Sindice Widgets: Lightweight embedding of Semantic Web capabilities into existing user applications.

Adam Westerski, Aftab Iqbal, and Giovanni Tummarello

Digital Enterprise Research Institute, NUI Galway,Ireland
{firstname.lastname}@deri.org

Abstract. One of the most important challenges that Semantic Web community faces is how to make evidence to the end users of the benefits coming from the use of standards such as RDF and OWL. In this paper we present a methodology by which it is possible to enhance existing web applications and directly deliver to the end users aggregated "views" of informations. These views are accessed by clicking on buttons which are injected into the HTML of the existing application by lightweight plugins. Upon pressing the button, e.g. next to the name of the author of a post, information is displayed which help the user perform useful tasks. We present examples in the area of social software and in the area of distributed bug and issue tracking.

Key words: search, aggregation, methodology, social software, bug-tracking

1 Introduction

Most, if not any, web applications today are conceived as a "data silo": information is always entered by the users directly accessing the system. Very seldom if ever the data comes from machine to machine communication. The Semantic Web vision promises smarter systems: at least in theory, it should be possible for a system to obtain and make useful use of information that is already out there.

Another way to see the advantages of Semantic Web technologies is to consider how they can help getting information interconnected.

Information can in fact be considered as something that should be conceptually linked but ends, in practice, distributed across several information systems. It is well known that today, even in enterprise settings, some of the most useful information is not only contained in classic document repositories but can be found distributed in social systems such as wikies, blogs, message boards, mailing lists etc. This is even more true in specific situations such as, for example, developer communities. In those cases further information silos also exists in issues and bug tracking systems, cvs respositores etc.

Semantic Web technologies have, in theory, the power of interconnect the information back to a single conceptual view and deliver it to the end user.

While the vision and the theoretical opportunities are clear to most, what has been notably lacking so far has been a pragmatic methodology to deliver visible forms of such aggregation.

In this paper we introduce "Sindice based Widgets", a lightweight methodology to immediately provide the users with extra functionalities which come from the aggregation of information across many distributed systems using Semantic Web technologies. While describing the general methodology we will refer to two main use cases that we have implemented "SindiceSIOC widgets" and "SindiceBAETLE widgets".

2 Use Cases

In this section we briefly describe some functionalities that we would like to obtain.

2.1 Creating interlinked online communities

As we said, it is often difficult to locate and interlink data across different social information spaces e.g. web forums, blogs, mailing lists etc. A very interesting feature is to be able to click on the name of a user and see which other information the user has been entering in other systems. Similarly when somebody mentions an entity (e.g. a URI) one would like to see other social systems, discussions and or users which have been dealing, e.g. commenting on, that entity.

These sort of use cases have been partially addressed, within the Semantic Web domain, by the SIOC [1] initiative. SIOC is a vocabulary that allows the representation of community generated data. Many plug-ins exist for exporting SIOC data into RDF, all of them are available on the SIOC project homepage[1].

We will see how the Sindice [2] powered Widgets (delivered by "Sindice SIOC" plug-in for the Wordpress blogging engine [3]) make this data visible and immediately usable by the end user in the form of buttons which appear next to the name of the users or next to hyperlinks (see Sec. 4).

2.2 Enhancing Bug-tracking environments

Modern software development is becoming an increasingly complex task with often dozens or hundreds of interconnected components.

Bugs and feature requests are usually tracked by software such as JIRA[2], Bugzilla[3] etc. These systems not only manage the lifecycle of bugs and issues but also allow developers to express and track dependencies. Those features work very well, but are limited to the scope of the specific bug tracking system installation. This proves to be an obstacle for complex systems which rely on

[1] http://rdfs.org/sioc/applications/
[2] http://www.atlassian.com/software/jira/
[3] http://www.bugzilla.org/

many libraries independently developed and tracked across different systems (particularly characteristic in the open source community).

As software developers log and fix bugs in their respective bug tracking systems, it would be highly desirable to automatically notify other bug tracker instances that contain related bugs. For example, there could be an issue which resolution depends on fixes applied in libraries developed outside of the current project. It would be very comfortable to have an up-to-date status from bug trackers of those libraries. Also the awareness of related bugs in external projects could be useful for task prioritization. Developers in big and popular projects could see which bugs have most dependencies in other systems that uses the software. This could be used as a suggestion for which bug needs to be fixed first.

Bug and issue tracking systems, as well as CVS allow developers to leave messages and therefore effectively become online communities which can have information integration similar to that discussed in the previous section. Similarly to the SIOC ontology, the BAETLE[4] ontology (Bug And Enhancement Tracking LanguagE) has been specifically developed to allow interoperability between systems supporting software development. BAETLE reuses SIOC concepts for social messages but adds concepts such as bugs, issues, priorities, status and others. As a result, bug and issue tracking systems using BAETLE could be made to interoperate with each other at data level.

In section 3 we will see how these functionalities are enabled by the Sindice-BAETLE plug-in for the JIRA bug tracking system.

3 Solution Architecture

The use cases we want to address all share similar requirements and involve similar steps. These can be detailed as follows (also see Fig. 1):

- RDF producing plug-ins or wrapper
- Sindice Semantic Index Engine
- Extended API Web Service
- Embedded widgets

3.1 Producing RDF

In order for information to be aggregated the online system must be made to produce RDF.

This is possible in several ways. D2RQ[5] is a project which can extract RDF out of relational databases, e.g. those that are backing the online software. These extraction tools can be deployed as a single package together with the code that will provide the enhancement for the user interface of the software (the widget

[4] http://code.google.com/p/baetle/
[5] http://www4.wiwiss.fu-berlin.de/bizer/d2rq/

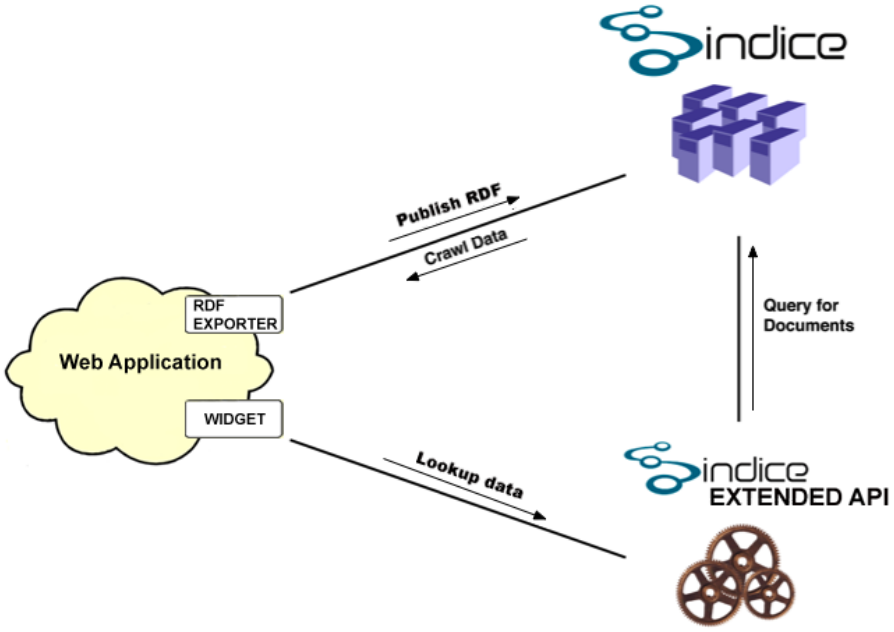

Fig. 1. Sindice Widgets Architecture.

injecting code) described in section 3.4. This is in fact the case of the Sindice-BAETLE Widget which deploys a D2RQ installation and produces BAETLE ontology based RDF describing each publicly accessible bug and issue within the installation.

3.2 Harvesting by the Sindice semantic indexing Engine

The Sindice engine provides indexing and search services for RDF documents. The public API[6], that Sindice exposes, allows to form a query with triple patterns that requested RDF documents should contain.

In order for the RDF documents to be indexed pings should be submitted. To accomplish this, Sindice exposes a special service that accepts the address of the RDF that should be indexed as a parameter[7]. The service guarantees that the document will show up in the index within 30 minutes. Furthermore due to big scaling capabilities of Sindice (currently indexing around 35 million documents), it turns to be a reliable base service for the Extended API (see Sec. 3.3).

The service output is always a document list. Therefore to extract some particular content from the documents an additional layer is needed. In our solution this is handled by the Extended API Web Service.

[6] http://sindice.com/developers/api
[7] http://www.sindice.com/developers/pingApi

3.3 Extended API Web Services

Sindice results very often need to be analyzed and refined before they can be directly used for a particular use case. In our solution, the required logic is wrapped in a domain specific web service - for example the Sindice SIOC API. The Extended API uses the basic Sindice service, but also performs several steps to clean, aggregate and cache the data. The Sindice SIOC API web service is then used directly by our widgets but can also be utilized by any other web software.

The Extended API is a back-end for the widget. All the information that the widget displays is provided by the API services. Each time a user requests information through the widget a service call is made. Based on the request parameters passed from the widget, the logic of the service forms a query to Sindice. In return, the service gets a list of documents that are loaded into a local RDF store. Next all the data to be displayed by the widget is extracted from the repository and sent back as a response (see Fig. 2).

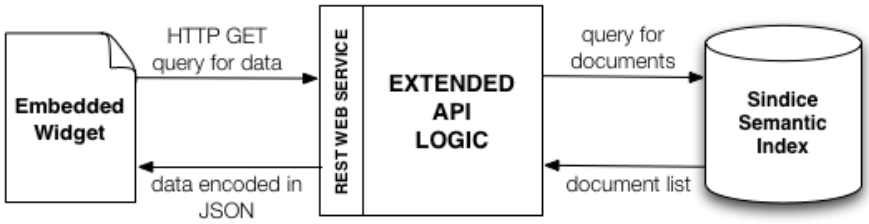

Fig. 2. Message flow in the Sindice Widgets solution.

3.4 Embedded widgets

The final part of the solution is a visual component that can be injected into some web system such as a blog or a bug tracker. The widget utilizes services from the Extended API to search for information and present it to the user. Additionally the component makes sure that information is not only consumed but also produced. Therefore it publishes and sends notifications to Sindice so it can index the new or modified resources.

The widget is a part of the solution that responds to actual user requirements regarding interconnecting communities, such as shown in the Use Cases section (see Sec. 2). In order to provide better customization we split the widget into two parts. First is a plug-in that injects code into the desired places in the specific web system. Additionally the plug-in is needed to access data stored in the system and expose it as RDF so that Sindice can index the information and provide it for others (see Sec. 3.2).

The second part is a Javascript(see Fig. 3) that renders all the content for the user or bugs and takes care of all the communication with the Extended API Web Services.

4 Implementation

The SindiceSIOC API and the SindiceBAETLE API are built on top of public Sindice API and offer a set of discovery and search services. For both of the APIs we have supplied sample widgets (see Fig. 3) that present the capabilities of services and let to demonstrate the described architecture in practice.

The current version of SindiceSIOC API[8] tracks the user activity and link mentions in posts and comments. The widget[9] that lets to take advantage of this service is built for the WordPress blog[10].

The current version of SindiceBAETLE API is still in early test phase but already enables to get related bugs based on bug URI and track bugs connected to a specified user. The current widget demo[11] supports JIRA bug tracking environment.

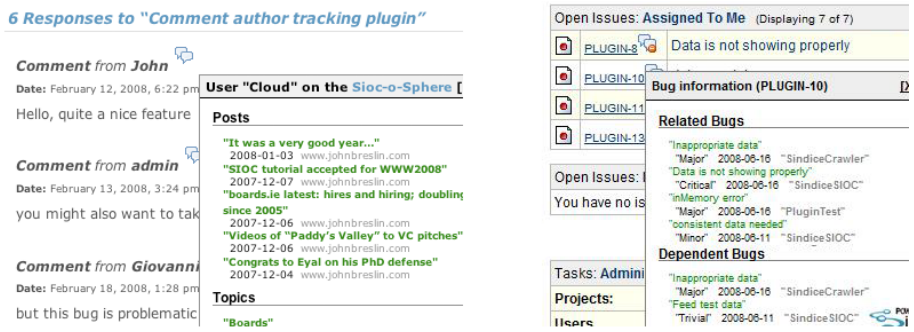

Fig. 3. Sample screenshots of Javascript popup windows from SindiceSIOC and SindiceBaetle.

[8] http://sindice.com/developers/siocapi
[9] http://sindice.com/developers/siocwidget
[10] http://wordpress.org/
[11] http://140.203.154.158:8083/secure/Dashboard.jspa

5 Related work

With respect to social networking websites there are two most notable tools that relate to our work. SezWho[12] has focused on community engagement by providing features like content rating, common user profile and community based reputation services. Similar aspects are also supported by Intense Debate[13]. Although they try to achieve a similar goal as presented in SindiceSIOC use case(see Sec. 2.1), they do not use any standard and instead rely on proprietary solutions. Also those tools do not take advantage of any common metadata standards. Therefore their capabilities are very narrow and dependent on individual success of plug-ins prepared by the companies. In case of SindiceSIOC we try to use already developed SIOC standard and consume the data that is already published on the web.

With respect to the bug tracking environments and metadata utilization, there is an interesting approach [4] for combining software code, code repository and bug tracking data.The software parses the XML output of an issue from the bug tracking system and from CVS. If an ID that identifies a bug is mentioned in the commit message, the information about the bug is automatically loaded into bug ontology model and then the bug can be associated to a particular fragment in the source code of a specific project.

Another related approach for open source software (OSS) communities is called Dhruv [5]. It provides a semantic interface that allows users to see detailed description of highlighted terms in the message posted during bug resolution in cross-links pages. This approach extracts information about software artifacts using some information extraction techniques such as noun phrases and code terms.

Although both of the the described projects take advantage of the metadata they do not put too much emphasis on portability and interconnectivity. We believe that those are very important benefits brought by the public metadata standards such as SIOC or BAETLE. Therefore, in SindiceBAETLE we try to develop a tool that can be deployed on different JIRA instances and can collect related information from different sources and present information in light weight widgets injected into HTML of existing applications.

6 Conclusions

Semantic Web is a very prominent and promising idea for the future of the current World Wide Web. However without applications that show the actual benefits for the casual Internet users it will be very hard to introduce this initiative successfully. In this paper we presented a lightweight methodology to enhance existing applications with Semantic Web capabilities.

[12] http://sezwho.com/
[13] http://www.intensedebate.com/

Interestingly, the methodology is general with respect to both the the target web applications that can be enhanced and the information domain. As microformats and RDFa gain diffusion, we believe that "skinning" them with useful information, e.g. generated links, will become a popular way to deliver Semantic Web added value to the Web users.

References

1. Bojars, U., Breslin, J., Peristeras, V., Tummarello, G., Decker, S.: Interlinking the Social Web with Semantics. In IEEE Intelligent Systems, 23(3): 29-40, 2008.
2. Tummarello, G. , Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In Proceedings of the International Semantic Web Conference (ISWC 2007), Busan, South Korea, 2007.
3. Westerski, A., Corneti, F., Tummarello, G.: SindiceSIOC: Widgets and APIs for interconnected online communities, Faculty Research Day, Faculty of Engineering, National University of Ireland, Galway, 2008
4. Kiefer, Ch., Bernstein, A., Tappolet, J.: Mining Software Repositories with iSPARQL and a Software Evolution Ontology. Mining Software Repositories, 2007. ICSE Workshops MSR apos;07. Fourth International Workshop on Volume , Issue , 20-26 May 2007 Page(s):10 - 10
5. Ankolekar, A., Sycara, K., Herbsleb, J., Kraut, R., and Welty, C. 2006. Supporting online problem-solving communities with the semantic web. In Proceedings of the 15th International Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 575-584.