Turning a Web 2.0 Social Network into a Web 3.0, distributed, and secured Social Web Application

Henry Story
21 rue Saint Honoré
77300 France
henry.story@bblfish.net

Romain Blin
Université de Saint-Etienne, Jean Monnet
10 rue Tréfilerie
F-42023 France
romain.blin@etu.univ-st-etienne.fr

Julien Subercaze
Université de Saint-Etienne, Jean Monnet
10 rue Tréfilerie
F-42023 France
julien.subercaze@univ-st-etienne.fr

Christophe Gravier
Université de Saint-Etienne, Jean Monnet
10 rue Tréfilerie
F-42023 France
christophe.gravier@univ-st-etienne.fr

Pierre Maret
Université de Saint-Etienne, Jean Monnet
10 rue Tréfilerie
F-42023 France
pierre.maret@univ-st-etienne.fr

ABSTRACT

This demonstration presents the process of transforming a Web 2.0 centralized social network into a Web 3.0, distributed, and secured Social application, and what was learnt in this process. The initial Web 2.0 Social Network application was written by a group of students over a period of 4 months in the spring of 2011. It had all the bells and whistles of the well known Social Networks: walls to post on, circles of friends, etc. The students were very enthusiastic in building their social network, but the chances of it growing into a large community were close to non-existent unless a way could be found to tie it into a bigger social network. This is where linked data protected by the Web Access Control Ontology and WebID authentication could come to the rescue. The paper describes this transformation process, and we will demonstrate the full software version at the conference.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering

General Terms

Experimentation

Keywords

WebID, user interface, decentralized social network

1. INTRODUCTION

The generation of Web applications that emerged after the 2001 .com crash, come to be known as the Web 2.0 generation. The main emphasis of services produced since then was on social and user-generated content. This was achieved by putting together different technologies: machine-readable formats like XML (Atom, RSS, etc.) or JSON, served over plain HTTP in the REST[1] architectural style, and displayed in rich user interfaces built up using JavaScript in the browser. Formats such as Atom and RSS which led to the Blogging movement were fundamentaly distributed. However Web 2.0 somehow evolved over time into more and more centralised architectures where a few service providers dominate the market. Most content is now written and produced by people on a handful of Social Networking sites. Where blogging allowed for distributed but necessarily public conversations, the Social Networks provided a sense of cosyness by allowing people to limit the visibility of their writings to their friends or other social groups. This cosyness is partly illusory though, as the servive providers get to see all the information from every person on their network, wheras each users sees only information from his closest friends.With some Social Networks providers claiming upwards of 500 million users, this asymetry of information starts to be very problematic leading to visions of Big Brother types societies [4]. But less dramatically it also results in a loss of potential: large service providers cannot provide for every need but tend to specialise on a few simple things they can do in an industrial setting. So while Web 2.0 did succeed in bringing the social aspec of the Web to the front of people's consciousness, it ended up with islands of non communicating social networks.

In parallel, the Semantic Web grew quickly under the impulse of the W3C, the academic community, and the industry, so as to build the foundations of distributed and linked data at a global scale. It is interesting to note that the first application of the RDF[2] specification was to blogging, in what is known as RSS[3], which stood for either Really Simple Syndication or RDF site summary [1]. Around 2001 the

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

[1]Representational State Transfer
[2]Resource Description Framework, `http://www.w3.org/RDF/`
[3]`http://www.rssboard.org/rss-specification`

influential and widely deployed FOAF[4] ontology appeared too. FOAF actually made it possible and easy to create a distributed social Web where people could create machine readable profiles which would allow people to describe themselves and link to each other no matter who controlled the server. As Semantic Web tools increased in quality and the understanding of linked data grew, so the size of the data cloud started to grow in an exponential manner. It turns out then that the Semantic Web solved the problem of creating a distributed social web, but not the problem of privacy: data on the semantic web is usually considered world readable. To allow data on the semantic web to be access controlled, one would first need to solve the problem of global agent identity.

This is what Initiatives such as WebID [6] have been developed to solve. WebId ties identification into the Web by following strictly all the best practices of Web Architecture, and so working in harmony with the design of the web.

In summary, where the Web 2.0 community brought local social networks to the fore, making great strides in client/server data and user interface design, the Semantic Web community developed the foundations for extending those results into the global distributed space. On this reading taking a Web 2.0 application and refactoring it into a Web 3.0 application is an exercise that every application on the web will end up going through.

Section 2 presents LifeShare, the students' project, which we used as our starting point . Next, section 3 explains the refactoring process of LifeShare, the encountered difficulties and some of the lessons learned in this process. Section 4 is a highlight of the demonstration screencast provided with this paper.

2. LIFESHARE

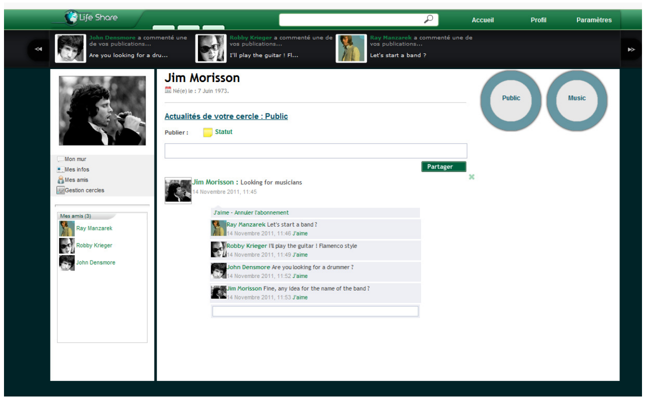
LifeShare is the product of a students project in the advanced Web programming course (SS11) at the University of Saint-Etienne. In this course, students were requested to build high-end Web applications inspired by popular services. The resulting LifeShare is a social network implementing features from mainstream social networks such as Facebook and Google+ : walls (fig. 1), picture and video sharing, etc. It is now an open-source project[5] under the MIT License. LifeShare uses a traditional three-tier architecture and was built using the following technologies:

- MySQL Database,
- JSP + Servlet,
- HTML + CSS,
- AJAX,
- Tomcat 7 server.

The project provides an high-end user interface similar to those of existing social networks. The data layer was clearly separated from the rest of the code. The application manages internally any item as part of a graph. Users, posts, photos are vertices of this graph. Vertices are connected to each other using a set of predefined relations. Privacy - in so far it is possible on a centralised network - was carefully

[4]Friend of a Friend, `http://www.foaf-project.org/`

[5]`http://bblfish.net/tmp/2011/05/11/lifeshare`



John Densmore a commenté une de vos publications... Are you looking for a dru...
Robby Krieger a commenté une de vos publications... I'll play the guitar ! Fl...
Ray Manzarek a commenté une de vos publications... Let's start a band ?

Jim Morisson
Né(e) le : 7 Juin 1973.

Actualités de votre cercle : Public

Jim Morisson : Looking for musicians
14 Novembre 2011, 11:45

J'aime - Annuler l'abonnement

Ray Manzarek Let's start a band ?
14 Novembre 2011, 11:46 J'aime

Robby Krieger I'll play the guitar ! Flamenco style
14 Novembre 2011, 11:49 J'aime

John Densmore Are you looking for a drummer ?
14 Novembre 2011, 11:52 J'aime

Jim Morisson Fine, any idea for the name of the band ?
14 Novembre 2011, 11:53 J'aime

Figure 1: User's wall on LifeShare

taken into account from the very beginning of the project. For instance, the deletion of a user account will lead to the removal of all the Web content generated by the user. This not only covers Web contents such videos, photos, posts, but also the informational content of the user's interactions such as "likes". LifeShare is a fully functional project and presents an attractive user interface.

3. REFACTORING

The modules composing LifeShare - written in the typical Web 2.0 pattern - needed to be refactored in order to turn it into a node in the larger Social Web. First LifeShare's internal data representation was using a local implicit data structure which had to be made explicit and turned into context free notation - namely RDF. Section 3.1 presents the different vocabularies used for this purpose. In order to enble LifeShare users to authenticate across LifeShare instances and indeed onto any other service they would want to, we integrated a WebID as described in section 3.2. Where LifeShare was built on the presupposition that any instance could access all data from its local datastore, the shift to a decentralised architecture required Lifeshare to be able to fetch and publish data using the simplest HTTP protocol as detailed in section 3.3. With decentralised data come issues of privacy which brought us to needing distributed access control rules. A Social Network with one user would not make sense in a traditional setting, but in the distributed space it clearly makes sense, allowing this application to targets lightweight devices such as the FreedomBox[6]. Some optimizations to prepare LifeShare for constrained deployment environments are presented in 3.4. Finally we present the lessons learned from this refactoring process in section 3.5.

3.1 Data Refactoring

The original LifeShare represented all its data as one large graph of relations connecting all types of resources it knew about: users, posts, pictures, videos, etc... All these relations were stored in a MySQL database in simple <subject relation object> fashion. The subjects and the relations were identified simply by numbers, which were then manipulated by a Java API. LifeShare being a multi-user social

[6]http://freedomboxfoundation.org/

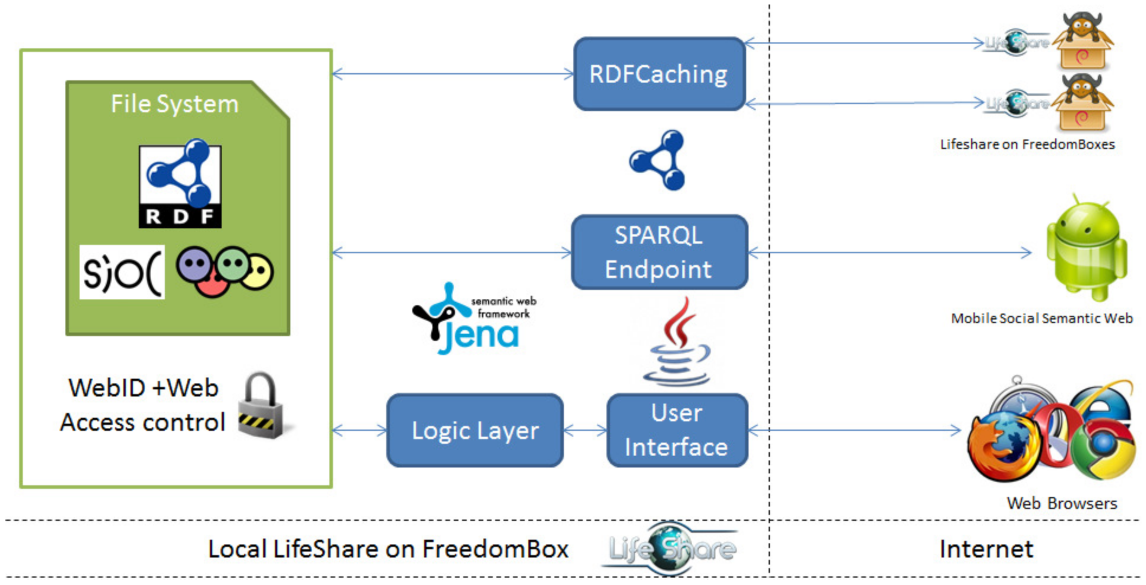

File System
RDF
SIOC
WebID +Web Access control
RDFCaching
SPARQL Endpoint
semantic web framework Jena
Logic Layer
User Interface
Lifeshare on FreedomBoxes
Mobile Social Semantic Web
Web Browsers
Local LifeShare on FreedomBox
LifeShare
Internet

Figure 2: Architecture of the refactored LifeShare

network different people could say contradictory things, so the graph could not form one consistent whole. And indeed the distinction about who said what, was encoded implicitly in the relations themselves. The <Bob knows Joe> relations would mean that Bob said he knew Joe for example.

Refactoring the one graph database required then to both make the number system explicit by replacing them with global identifiers, and to also make explicit who said what by placing statements made by each user into separate graphs. This was done by changing the implementation of the data access layer from one that made calls to the SQL layer to one that simply built in-memory models using the Jena API [3] by reading and writing files on the disk with clearly readable names in a simple understandable directory structure. Optimizations were left for later. This had a few adavantages. Firstly the data could then easily be read from the file system making it easier to visualise the process of publishing it, and making it easy to debug - the file system being one of the most trusted pieces of software on any computer. Secondly, this clarified issues of graph naming by building on well developed intuitions people have of how files belonging to different people should be organised, and how this information should be split. We used the FOAF ontology to describe the user's profile, his preferences and his social graph, which we then extended with the *relationships*[7] vocabulary when needed. The SIOC ontology [2] was used to describe users' posts, comments and more generally any generated content on the platform. We then of course replaced identifiers with URLs giving every agent on the system in the process a WebID. This then enabled the first interesting new feature to be added: universal drag and drop[5]. Taking advantage of the HTML5 drag & drop feature[8] users can now add friends by draging their local or remote profiles into their circle.

[7]http://vocab.org/relationship/.html
[8]http://dev.w3.org/html5/spec/Overview.html\#dnd

3.2 Authentication and privacy management

The WebID protocol [9], also known as FOAF+SSL [6], enables secure global, passwordless authentication. With WebID a user can authenticate to any enabled site in one click without requiring an exchange of passwords, or even the setting up of an account. Without this a distributed secure social web would be unmanageable, as one would have to create accounts on each of one's friends networks, invent a different password for each of them, and then still link all the data. The WebID being a pointer to the profile, it ties a users identity into the web of relations, which can be used to form a web of trust.

This requires that creating a Client Certificate be extreemly easy. Using the html5 keygen element a WebID Certificate can be in fact reduced to a one click experience. The presentation layer therefore had to be updated and an extra servlet added for the certificate generation process. The LifeShare server had to be enabled to work on https and request client certificates when needed.

Finally we semanticised the Access Control Layer by using the WebAccessControl[10] ontology to provide decentralized Access Control List (ACL). A person with a profile hosted by any site can take part in a group hosted by any other site. Consequently we updated the underlying part of the circle privacy management using the *read-write-web* implementation of WebAccessControl[11].

3.3 Exposing the data

LifeShare Web 3.0 aims at decentralization. To enable several instances of LifeShare to communicate with each other and with the Semantic Web, we made the data available over HTTP, fetchable with GET, queriable with SPARQL ASK and SELECTs and updatable with SPARQL update as well as with HTTP PUT and POST. As a result LifeShare is then accessible from semantic mobile applications,

[9]http://www.w3.org/2005/Incubator/webid/spec/
[10]http://www.w3.org/wiki/WebAccessControl
[11]https://dvcs.w3.org/hg/read-write-web/

for example the *Mobile Social Semantic Web* [7] application for Android. The *read-write-web*[12] server written in Scala was used to allow the files generated by the rewritten LifeShare instance to be published and edited from the web using HTTP and SPARQL.

3.4 Lightweight application

To ensure that the application can be used on lightweight devices, such as the FreedomBox[13], the refactoring process provided *de facto* several optimizations. Instead of working with the complete graph, as it was formely done in the first version of LifeShare, we now use file sized sections of the graph. The netty or jetty based framworks are very light weight and should allow further memory savings by opimising threads using Java NIO, which allows one thread to work on a very large number of open connections and files.

3.5 Implementation - Lessons learned

We present the overall resulting architecture after refactoring in figure 2, compliant with the one proposed by Yeung [8]. The biggest refactoring operation was related to moving away from the SQL datatabase to using RDF files stored on the filesystem. This required rewriting the whole data access layer - but luckily it was quite small and so the process could be finished in a little under two weeks. Refactoring the UI layer to make better use of the url identifiers was another important task which we are about to start. We want to move to having that UI layer communicate in rdf formats so as to unify the data access layer throughout the application. Adding WebID authentication and public key creation was then comparatively simple.

4. DEMONSTRATION

We will show how end users can benefit from using the Web 3.0 LifeShare in the following ways. Our demonstration uses two machines, one running the original version and one running the refactored version in order to show to the end users the difference between the two versions. The demonstration consists of two parts. The first part follows the steps of a standard user joining the system and shows the benefits from an end user perspective. The second parts presents the different refactoring steps required to transform the application, from a developer perspective. The demonstration starts with account creation and login. Using WebID in the refactored version enables users to login without username and password and allows the import of existing FOAF data for non registered users (not registered on the platform, i.e. these users already own a FOAF profile hosted on the web). In the meantime we show the slowness of the process of creating a new account on the Web 2.0 version, where users must fill a standard form and validate their email adresses. The ease of using WebID is largely demonstrated by the gain of time and the reduced effort for the end user. Next, we present how a logged user can add some friends. We first present the standard search which is common to the two versions. Then we show the highlight of our demonstration which enables users to add friends using a simple drag'n drop from a FOAF profile. This feature enables end users to add friends that are not in the LifeShare platform but own a FOAF profile. It demonstrates through a real-life example the power of combining Semantic Web vocabularies with HTML5. Once some friends have been gathered in the user profile, this latter wants to share messages on his/her wall. However not all his/her friends are concerned with the message. We present the combination of circles management and Web Acess Control for privacy preservation. The second part of the demonstration consists of a short slideshow presenting the refactoring process. We lay stress on the encountered difficulties, that we illustrate with code samples. The two parts of the demonstration can be independently presented, depending on the audience. A screencast is available at the following adress : `http://bblfish.net/blog/2011/11/11/`. It presents the original Web 2.0 version of LifeShare and its architecture, and describes the different transformations that have been applied, as well as some killer features of the refactored version.

[12] `https://dvcs.w3.org/hg/read-write-web/`
[13] `http://freedomboxfoundation.org/`

5. REFERENCES

[1] D. Ayers and A. Watt. *Beginning Rss & Atom Programming*. John wiley & sons, inc., 2005.

[2] J. Breslin, A. Harth, U. Bojars, and S. Decker. Towards semantically-interlinked online communities. *The Semantic Web: Research and Applications*, pages 71–83, 2005.

[3] J.J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. ACM, 2004.

[4] George Orwell. *1984*. Secker and Warburg, 1949.

[5] H. Story. Universal drag and drop. *The Sun BabelFish Blog*, December 2006.

[6] H. Story, B. Harbulot, I. Jacobi, and M. Jones. 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)*. Citeseer, 2009.

[7] S. Tramp, P. Frischmuth, N. Arndt, T. Ermilov, and S. Auer. Weaving a distributed, semantic social network for mobile users. *The Semantic Web: Research and Applications*, pages 200–214, 2011.

[8] C.A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, 2009.