End-User Programming and the Advent of Sharable, Social Machines

Evan W. Patton[1], Dominic DiFranzo[1], and Deborah L. McGuinness[1]

Tetherless World Constellation
Rensselaer Polytechnic Institute
Winslow Bldg
110 8th Street
Troy, NY 12180
[pattoe, difrad, dlm]@cs.rpi.edu

Abstract. We present a web application that allows users to reuse RDF content from existing sites (e.g., DBpedia), extract data from the social networks like Facebook and Twitter, and manipulate those data through JavaScript applets. Users combine these applets for processing and visualizing data in a Graphical User Interface, similar to scientific workflow systems. Unlike workflow systems, users are encouraged to view and interact with data interactively and manipulate it in an exploratory fashion. User-constructed applet pipelines are encoded as RDF files, allowing publication and collaboration, thus taking a critical step forward in moving to an end-user programmable social machine environment.

1 Introduction

Web 2.0 services (e.g., MediaWiki, Facebook) have transformed the web into a collaborative space for creation and consumption of content. The success of these tools is in large part to simple WYSIWYG interfaces accessible to end-users and third party applications that extend their functionality. The content created by these sites are designed for human consumption, often making it less convenient for machines to make use of the wealth of data they contain. Machine-readable annotations, notably RDFa [1] are helping to bridge this gap, but users are often unaware of such annotations. Our Semantic Application Framework (SAF) enables web consumers to identify machine-encoded resources on the web, manipulate them through a series of applets, and build and share applications to exploit machine-understandable content.

Mashups are a popular form of combining rich heterogeneous data together in visualizations, allowing for new patterns and correlations to be seen. Generating them, however, requires programming experience to create web pages, JavaScript, etc. Projects such as MIT's Simile (cf. [3]) or JavaScript libraries, like those from MITRE [4], help lower the barrier to entry, but more needs to be done so that web consumers can easily become producers of mashups. SAF is designed to bridge this knowledge gap and encourage users to combine algorithms and visualizations in a WSYIWYG manner to manipulate data and share the products with friends, family, and colleagues.

2 Use Case

Consider a group of friends interested in following social interactions on the web during the election of a new government. This group of friends who are tracking online posts may be interested in a few of different topics: *Healthcare Reform*, *Deficit Spending*, and the *Environment*. They wish to collect posts from Facebook and Twitter, filter them according to these key words, and plot the information on a timeline and a map to better visualize concerns across the country. They also want to be able to annotate their findings with possible explanations.

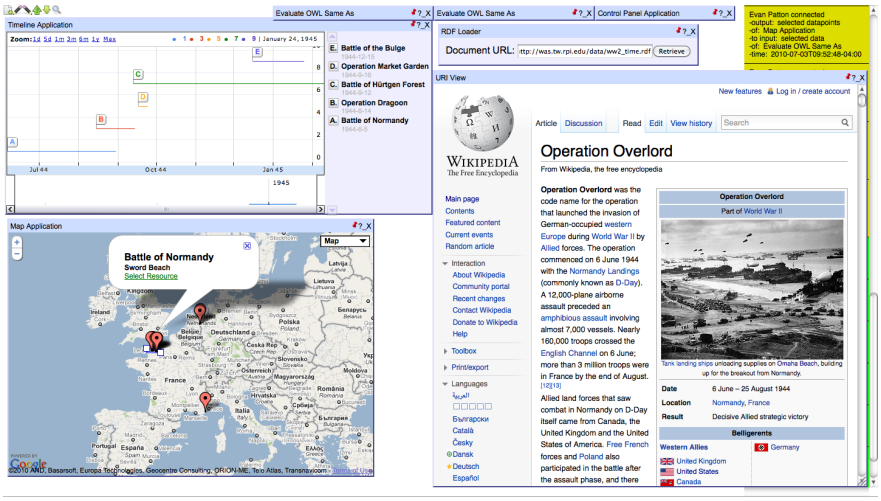

Fig. 1. Users build complex applications from applets in a workspace. These workspaces can be exported for email sharing or to be hosted on the web. Above: A SAF application combines DBpedia data with data in the WGS (See http://www.w3.org/2003/01/geo/) and OWL Time vocabularies to relate a Wikipedia article with a timeline and map.

3 Semantic Application Framework

The Semantic Application Framework (SAF) provides the necessary tools for individuals to build a web application that loads data from one or more sources and renders them via JavaScript applets. Each applet is a self-contained collection of JavaScript and server-side scripts that are used to provide interactions between the user and web data. The Facebook applet, for example, loads necessary scripts for interacting with Facebook and extracts posts as RDF using the OpenGraph Protocol (OGP) and the Semantically-Interlinked Online Communities (SIOC) vocabulary. These applets each have inputs and outputs that can be piped together, creating a mashup that others can view, interact with, and (with permission) edit.

The applets are hosted online, identified by publicly dereferenceable URIs. There are two reasons for this. First, making all of the source code publicly available allows for review of code, thus potentially improving its trustworthiness. Second, the world wide web is open, so as long as one can provide a URI to an applet, it can be included. This allows users to discover and use new applications without needing a centralized repository. Our framework includes a launcher applet that makes it easy for individuals to instantiate social networking and visualization applets.

SAF requires that an applet provide some interface for the user, but no restrictions are placed on how the applet presents its content. For example, the Timeline applet uses the Google Timeline Visualization, written in Flash, to render data encoded using OWL Time in a timeline. Most applets also serve as mediators, translating RDF content from other applets into objects the uderlying tool understands and then translating its output into RDF for other applets to consume. Applets can also act as filters, selecting resources for other applets to render to the user.

The individuals in the previous example could aggregate the data from Facebook and Twitter applets, passing that output to a filter applet that selects resources when its values for a particular property contain keywords (e.g., deficit, HCR, etc.). All data are passed along, even if they weren't marked important by any applet. This is an important distinction from workflow systems where data can be discarded at each step, and it allows applets further along in the flow to revisit data overlooked by previous applets.

3.1 Applet Development

SAF provides developers with a set of classes to make it easy to create applets. An applet begins by creating bindings, settings, and a user interface. End users link bindings of different applets to combine them into an application (see Fig. 2) in a manner similar to workflow systems (e.g. [7]) or XML pipelining systems (e.g. [5, 6]). Users then use the applet interfaces to manipulate data or identify a subset for further processing by other applets. These interfaces can take on many forms, from simple <select> elements to more complex interfaces such as the Maps applet, which utilizes the Google Maps API to render location data.

Unlike existing systems, SAF allows for users to continually explore, visualize, and link data as they build their application. By exploring data, users can find new ways to present the data to their audience. Changes to applets at different points in the workspace propagate from one to the next in real time, keeping consistency throughout the application.

In addition, applets that make use of the SAF API make known their settings and user interface structures to the system. This allows the framework to express the configuration of an applet in RDF. Therefore, complete applications can be serialized as a single RDF graph encoding the relationships between applets, bindings, settings, and interfaces. Exporting an application produces this graph in XML form, along with a stylesheet that transforms the application back into an XHTML page so that it can be hosted on the web.

3.2 End-User Programming

End-user programming is embodied in a number of different ways, from scientific workflow systems [7] to web interfaces for manipulating page content [8]. One issue with workflow systems is that various computation points are configured through simple options or by importing text-based configuration files. Once the workflow runs, researchers can view the results and make changes to the initial parameters. To provide a more useful mechanism, SAF encourages application construction to be done in the style of rapid prototyping, allowing users to quickly add applets to query, filter, and visualize data. Application construction is now a process of exploring data through linking and interacting with applets, allowing for a more exploratory development of data flows.

Consider the example given in Sec. 2. To construct this application, one would create an instance each of the Facebook and Twitter applets. These applets would have their outputs combined by a third applet, merging the two datasets. One could then take the output from this third applet and supply it to a filtering applet that looks for phrases (e.g. *Healthcare Reform*). The selected data would then be passed to an instance of the Timeline applet. Thus, all of the posts matching the filter would appear on the timeline. If one is interested in geolocation data, the timeline and the map applets could be linked together. Then, as one moves through the timeline, the map is updated appropriately to show where posts during a particular time frame originated. Ultimately, an individual can build an application with this complex behavior without having to write a single line of JavaScript.

3.3 Annotating Data

SAF already provides simple provenance representations of user interaction [9]. To provide a more complete tool for individuals to annotate existing resources, a new SAF applet, Annotate, is available that allows users to make annotations on any named resource. It does so by providing a server-side script that makes SPARQL/Update calls to a public triple store. When a user selects a resource in the Annotate applet, any existing annotations created by other SAF users will be recalled from the endpoint. This allows groups of individuals to collaboratively discuss arbitrary web resources across SAF sessions.

3.4 Access Control

To help users control what content is extracted from social networking services, SAF provides unified access control for all applets that allows users to specify what individuals or groups can access and manipulate data through those applets. SAF uses a simple access control list (ACL) style mechanism, even though semantic alternatives [11] are available, due to the fact that most applets require no forms of access control. Applets wrapping existing Web 2.0 services (e.g., Facebook) can provide custom access control objects that utilize the service's existing framework for access. This allows developers to extend the access control mechanisms in SAF with something familiar to the user.

When an access control check fails, the framework prevents the applet from sharing its data, effectively removing it from further processing. This is in conjunction with tools like the timeline or map, where patterns of behavior could be inferred and used maliciously by public viewers of an application.

4 Social Machines

Combining different applications to produce better methods of consuming data is essential to the creation of social machines [12]. To that end, all of the data passed between applets within SAF must be stored using RDF. Additionally, the entire application structure is encoded using RDF, making a SAF application a semantic web resource. This means it is possible to query application descriptions and find out which users have been actively participating in the creation and discovery process and which have not. SAF also makes it possible for users to merge data from many different social networks, make it available outside of the boundaries defined by those networks, and manipulate it as they please. These two key ingredients allow for a richer, user-driven experience with respect to application mashups and will ultimately allow users to build richer social constructs than those currently offered in Web 2.0 communities.

5 Conclusion

SAF provides a footing for developers to expand the capabilities of users interested in exploring content on the web. Users can take applets and combine them to visualize data from many different sources and share their creations with other individuals. Additionally, the new annotation system allows individuals to annotate data from existing applications for public consumption. These two advances will allow users with little to no experience in web technologies to build mashups and look for interesting patterns in web data.

SAF provides a common access control mechanism that can be mapped to from existing infrastructure, allowing users to make public applications while still retaining control over users accessing specific content. However, this mechanism is a still a first step toward the creation of unified access control mechanisms across applications on the semantic web.

6 Future Work

Scalability of this system is a primary concern. Since each application has its own products, users manipulating those products at each point causes a linear growth by application number in the amount of data that needs to be stored. This means that while it might be feasible to use one application on one gigabyte of triples, using four or five may exhaust the resources of most consumer machines.

As part of the development of this framework, we are planning on experimental tests with end users. Currently, all of the individuals accessing this system

have medium to high programming proficiency and are well versed in the Semantic Web, making it difficult to judge the complexity of interacting with the system.

7 Notes and Acknowledgements

The authors would like to thank Peter Fox for lively discussions that led to the development of use cases driving this work. EWP is funded by a Graduate Research Fellowship from the National Science Foundation. SAF is available at `https://was.tw.rpi.edu/`

References

1. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and processing. Technical report, W3C (October 2008)
2. Krötzch, M., Vrandečić, D., Völkel, M.: Semantic mediawiki. In: The Semantic Web - ISWC 2006. (November 2006) 935–942
3. Huynh, D.F., Miller, R.C., R.Karger, D.: Potluck: Data mash-up tool for casual users. In: Proceedings of the 6th International Semantic Web Conference. (November 2007) 239–252
4. Sheth, A., Gomadam, K., Lathem, J.: SA-REST: Semantically interoperable and easier-to-use services and mashups. Internet Computing, IEEE **11**(6) (Nov. 2007) 91 –94
5. Phuoc, D.L., Polleres, A., Tummarello, G., Morbidoni, C.: DERI pipes: visual tool for wiring web data sources
6. Morbidoni, C., Tummarello, G.: Semantic web pipes. Technical report, Digital Enterprise Research Institute (November 2007)
7. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: an extensible system for design and execution of scientific workflows. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 2004. (June 2004) 423 –424
8. Bolin, M.: End-user programming for the web. Master's thesis, Massachusetts Institute of Technology (2005)
9. Patton, E.W., DiFranzo, D., McGuinness, D.L.: SAF: A provenance-tracking framework for interoperable semantic web applications. In: International Provenance and Annotation Workshop, 2010. (June 2010)
10. Seaborne, A., Manjunath, G., Bizer, C., Breslin, J., Das, S., Davis, I., Harris, S., Idehen, K., Corby, O., Kjernsmo, K., Nowack, B.: SPARQL update. Technical report, W3C (July 2008)
11. Kagal, L., Berners-Lee, T.: Rein: Where policies meet rules in the semantic web. Technical report, Distributed Information Group, Massachusetts Institute of Technology (2005)
12. Hendler, J., Berners-Lee, T.: From the semantic web to social machines: A research challenge for AI on the world wide web. Artificial Intelligence **174**(2) (February 2010) 156–161