Project Report

About

WeCompare.ch

Michael Luggen

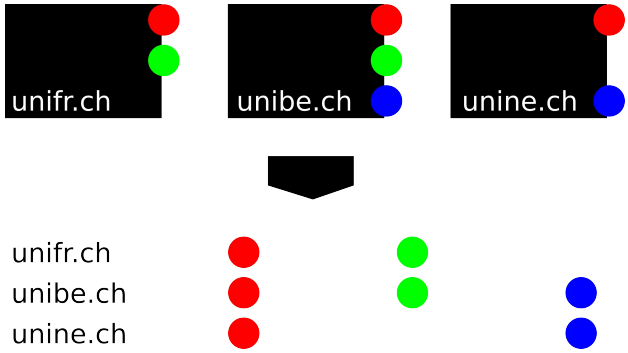
Jörg Schenker
Mariusz Wisniewski

Universities of Fribourg, Neuchatel, Bern, December 2011

Idea

WeCompare is a system allowing to make comparisons between different things on the web.

When people usually want to compare three different restaurants; they go on the web, find the website of each of the restaurants and write on a piece of paper the different attributes they are interested in. Once they have the attributes they are interested in for each restaurant, they make a comparison between all three of them and make their choice.



This goes for anything, not just restaurants. Travels, cars, clothing, apartments or computers could be another example.

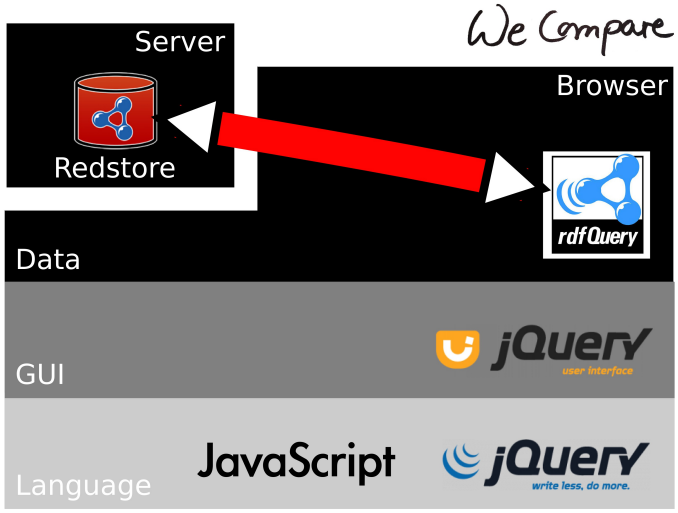
With WeCompare we wanted to speed up this process by allowing the user to have some sort of interface allowing, while surfing on websites, to take the information that is interesting and to put it in some sort of list, so that a user can retrieve it easily when he is ready to make a comparison. We also wanted to display the information in a way that helps the user make the comparison; and the main goal is that all of this should be easier and faster using our system than doing it the old way with paper and pencil.

Another interesting thing we wanted to do with this project, is to take the information chosen by the user and turn it into RDF triples that we would store on a database, so that it could be reused by our or by other systems.

Implementation

Our system is implemented as a client side application. Only the data storing is handled by the server.

We have decided that our system will be accessible by users as a webpage allowing them to surf on the internet while having a menu on the right side for the information retrieval. A user can drag and drop any information he finds useful into tags, for which he can choose a title, and once he is done with a page,



he can save the information and go on another website. Once he feels he has enough material for a comparison, he can just click a button which will open a window with all the information regarding each website displayed in a way that will help him make the comparison.

All the information chosen by the user will also be turned into RDF triples and stored on our server for later use.

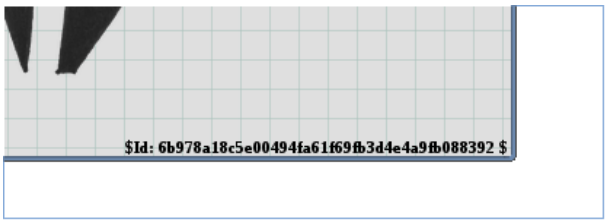
Since our system is a client side application, we decided to use Javascript as the language in which we coded the behavior of our system. The JQuery library also seemed to be a good way to improve the possibilities and to do the graphical part of our system.

For the database, we chose to use Redstore, which is a lightweight triplestore allowing us to store RDF triples and make queries on the stored data.

To communicate between our system and the database, we use rdfQuery, which is an easy-to-use Javascript library for RDF-related processing. We can use it to parse RDF embedded in our page, query over the facts it contains and work in concert with our server-side triplestore.

The website is finally hosted with apache. Apache is configured with a proxy to redirect certain URLs on port 80 to port 8989, where Redstore runs. Otherwise Javascript would not be able to connect to Redstore because of security reasons.

To be able to work easier on the project, we added a small tag on the bottom of each page. This tag shows the current git version. Like this it's very easy to see on which version you are currently working.



Further we added a cgi script on the server, which enables each developer to update the git repository on the server by simply opening the URL http://www.wecompare.ch/git/pull.sh in his browser. This helped a lot for fast development cycles, and enabled us to do a lot of testing, directly on the final product.

Finally all the images we used on the page, inclusive background, are hand drawings specially done for our needs or were created using Inkscape. We used random numbers to shuffle the images on the start page, generating us 27 different start pages. Each time the user visits us, it looks a bit different. This should make sure the page stays interesting, even after the 10th visit.

How does our system work?

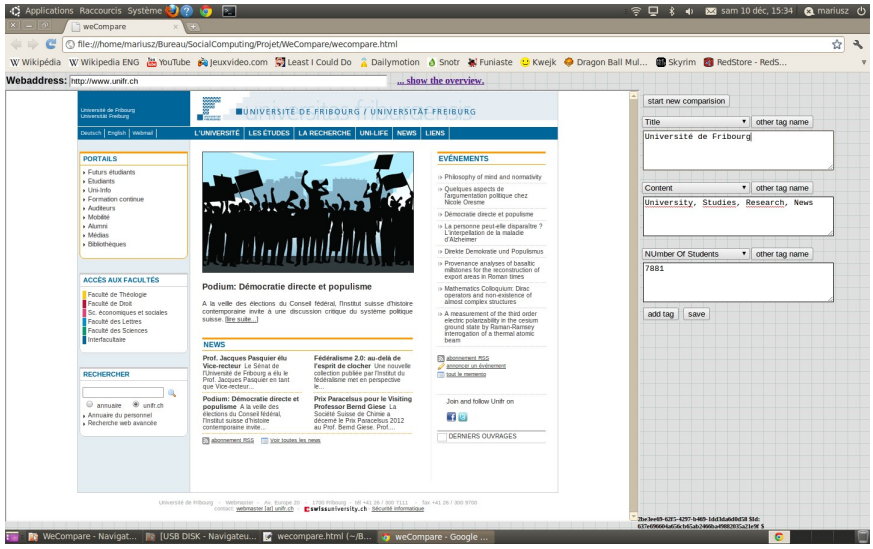
Basically, a user has to connect on our webpage:

http://www.wecompare.ch/

Once he reaches the home page, he has to enter the first website he wants to visit for his comparison. When this is done, another page is loaded.

On the new page, the user has a browser on the left which is currently browsing the page he asked, and a textarea above the browser in case he wants to enter the address of another webpage.

On the right he has the possibility to create a new tag by clicking a button. The total number of tags that can be opened has been fixed to ten. Each tag when opened is empty. The user can choose a title for its tag in a list containing all the titles possible for the given webpage. We took a list of already existing title from www.schema.org. The user can also create his own title by clicking on a button.



The user should drag and drop all the information he needs from the website into the tags. Once this is done, he just has to click on a button which will open a new window with the list for the comparison or another which will open a new website and store the previous information. All the data put into the tags is transformed into RDF triples and stored on the database for further use.

The comparison window shows a table with every website browsed and the different tags chosen. The user can then compare the different pieces of information he put in the tags. We added some features to help the user make the comparison. Lines can be reordered by the user in different ways to give him better readability.

Testing

Once we had finished with the development part of the system we put it on the web and

ran some tests, so as to see if the users would feel comfortable using it and to build up the data on our database. We asked 6 people to make a certain number of comparisons, rate the system and then we checked our database for the created triples.

Basically, the users found the system a bit complicated in the beginning, therefore we added a small guide on the main page. But once they had spent a few minutes on it, they started to feel comfortable using it. We asked them to make a few comparisons on the website of the university of Fribourg in order to add some data in the database that could be reused by the other testers.

We can see that the database is growing each time a user uses our system. New RDF triples are added, created by the users. This is the crowd-sourcing aspect we wanted our system to have. We have data concerning any kind of information and we can order it regarding the websites from which it was taken. This will allow us to build huge amounts of RDF data in the future, that we will be able to use for different kinds of purposes.

http://www.golem.de/	http://schema.org/0.9/articleSection	Von der Weiterentwicklung seines erfolgreichen Minecraft hat sich Markus Persson zurückgezogen. In seiner Freizeit hat er jetzt im Rahmen eines Programmierwettbewerbs mal eben Minicraft entwickelt.
http://www.golem.de/	http://wecompare.ch/0.1/website	http://www.golem.de/
http://www.bettybossy.ch/	http://schema.org/0.9/description	1. Ofen auf 60 Grad vorheizen, Suppenteller und -schüssel vorwärmen. 2. Öl in einer grossen Pfanne warm werden lassen. Zwiebel und Knoblauch andämpfen. Gemüse beigeben, kurz mitdämpfen. 3. Bouillon dazugiessen, aufkochen, bei mittlerer Hitze zugedeckt ca. 15 Min. köcheln. Gemüse mit der Flüssigkeit fein pürieren. 4. Fischwürfel in die Suppe geben, bei kleinster Hitze zugedeckt ca. 5 Min. ziehen lassen. Suppe würzen, in die vorgewärmten Suppenteller verteilen. Etwas Mayonnaise auf die Brotscheiben verteilen, auf die Suppe legen, restliche Mayonnaise dazu servieren.
http://www.bettybossy.ch/	http://wecompare.ch/0.1/website	http://www.bettybossy.ch/
http://www.bettybossy.ch/	http://schema.org/0.9/cookTime	ca. 40 Min.
http://www.bettybossy.ch/	http://schema.org/0.9/recipeCuisine	Bourride (Gemüsesuppe mit Fisch)
http://www.wildeisen.ch/	http://schema.org/0.9/description	Für 4-6 Personen 1 Die Rindshuft auf der Ober- und Unterseite gitterförmig nur ganz leicht einritzen; auf diese Weise kann die Marinade besonders gut ins Fleisch gelangen. Auf eine Platte legen. 2 Den Ingwer und den Knoblauch schälen und sehr fein hacken. Die Peperoncini der Länge nach halbieren, entkernen, in Streifchen schneiden und diese klein würfeln. 3 In einer kleinen Schüssel Zucker, Sojasauce und Limonensaft gut verrühren, bis sich der Zucker aufgelöst hat. Ingwer, Knoblauch, Peperoncini sowie das Sesamöl beifügen. Diese Marinade über das Fleisch auf der Platte giessen. Zugedeckt bei Zimmertemperatur mindestens ½ Stunde oder im Kühlschrank mindestens 1 Stunde marinieren lassen. 4 Die Rindshuft aus der Marinade nehmen und diese mit einem Messer leicht abstreifen. Das Fleisch auf dem heissen Grillrost rundum je nach Dicke des Stückes und gewünschter Garstufe insgesamt 12–15 Minuten braten, dabei regelmässig mit restlicher Marinade bestreichen. Die Rindshuft vor dem Tranchieren 4–5 Minuten ruhen lassen. Dann leicht schräg in dünne Scheiben aufschneiden. Die Beilage Tomaten-Mango-Raita 2 mittlere Tomaten waschen, waagrecht halbieren, die Kerne entfernen und das Fruchtfleisch klein würfeln. 1 kleinere Mango schälen, das Fruchtfleisch vom Stein schneiden und ebenfalls klein würfeln. 1 Knoblauchzehe schälen und durch die Presse in eine Schüssel drücken. 1 Teelöffel gemahlenen Kreuzkümmel, 1 Messerspitze Cayennepfeffer, 2 Becher griechischen Joghurt oder sauren Halbrahm (ca. 350 g insgesamt) und etwas Salz beifügen und gut mischen. Tomaten- und Mangowürfelchen beifügen. 1 kleinen Bund frischen Koriander oder glattblättrige Petersilie hacken und dazugeben. Vor dem Servieren die Raita mindestens ½ Stunde kühl stellen. Tipp Anstelle von Tomaten und Mango kann man auch Gurke und Frühlingszwiebeln verwenden.
http://www.wildeisen.ch/	http://schema.org/0.9/cookTime	ca. 90. min
http://www.wildeisen.ch/	http://schema.org/0.9/recipeCuisine	Asiatisch marinierte Rindshuft
http://sourcemaking.com/	http://schema.org/0.9/patternStructure	The Abstract Factory defines a Factory Method per product. Each Factory Method encapsulates the new operator and the concrete, platform-specific, product classes. Each “platform” is then modeled with a Factory derived class.
http://sourcemaking.com/	http://wecompare.ch/0.1/website	http://sourcemaking.com/
http://sourcemaking.com/	http://schema.org/0.9/patternShortDescription	Creates an instance of several families of classes
http://sourcemaking.com/	http://schema.org/0.9/patternName	Abstract Factory

Problems encountered

We had some problems coding in Javascript, since none of us ever really used it for a serious project before. We had to spend some time before getting comfortable coding in it and using the JQuery library.

Database Interfacing

One of the biggest problems we encountered was the connection between the database and the client sided application. Indeed, connecting and querying the database from Javascript was really problematic, and rdfQuery was not as useful as we expected it to be. RdfQuery helped us to create databanks to store the information for each webpage in the browser; but further we needed to find the right combination of interfaces to put this data afterward through the use of Ajax in the Redstore. We used JSON, a lightweight data-interchange format to do so.

Same origin policy

A browser security feature is that Ajax queries need to be issued on the same domain name as the origin of the website. This means we had to find a way to put the Redstore interface in the same domain space as the website itself. After a lot of tries we got this working by Reverse-Proxying through Apache2 to the Redstore. At the end the database interface is reachable at www.wecompare.ch/redstore/.

IFrame URL fetching

We have planned that the user can browse in the IFrame in which the current website is displayed. If the user clicks on a link the according website is loaded, and the system saves the data from the old site in the triple store. Unfortunately this is not possible. Because of the browser implementations which don't allow to observe the behavior in the IFrame.

Without any solution to this problem the user needs to put himself the actual URL in the Addressfield at the top of the site.

Limitations

No protection of the database !!

Improvements

Our system offers basic functionality; we could however improve many things.

The first thing is, except for the titles possible for each tag, we do not really reuse the data we stored on the triplestore. We thought about how to reuse the data another user already selected for a given webpage, but couldn’t really agree between us on how this should be done. This data however should be reused in a way improving the quality of the service each time a user uses our system.

What we learned

We all got to learn how to code using Javascript, which is a good experience. We also learned how to manipulate and query RDF data on a database.

We now have gained some experience with each element we had to use in this project such as Jquery, Redstore and Apache.

One of the main things we learned is that it is not easy to create a system that will be pleasant and useful for people to use; and that even if the concept seems simple; it really isn't that simple to put it in practice.