Creating Distributed Social Networks with Linked Open Data and FOAF+SSL

[or “What’s your favourite cheese?”]

Toby Inkster BSc (Hons) ARCS <http://tobyinkster.co.uk/>

March 2009

The **#swig Cheese Lovers' Club** has decided to set up its own web site, allowing people to join the club and to share information about their favourite cheeses.

Despite all sharing a passion for cheeses, the cheese lovers are all very busy people, so don't have time to type in their favourite cheeses. Therefore they want to hook into the growing collection of linked open data, and simply select cheeses from DBpedia, the semantic web version of Wikipedia.

Also, despite all being very clever, the cheese lovers are very forgetful people, and don't like having to set up new user names and passwords for every online service they use. Therefore, they want to use an emerging technology, FOAF+SSL for identification.

The Database

Rather than use a traditional, relational database management system, the club opt to use a triple store for their information. As only a few people will be expected to visit the web site each day, it is decided that a flat N-Triples file will be adequate storage. If site traffic exceeds expectations there is a clear route to scale up to a bigger system, such as Seseme, Jena or OpenLink Virtuoso. As well as powering the web site, the club can directly expose the N-Triples file to the web, allowing it to act as the web site's (read-only) API.

The club itself can be represented in the database as a `foaf:Group` resource with `foaf:member` links to `foaf:Person` resources for each club member.

A person's cheese preferences will be represented by a `foaf:topic_interest` link to a cheese. (Technically this doesn't say that a person likes a cheese, but just that the person has expressed an interest in a cheese. That is close enough for the purposes of an example, but in real life, we'd probably want to define a more specific predicate.)

Populating the Database with Linked Open Data

The database is populated with an initial list of cheeses by performing the following SPARQL query on DBpedia.

```
SELECT DISTINCT ?resource ?label ?page ?depiction ?countrylabel
WHERE {
        ?resource
                <http://dbpedia.org/property/wikiPageUsesTemplate>
                        <http://dbpedia.org/resource/Template:infobox_cheese> ;
                <http://www.w3.org/2000/01/rdf-schema#label> ?label ;
                <http://xmlns.com/foaf/0.1/page> ?page .
        OPTIONAL {
                ?resource <http://xmlns.com/foaf/0.1/depiction> ?depiction .
        }
        OPTIONAL {
                ?resource <http://dbpedia.org/property/country> ?country .
                ?country <http://www.w3.org/2000/01/rdf-schema#label> ?countrylabel .
        }
        FILTER ( lang(?label) = "en" )
        FILTER ( lang(?countrylabel) = "en" )
}
```

Note that the `dbprop:wikiPageUsesTemplate` predicate is used as a proxy for the `rdf:type` query that might be expected in there, as in practice it seems to return a more useful set of data.

Authentication using FOAF+SSL

FOAF+SSL is a simple semantic-web-based authentication system. There are no user names and passwords to remember. You obtain a FOAF+SSL client certificate, install it into your browser, and then any FOAF+SSL-enabled site can identify you. Enabling FOAF+SSL for an HTTPS web site using Perl is a doddle. Simply install the `CGI::Auth::FOAF_SSL` module from CPAN and follow the instructions.

As soon as someone visits the club's home page, we know who they are and we can check our database to see if they're a member. Members get to explore the site; non-members are given the opportunity to join.

When someone opts to join, there are no forms to fill out: the sign-up process is as simple as clicking "Join". Using FOAF+SSL and a little SPARQL, the site can determine their name, homepage, photograph and so forth from their FOAF file. This data is cached in the site's database. (The site should rebuild its cache of personal data occasionally to ensure up to date information, but that feature has not been built yet.)

Because there is no long sign-up process acting as a barrier to becoming a member, potential members can start using the site straight away. (Indeed, if we wanted to we could remove the sign-up step altogether and simply add a person as a member as soon as they've picked a cheese to add to their favourites list.) Simple sign-up processes tend to encourage more members to join than arduous ones do.

Icing on the Cake

Cheesecakes tend not to have any icing, but here are some other noteworthy features:

- Pages are marked up with XHTML+RDFa, so even tools that don't discover the N-Triples API have access to most of the underlying data.
- Excluding `CGI::Auth::FOAF_SSL` and third-party modules, the entire site is less than 700 liberally spaced lines of Perl code.

Ideas for Further Development

- Allow members to write comments on cheeses, or start a "cheese diary" listing what cheeses they've eaten and when.
- Allow members to add cheeses even if they are not listed on Wikipedia/DBpedia. This would allow them to share information about local artisan cheeses which may not have wide distribution.
- Recommend cheeses to members based on comparing their current favourites with other members' preferences.
- Link up with a wine club (that uses a similar architecture) to recommend wines and cheeses that suit each other.

Links

FOAF+SSL

- Obtain a FOAF+SSL Certificate
- FOAF+SSL Wiki
- `CGI::Auth::FOAF_SSL` Perl module
- foaf-protocols mailing list for FOAF+SSL discussion

DBpedia

- DBpedia SPARQL Interface
- Example cheese: *Cheddar cheese* on Wikipedia and DBpedia

#swig Cheese Lovers' Club

- Club homepage
- Public (non-HTTPS) API
- Full source code

Acknowledgements

Thank you to the subscribers of the foaf-protocols mailing list, especially Henry Story, not only for the development of FOAF+SSL, but also for helping beta-test the cheese club website.