Adriatic Sea by a strip of land called the Lido. The region around Venice is called the Veneto

Radius (m): 100
Result limit: 20

Found: 3. Current: 1

```
"location": {
 "@type": "http://www.w3.org/2003/01/geo/wgs84_pos#Point",
 "lat": "45.45",
 "long": "12.4"
},
"label": "Province of Venice"
}
```

Fig. 1: Display detected-from-text famous locations (and images) on the map



Venice is in on northeast coast of Italy. It is protected from the Adriatic Sea by a strip of land called the Lido. The region around Venice is called the Veneto

Radius (km) 50
Text search:
Time after (mm/dd/yyyy)
Time before (mm/dd/yyyy)

Radius (m): 100
Result limit: 20

Found: 39. Current: 6

```
"at Time": "2009-01-24T09:53:01Z",
"location": {
 "@type": "http://www.w3.org/2003/01/geo/wgs84_pos#Point",
 "lat": "45.8686",
 "long": "12.2848"
}
}
```

Fig. 2: Display music events (and images) nearby detected locations on a map

knowledge base according to the Linked Data principles [2]. It allows users to directly edit displayed resources in the map view, which simplifies the process of extending data quantity and improving data quality. Similarly, we encourage developers to contribute Linked Widgets to our mashup platform to enable new combinations of LOD datasets. Furthermore, even end users can create new widgets from composed applications without any programming. Users of the community can finally share, reconfigure, or edit the created LOD applications.

The *openness* feature improves the way users can store and exploit data of the available LOD applications. This means that data valuable for the community should be accessible for anyone in a well-structured format. For example, the input and output models of Linked Widgets as well as the metadata of widgets, widget collections, and composed LOD applications of the mashup platform are published via a SPARQL endpoint. Third parties can develop additional functionalities for the platform, e.g., *widget search*, or *input/output model matching* which enables users to find – given input/output A – all output/input B from all other widgets such that the connection between A and B is valid. Users then can discover all possible connections between widgets to continuously build more applications.

Next, we design the architecture around the idea of *connectedness*, which is implemented via two concepts, i.e., *data connection* and *functionality connection.* Since we, except for domain-specific applications, have many interconnected LOD datasets, the architecture and its implementation should not restrict themselves to a small number of datasets. Instead, combining two or more datasets and enriching the data with additional value creates exciting opportunities.

From the *openness* feature we can derive the *functionality connection* and the *reusability* feature. Anyone can contribute new functionalities to an application, however, those functionalities should not be separated from each other. It should be possible to connect and reuse them in an effective and efficient manner. Our mashup platform supports reuse in four ways: (i) Users can creatively combine Linked Widgets from different developers to compose LOD applications, (ii) they can reuse LOD applications from others, but change the parameters of the constituted widgets, (iii) they can reuse a composed LOD application as a new widget, and (iv) based on available widgets, developers can implement new widgets to support new use cases.

Another architectural design consideration for LOD applications, which integrate data from multiple sources, is to decide where the data processing task should be performed. The alternatives are to either do it locally at the client application or remotely at the server side. Because a server of high quality datasets can easily become overloaded with too many requests from clients, we should make use of client resources whenever possible. An example is Linked Data Fragments [6] in which the client itself will execute complex SPARQL queries after receiving the data fragments – corresponding to its defined triples – from the server. In the mashup platform, currently, data processing is similarly done on-the-fly in the client's browser. However, to deal with additional types of data, i.e., stream data, real-time data, big data, and long-time processing data in future, we will consider widgets just as a user interface on the client and the actual data retrieval and processing tasks, instead, could be performed at the server.

3 Frameworks for Interactive LOD Web Applications

Web platforms can provide an ideal environment for creating accessible, sharable, extensible, and maintainable LOD applications. Interactive web applications typically rely on JavaScript libraries to provide rich graphical user interfaces on the client side (e.g., drag and drop). Choosing an appropriate library is crucial and may considerably reduce development effort and improve results. This section sets up a guideline for novice developers of LOD web applications.

The are many free and open source JavaScript libraries/frameworks. For the implementation of our Linked Widget platform, we evaluate (i) YUI[7] – a free, open source JavaScript and CSS library for building rich interactive web applications, (ii) WireIt[8] – an open-source JavaScript library to create web graph

[7] `https://yuilibrary.com/`
[8] `http://neyric.github.io/wireit/docs/`

editors for dataflow applications, (iii) Sencha Ext JS[9] – a JavaScript framework with an MVC architecture and modern widgets, (iv) GWT[10] – a development toolkit for building complex browser-based applications, and (v) SmartGWT[11] – a GWT-based framework featuring a rich palette of GUI elements.

Before deciding to use a library, developers have to address a number of questions. First, how much development time is available; is the result supposed to be a prototype or a ready-to-use product? Next, are there any requirements regarding compatibility with devices and browsers (e.g., touch devices and different browsers and versions)? What is the maximum allowable size for the loaded web resources? Which UI elements will be used in the application? Finally, developers have to read the documentation of potential libraries/frameworks to select the most suitable one for their application.

If minimizing the size of the necessary web resource (i.e., images, CSS and JavaScript code loaded for executing the application) is an important consideration, then YUI and GWT are good options because they allow developers to select the modules they would like to use on their page instead of the whole library. Other frameworks, e.g., SmartGWT, will result in the client browser loading around 4MB in total, even for a single and simple feature.

After carefully evaluating the alternative libraries/frameworks, we found that GWT meets most of our requirements. Essentially, GWT allows web developers to create and maintain complex JavaScript front-end applications in Java. In addition to its basic user interface elements, which can be inherited and extended easily, a large number of advanced elements contributed by the GWT community are available. Developers write their application in Java, which can then be compiled into optimized JavaScript by the GWT Java-to-JavaScript compiler. The compiler itself ensures that web applications run on different browsers.

Finally, making use of GWT helps developers to not only rapidly develop their prototype applications, but also makes it straightforward to finalize it into a complete product. Using other frameworks to extend already supported UI elements with new features, e.g., maximizing/minimizing a window, developers have to read, understand, and add their new source code to complicated and intricate JavaScript and CSS code of the library, which is difficult and time consuming. GWT easily supports such tasks, because its implementation language is Java – an object oriented programming language in which the concept of inheritance is much clearer than in JavaScript – an object-based language only.

4 Data Format for Lightweight LOD Applications

In lightweight LOD applications and LOD applications which include on-the-fly data processing and data transmission, it is important to choose an appropriate data format. A good decision will save a considerable amount of resources, i.e., CPU power, memory, and time.

[9] `http://www.sencha.com/products/extjs/`
[10] `http://www.gwtproject.org/`
[11] `http://code.google.com/p/smartgwt/`

We evaluated RDF, OWL, XML, JSON, or JSON-LD as potential data formats for our mashup platform. Among those, we found that JSON-LD, which "*combines the simplicity, power, and web ubiquity of JSON with the concepts of Linked Data*" [5] is the most appropriate for our needs. Since January 2014 it has been an official web standard recommended by the W3C. Compared to RDF, JSON-LD is more human-readable and takes less memory to present the same information. Additionally, in simple cases of Linked Widget interaction, where the output data model of a widget fits the input data model of another widget exactly (i.e., they have exactly the same structure or the output is a subset of the input), due to the JSON format, widgets can directly receive data from others without further processing tasks. In more complex cases, the output of a widget needs to be modified to be compatible with the input of another widget. In these cases, JSON-LD enables the platform to create a SPARQL query to perform this additional data adaption task.

5 Conclusion

In this paper, we outline a mashup platform as an open framework to manage and reuse applications on top of Linked Data. We expect that by connecting users, developers, and Linked Data, we can contribute to the diffusion and dissemination of Semantic Web technologies. We then discuss three lessons learned while implementing the platform and we consider as being useful for developers. LOD applications should follow an open and connected architecture which then should be deployed by developers on the web to be easily accessible, sharable and extensible; for lightweight applications, we evaluate JSON-LD as an ideal option for data transmission. It is concise and readable for both humans and machines.

References

1. Auer, S. et al.: DBpedia: A Nucleus for a Web of Open Data. In: 6th International Semantic Web Conference, pp. 722–735. Springer, Heidelberg (2007)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J.Semantic Web Inf. Syst., 5(3), pp. 1–22 (2009).
3. Trinh, T.D. et al.: Linked Widgets-An Approach to Exploit Open Government Data. In: 15th International Conference on Information Integration and Web-based Applications & Services, pp. 438–442. ACM, NY, USA (2013)
4. Trinh, T.D. et al.: Open Linked Widgets Mashup Platform. In: AI Mashup Challenge co-located with the 11st European Semantic Web Conference. CEUR-WS.org (2014)
5. Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., Lindström, N.: JSON-LD 1.0, `http://www.w3.org/TR/json-ld/` (2014)
6. Verborgh, R. et al.: Web-Scale Querying through Linked Data Fragments. In: 7th Workshop on Linked Data on the Web. CEUR-WS.org (2014)