Internship report

Alipi Project

El Hatimi Taha
Internship supervisor : Mr. TB Dinesh
Tutor : Mr. Allali Julien

8 novembre 2012

Table of contents

Presentation

Introduction

As part of the 2nd year teaching at our engineering school, this internship is intended to let us put into practice the technical skills and the knowledge we acquired during these two first years.

Starting from June, this internship lasts for a duration of three to four months. Contrarly to some other engineering school, Enserib-Matmeca chose this minimal duration of three months so that we can get a real experience in a formal environment of a structured company.

Actually, in three or four months, we have a plenty time to, first of all, learn about the company structure, and be integrated to working team we will work with, before getting started in the technical part of our internship.

As far as I'm concerned, since we have the opportunity to do our internship in a foreign country, I chose to apply for each internship offer in an English speaking country. The purpose of this choice was mainly to develop my english speaking abilities, and get a new living experience in a different country with a new culture, which can only be formative for my personality.

Knowing the agreement that our school have with some universities, I first applied for an internship at Oulu university in Finland. It's only when I spoke to Mr Fouresti, our Database teacher in the computer science department, that I knew about the opportunity of doing my internship in Servelots in India. Since I stilled didn't have any response from Oulu University, I finally opted for this three months indian experience in Bangalore, especially that the open source domain has always been a point of interest for me, and that internship subject regarding the web technologies is what I was looking forward to working on.

The Company

Servelots is a web service provider for Small to Medium Enterprises which was founded in 1999 by a group of Computer Scientists who wanted to provide a highly cost effective but user friendly software for SME's with a special focus on the organizations working in the social development sectors.

The company develops and distributes Free and Open Source toolset that enables the organizations to manage their information and communication without software development dependency.

Servelots has a social responsibility arm named Janastu. Janastu provides IT services for the voluntary and civil society organizations. More information on Janastu can be found at :

http ://www.janastu.org

Projects and role in the company

As servelots is a start-up with about ten employees (including the interns), there are actually, literally speaking, no departments. Nevertheless, we can distinguish many projects that the company is working on. In addition to some Androïd applications, the main projects are :

- In the first place Alipi, which has been discussed for two years in the RMLL - Rencontre mondiale de logiciels libres, in a presentation given by Mr Dinesh - my internship supervisor. Alipi is a generic Web-framework for developing and rendering narratives that assist in accessing Web-content across cultural boundaries. Re-narration Web is therefore about Web-accessibility for digital Inclusion or e-inclusion. Indeed, Servelots has been working on the Alipi project, in collaboration with the IIT of Hyderabad, and other entities around the word, for more than two years. But the current version of this project, has just been developed during this year, following the new specifications of the project, dropping the first year work. Saying that, it doesn't mean that this previous work was useless. In fact, It is only with sharing ideas and trying several perspectives that we can get into the right specifications of a project, especially when many entities are involved, as it is the case for Alipi.

- In the second place, a new project also discussed in RMLL 2012, is the Hampi Digital Heritage Project. A digital heritage site is where you enter a heritage site by the click of a button without actually being at the heritage site. One gets to explore in great detail the murals, carvings, architecture and videos of cultural heritage. A lot of technology for this involves stitching photographs of high resolution and techniques to render these in 2D and a 3D that simulates experience of a physical walk. In the Indian Digital Heritage Project, a digital heritage walk is to be experienced in a web browser, and tools being developed are to enable people to set up their own digital heritage sites in the future. One significant aspect of interest, in a heritage walk, is having narratives of the many artifacts at the heritage site. Often these narratives are of mixed-media, that pull up

related videos, scholarly discussions or similar artifacts of the same period. Effectively this work will involve a tedious job of collating all related bodies of work, else where in the world, for every artifact of interest.

The Hampi project was actually meant to be the subject of my internship. However, since it is a new project, I wouldn't have worked in a team, especially that Servelots hadn't got the funds from the government of India for this project yet. That said, the Servelots team thought that it would be more interesting if we - the three interns from Enseirb-Matmeca, work on the Alipi project with the rest of the project team. The idea is actually to be able to benefit from our presence to make a great progress on Alipi.

Abstract

As in any other internship, the first step is to get familiar with the documentation and and the code which I will be working into. Since it is an Open-Source project with non-exact specifications, many ideas of developing Alipi were brought into discuss. Thus, we had many meetings on the two first two weeks, presenting some other Open-Source projects from which we could get some ideas for Alipi, or discussing about the opportunities we had to improve the actual version of Alipi. Mainly, we were discussing about Semantic Web technologies, and as no one was familiar with that, we were asked to collect informations and read documentations (specially the SIOC project one which provide a specification of the W3C recommendations for the semantic web), so that we could decide about which API or libraries we could use for Alipi. Before giving an abstract of the most important meetings we have, here is a short description of the actual version of Alipi :

The problem of Alipi actually was mainly to provide it to a large community of users. Otherwise, the use of it would still limited to the developers community which are aware of it. This said, Servelot's developers were working on providing a Javascript addition which will contain the tools for Re-narration web page, add links or objects (audios, images...) in order to simplify the original's web page content. The idea is to identify each element by its xPath so that it can be modified independently. The principal programming technologies used to provide this tools are Python, Javascript and jQuery for the client side, and Wsgi as an interface between web servers and web applications. You can try re-narrating a web page using this link :

http ://alipi.us

The tools available provide sharing options and accessing to other re-narrations of the same web page.

Among these meetings, one was a discussion of The XrayGoggles project of Mozilla, which allows to mix a web page and change data so that it could be shared into a blog. Since it is an Open Source project, the idea was to get inspired from their way of detecting elements and mixing them up, to enrich the alipi project.

Another point is the way they are supplying their service : by providing a Javascript function which, when added to bookmarklet bar, could be activated on a web page by clicking on it. This was a really good point we could add to Alipi project, so that it would be possible to add the Alipi toolbar by pushing the bookmarklet button, without copying the URL to «http ://alipi.us».

I spent few days on xRayGoggles code, trying to get inspired form for it to improve some functionalities on Alipi, before my internship supervisor told me to drop it out, since each file was thousands of almost non-commented lines.Another important meeting was the one where we learned about Selenium, a web browser automation, which will be used on the testing part.

Technical work and technologies used

Tasks Description

We actually had some other meetings, sometimes with persons from other companies. It was for sure constructive, as a python presentation, and a debrief about Hampi project, which was supposed to be my internship subject in first place.

As it was said below, the internship subject I was supposed to work on is the Hampi project, on which we had a meeting. It is actually a Digital Heritage project of reproducing Hampi's sites as murals or temples in 2D or 3D so as it could be visited in a virtual way. My part of word would have consisted to use semantic web technologies to link this Hampi site to other digital heritage site around the word to build a cultural community of Heritage website.

Nevertheless, as no one was familiar with those technologies, we were advised to get started with the Alipi project and extend the existing using semantic web technologies. It was actually meant that all of us were working on almost the same tasks. Indeed, the aim was to get familiar with all the sides of the project and the web technologies used.

Once this first step of the internship was over, we asked our supervisor to assign to each of us specific tasks. Thus, after having a meeting, the task which was assigned to me consisted to extend the use of Alipi project to ≪Comments≫. The aim is actually to enable adding, in addition to re-narrations (translations), comments. So mainly the idea at start is to add to the Alipi Javascript addition a button which will lead to a pop up enabling the user to write his comment and post it into a blog.

First of all, I spent some time on reading Javascript, Jquery and MongoDB's documentations to get familiar with the existing code, in which there is almost no comment that could explain it. Then, I started working on the client side. I was able to add the comment button which leads on the mouse click to the

popup window where the user can write his comment. The next step was then to work on the server side so that data could be stored into MongoDB database, and then pushed into the browser to be shared in a blog. I also used an API to fetch the comments metadata and generate tweets which can be accessed in a web page as feeds. This was mainly the first task of my internship.

Naturally, to complete the work, the comments has to be shown on the concerned web page. Indeed, the idea was to provide a functionality which can display easily the comments written about a paragraph, an image, etc... in a sidebar displayed on a button click. The web technologies used for this task are almost the same used for the previous one, that are mainly Jquery, Javascript, and Python for the server part of fetching the data from the MongoDB database.

As I mentioned in the abstract of the Alipi project, the actual aim is to provide this tool to the largest community of users. Indeed, following the W3C recommendations, the most efficient technologies to reach this goal is to use the Semantic Web technologies.

During the first month, and before each one took specific tasks to work on, we were mainly trying to get familiarised with those technologies. Actually, when a user try to search for some information on Google for example, the automation doesn't take into consideration the semantic meaning that the user is seeking. The research results are automatically fetched regarding the exact spelling of the words mentioned on the query. Here comes the idea behind the perspective of the Semantic Web. Indeed, if we can link the data to semantic metadata so that every information on the web would have a sense that will link it semantically to other data, we will be able to build a graph that will interlink the information, regarding the semantic metadata related to each information on the web.

This graph, regarding the W3C recommendations, had been specified as the RDF extension. Representing the information in this format allows to provide all the needed metadata, and store it into an RDF Store - the repository for the Python RDF library - and be able to retrieve the Datas using SPARQL queries.

The task of one of the other interns was actually to switch from the actual version of Alipi using MongoDB database into the RDF and Sparql technologies which will allow the use of Semantic Web. However, according to my supervisor, this was like an experiment using of those technologies. Thus, for my work, he advised me to use the same technologies used on the current version of Alipi, that is to say MongoDB.

Though, in the purpose of making my work extendable, I thought about ending my work with providing an ontology schema which will allow, in the future, to use the comments functionalities in Semantic way, and switch easily from a MongoDB storage into the RDF Store graphs.

Client side

Javascript technologies

While HTML is used to store the content and formatting of a web page and CSS encodes the style of how the formatted content should be graphically displayed, JavaScript is used to create rich effects or rich web applications. However, the umbrella term "JavaScript" as understood in the web browser context contains several very different elements. One of them is the core language (ECMAScript), another is the DOM (Document Object Model).

The Core Document Object Model is standardized by the W3C. It defines language-agnostic interfaces which abstract HTML and XML documents as objects and mechanisms to manipulate this abstraction. Among the things defined by the DOM, we can find :

- The document structure, a tree model, and the DOM Event architecture in DOM core : Node, Element, DocumentFragment, Document, DOMImplementation, Event, EventTarget, ...
- A less rigorous definition of the DOM Event Architecture, as well as specific events in DOM events.
- Other things such as DOM Traversal and DOM Range.

Every web developer has experienced that the DOM is a mess. Browser support uniformity varies a lot from feature to feature. The main reason for this situation is the fact that many important DOM features have had very unclear, if any, specifications. Also, different web browsers have added incompatible features for overlapping use cases (like the Internet Explorer event model). The current (as of June 2011) trend is that the W3C and particularly the WHATWG are defining older features in detail, in order to improve interoperability. Following this trend, browsers are improving their implementations based on these specifications.

One common, though perhaps not the most reliable, approach to cross-browser compatibility is to use a JavaScript library. These libraries abstract DOM features and ensure their APIs work similarly in different browsers. Some of the most widely used frameworks are jQuery, prototype, and YUI.

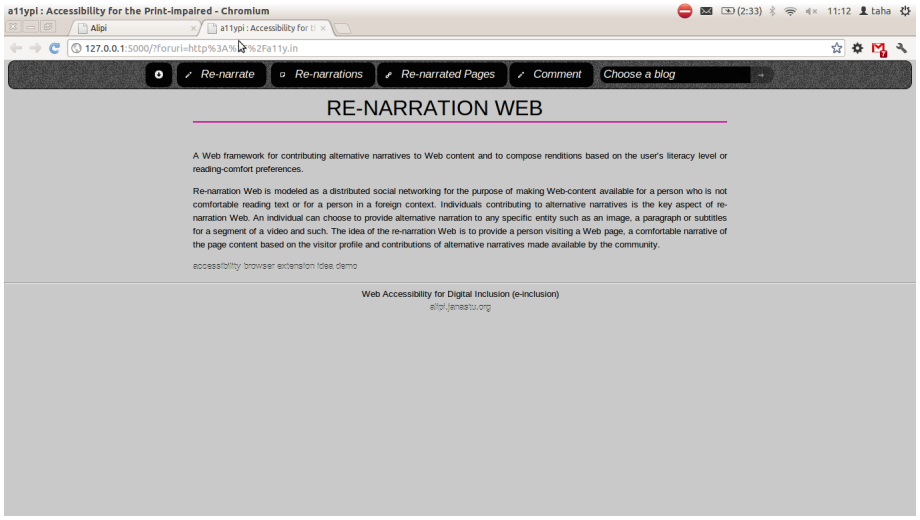
The framework used for this project client side is Jquery, choosed for its large documentation and the several oportunities it offers in term of widgets that will be we used for our web application.

The Alipi bar

The purpose of using the Alipi re-narration or comments is to be able to use on all the web pages. Therefore, several ways were discussed to reach this goal. The first idea is to create a javascript extension that allows the addition of a javascript bar wherever we want to edit a web page, ie to add a re-narration or a comment and post it into a Blog. Another idea was to provide an interactive Alipi toolbar that would be integrated to the browser, as a Yahoo search toolbar for example, so that the functionalities needed would have been always available and shown in the browser in use. As this idea came later, it would be implemented by another intern, with the help of a Master student from the IIT of Hyderabad. At this point of work, the first method will be described and used for the rest of the work. Indeed, using Javascript, it is possible to manipulate the HTML body using the Jquery selection and methods, with$, instead of the DOM, and then to add the code needed for showing the buttons for the *Alipi* bar. Below is the code that allows the addition of the main *Alipi* bar :

```
javascript:(function(){location.href=
'http://y.a11y.in/web?foruri='+encodeURIComponent(location.href);})();
```

Here is a screenshot showing the result of adding this bar on a specific web page :



jQuery

jQuery is not a language, but it is a well written JavaScript code. As quoted on official jQuery website, "it is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development."

jQuery is very compact and well written JavaScript code that increases the productivity of the developer by enabling them to achieve critical UI functionality by writing very small amount of code.

- It helps to improve the performance of the application
- It helps to develop most browser compatible web page
- It helps to implement UI related critical functionality without writing hundreds of lines of codes
- It is fast
- It is extensible – jQuery can be extended to implement customized behavior

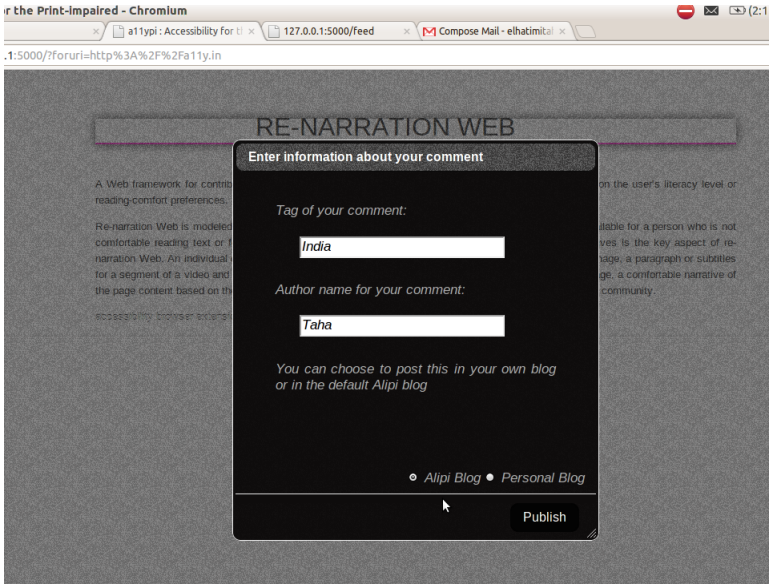
Other advantages of jQuery are :

- No need to learn fresh new syntaxes to use jQuery, knowing simple JavaScript syntax is enough
- Simple and cleaner code, no need to write several lines of codes to achieve complex functionality

As explained before, the aim from this part of the project is to enable adding comments about a web page. In that context, jQuery is meant to be the perfect tool for showing pop-up windows that will allow users to add their comments about a specific element on the web page, and the metadata related to its, as the langage, the community, the author...

The Javascript functions using jquery, and specifically the widget *Dialog*, allows to pop up a window with the text area for the comment text, and then another one with the metadata fields that will be filled by the user. To show the available comments in a web page, another function had also been written. The code in the ANNEX 1 is an example of the function that pop ups the window with the metadata fields.

Here is a screenshot of the result in the browser, as soon as this function is called on click :



The JSON Data Format

Once the metadata are filled in by the user, it has to be stored in some specific format that can be used later to retrive the information. The two perspectives that were discussed are the XMLl format and the JSON format.

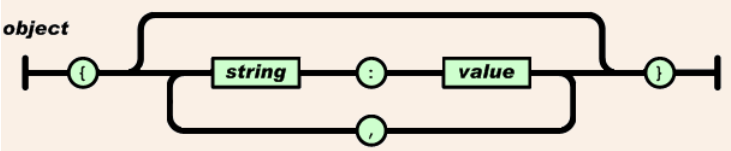
The documentation I have read showed me that XML is not well suited to data-interchange, much as a wrench is not well-suited to driving nails. It carries a lot of baggage, and it doesn't match the data model of most programming languages. There is another text notation that has all of the advantages of XML, but is much better suited to data-interchange. That notation is JavaScript Object Notation (JSON).

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON has actually the same benefits of simplicity, extensibility, interoperability and openness, and the ease of converting XML to JSON makes JSON ultimately more adoptable.

JSON is built on two structures :

- A collection of name or value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.



Here is an example of the Json Format that results after when a user writes a comment :

```
"0": {
    "xpath": "/SPAN",
     "about": "http://a11y.in",
     "author": "Taha",
     "data": "I like this website but the title should be changed",
     "com_id": 0.5331782348665307,
     "blog": "http://testalipi.blogspot.com/2012/11/
a11ypi-accessibility-for-print-impaired.html",
     "tag": "India",
     "bxpath": "//*[@id='post-body-7069836616878483086']/p",
      "_id": "509b46ed1d41c80ad9f4f4ef",
     "type": "comment"
   },
```

- *xpath* : the exact xpath of the element commented
- *about* : the web page commented
- *author* : the author of the comment
- *data* : the text of the comment
- *com id* : the ID of the comment
- *blog* : The blog where the comment will be posted
- *tag* : the tad of the comment (either a Language, or a Community...)
- *bxpath* : the xpath of the comment on the blog where it's published
- *type* : A type to distinguish comments from the re-narrations

Server side

Python

After giving a brief abstract of the client side of the project, let us describe the Client-Server interaction. The programming langage used for coding the server part is actually Python.

Python is a high-level programming language - a language that does not deal with the concepts of pointers and memory storage/allocation unlike C. Python is also a general purpose programming language - a programming language that can be used widely in many different applications and software, especially in our case in the web development.

Actually, the developers at Servelots, the company where I did my internship, were already working with Python. To their opinion, Python is fast enough for their projects and allows to produce maintainable features in record times, with a minimum of developers. Indeed, I had almost the same opinion about Python, since it provides a rapid development (no declaring types...) shorter code, which is easy to read and also to learn, and is very portable. Moreover, Python provides tons of librairies for every kind of use, and is hugely powerful.

That said, we are using Python scripts to launch our application on the web, and specifically to add the Javascript scripts to the HTML body, so that the Alipi Bar can appear on the top of the current web page. Some of those Python scripts are in the ANNEX 2.

WSGI

At first, the interaction between the Alipi framework and the web server was based on WSGI files. WSGI, the Web Server Gateway Interface, defines actually a simple and universal interface between web servers and web applications or frameworks for Python. Indeed, WSGI was created as a low-level interface between web servers and web applications or frameworks to promote common ground for portable web application development.

The WSGI has two sides : the "server" or "gateway" side, and the "application" or "framework" side. The server side calls the application side, providing environment information plus a callback function (for the application to use to convey headers to the server side), and receiving web content in return.

However, on advice from my internship advisor, I have been told to integrated the WSGI files and its functionalities, as the data storing in the MongoDB database, into the server file *alipi.py* which also contains the others script that launched the framework, as explained before above.

Blog posting and feeds

The purpose here is to describe the data flow after being stored in the JSON format. Actually, before even storing the data on the MongoDB database, which is used for the whole project, the JSON data flow, as represented on the section above, will be directly pushed to be posted in a blog. For now, it is only possible to publish the comment, so as the re-narration, into the Alipi blog, which is the public blog designed for the re-narrations and comments publishing. The functionality that would allows the user to publish his comment or his re-narration into his own blog is still not implemented. This functionality which, which is also implemented in the python file *alipi.py*, is accessed from the javascript file *ui.js* - that implements the client side - once the function **postForm** is called :

```
postForm: function()

     {
a11ypi.com['type'] = 'comment';
a = a11ypi.getParams();
a11ypi.com['about'] = decodeURIComponent(a['foruri']);
a11ypi.com['title'] = document.title;
a11ypi.com['author'] = $('#authcom-select').val();
a11ypi.com['data'] = $('#comment-text').val();
a11ypi.com['tag'] = $('#tag-select').val();
console.log(a11ypi.com);
$.post("http://127.0.0.1:5000/post/", a11ypi.com, function(data){});
 },
```

In the function above, we are retrieving the data filled by the user in the textareas, and filling the JSON array, so that it can be pushed as soon as the POST method is called as shown on the last line of the function. Actually, this line is referencing the */post* in the server file *alipi.py*.

Indeed, for the implementation of our web application, we used a microframework named **Flask**. The *micro* means Flask aims to keep the core simple but extensible. Flask was really helpful with the POST or GET methods. For example to refer the */post* from the function above, I just needed to add this line before the function concerned in the python file :

```
@app.route('/post/', methods=['POST'])
```

The function that allows the publishing of the comment in the blog is below in the ANNEX 3.

In addition to the Blog posting, the data filled by the user and stored in a JSON format will also pushed into a feeds page. Those feeds are meant to be like an up to date page which will display every new comment or re-narration.

Here an example of the feed generated from the Json showed above :

```
@Taha had commented http://a11y.in//SPAN at
http://testalipi.blogspot.com/2012/11/a11ypi-accessibility-for-print-impaired.html
//*[@id='post-body-7069836616878483086']/p for India
```

MongoDB

The database used for this project is MongoDB. First of all, here is a brief abstract that explains the reason that leeded the Servelot team to choose MongoDB for the Alipi project. Actually, MongoDB simplifies development. Data in MongoDB is stored in JSON-like documents with dynamic schemas, providing flexibility during the development process.

Moreover, MongoDB is Document-oriented :

- Documents (objects) map nicely to programming language data types
- Embedded documents and arrays reduce need for joins
- Dynamically-typed (schemaless) for easy schema evolution
- No joins and no multi-document transactions for high performance and easy scalability

The python function that allows the storing into the MongoDB database from the Json format is available in the ANNEX 4.

Extensibility to Semantic Web

The Semantic Web, as originally envisioned, is a system that enables machines to "understand" and respond to complex human requests based on their meaning. Such an "understanding" requires that the relevant information sources be semantically structured. The Semantic Web is regarded as an integrator across different content, information applications and systems. It has applications in publishing, blogging, and many other areas.

The standard promotes common data formats on the World Wide Web. By encouraging the inclusion of semantic content in web pages, the Semantic Web aims at converting the current web dominated by unstructured and semi-structured documents into a "web of data". The Semantic Web stack builds on the W3C's Resource Description Framework (RDF).

If we consider for example these sentences :

- The Beatles was a popular band from Liverpool.

– John Lennon was a member of the Beatles.
– "Hey Jude" was recorded by the Beatles.

Sentences like the ones above can be understood by people. But how can they be understood by computers ? Statements are built with syntax rules. The syntax of a language defines the rules for building the language statements. But how can syntax become semantic ?

This is what the Semantic Web is all about. Describing things in a way that computers applications can understand it. The Semantic Web is not about links between web pages. It describes the relationships between things (like A is a part of B and Y is a member of Z) and the properties of things (like size, weight, age, and price).

The RDF (Resource Description Framework) is a language for describing information and resources on the web. Putting information into RDF files, makes it possible for computer programs ("web spiders") to search, discover, pick up, collect, analyze and process information from the web.

Several defined ontologies are already defined and validated by the W3C on the web. It actually helps developers to use those ontologies or even defining their own ontologies by using available ontologies and integrating them in their new ontology. Two of the most popular popular ontologies used on the web nowadays are FOAF (Friend Of a Friend), wich defines a vocabulary for expressing personal profile and social networking information, and DC (Dublin Core) which has some predefined properties for describing documents.
Here is an example of an RDF file using the DC ontology :

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc= "http://purl.org/dc/elements/1.1/">

<rdf:Description rdf:about="http://www.w3schools.com">
   <dc:description>W3Schools - Free tutorials</dc:description>
   <dc:publisher>Refsnes Data as</dc:publisher>
   <dc:date>2008-09-01</dc:date>
   <dc:type>Web Development</dc:type>
   <dc:format>text/html</dc:format>
   <dc:language>en</dc:language>
</rdf:Description>

</rdf:RDF>
```

As said in the chapter **Abstract** above, the aim of the project Alipi now

is to swith into Semantic Web. That is said, we need to move from a dataBase storing on MongoDB into a RDF data format, regarding a specific ontology. And the access to those Rdf files will be ensured by retrieving the RDF files using SPARQL Queries, which is a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. That was actually the internship project of another intern, but I also discussed with my internship supervisor about the ontology for the comments I worked on. The idea is now to extend all the work into semantic web ontologies. I actually worked on an ontology, but the purpose is to be able to write a generic ontology that will suit both the re-narrations and the comments, and even an another up-coming project if possible. Indeed, the purpose is to make the alipi semantic web project the most extensible as possible, which would facilitate the futur tasks.

Conclusion

Knowledge and Skills acquired

It is known that in university we learn more the theory part of the course studied, but during an internship we have the opportunity to learn the practical part and acquire more skills and experience. And in addition to that, since I had the chance to do my internship in India, I've also putten my English into practise. Moreover, it was really formative to acquire this technical knowledge in English.

I got the habit of searching of the documentation in English instead of French, and I found this very useful, since there are much more helpful computer science forums and blogs in English than in any other language. Concerning the technical skills I acquired, the subject of this internship matches my exact point of interest in computer science, which is the Web development and Web service.

Actually, I've been able to learn other technologies than the one learned in school, as Javascript, Jquery, Python web libraries, etc... and especially the use of Semantic web technologies, which I consider as the future of the Web.

Review on the abroad internship experience

This internship was definitely not just a technical experience to put into practice my computer science knowledge. Moreover, this human experience will be forever engraved in my memory. Indeed, the Servelots team was so kind and friendly to us that I never felt the stress of working in an office. I didn't even need to ask them for help when I got blocked on some work, most of them were always checking on me, asking how are things going...

Moreover, the experience was not limited to our work in the office, we shared every single experience with each other during those three months, especially when we travelled together. An atmosphere of good mood was always present.

I would have never lived this intense experience without the help of Mr Dinesh. He actually welcomed me in his house, and made me feel like I was at home. It is true that sometimes missed my place and my friends in France, or my family in Morocco, but Indian people are so warm-hearted and accepting that they can make you feel good even in the hardest moments.

References

- http ://www.w3schools.com
- http ://www.wikipedia.com
- http ://www.w3.org/standards/semanticweb/
- https ://developer.mozilla.org/en/docs/
- http ://sioc-project.org/ontology
- http ://janastu.org

Annexes

```
publishComment: function(){
if (a11ypi.target == false ) {
    var publishCom_template =
'<div id="commenttargetoverlay" title="Enter information about your comment" class="alipi ui
'<label id="com-lab" class="alipi" >Tag of your comment: </label> '+
'<input id="tag-select" class="alipi" placeholder="Type community/language name"/>'+
'<img id="tag-img" src="http://localhost/wsgi/images/db_loading.gif" /> '+
'<label id="com-lab2" class="alipi" >Author name for your comment: </label> '+
'<input id="authcom-select" class="alipi" type="text" placeholder="John" /> '+
'<div id="blogset-com" > You can choose to post this in your own blog or in the default Alip
'<p id="tar-p" ><input id="com-check1" class="alipi" type="radio" name="blog" /> '+
'<label id="com-lab3" class="alipi" > Alipi Blog</label><input id="com-check2" class="alipi'
'<label id="com-lab4" class="alipi">Personal Blog</label></p></div>';

    $('body').append(publishCom_template);
    $('#commentoverlay').css('display','none');
    a11ypi.target == true;
}
    $(function() {
$( "#commenttargetoverlay" ).dialog({
    height:500,
    width:450,
    modal: true,
    buttons: {
Publish: function() {
    if ($('#tag-select').val() == '' || $('#authcom-select').val() == '' || ($('#com-check1'
alert("please fill in all the details");
    }
    else {
if($('#your-check').attr('checked') != undefined)
{
    alert("please fill in all the details");
}
else{
    $(function(){
```

```
$('#commenttargetoverlay').dialog('close');
$('#pub_overlay').slideUp();
$('#element_edit_overlay').hide();
var success_com_template = '<div id="success-com-dialog" title="Posting your comment" class=
    '<p style="color:#aaa"><b>Please wait !!!</b></p><p style="color:#aaa">Your contributio
$('body').append(success_com_template);
$(function() {
    $( "#success-com-dialog" ).dialog({
modal: true,
    });
});
    });
    a11ypi.postForm();
    $('#success-dialog').dialog('close');
    $(document).ready(window.location.reload());

}
    }
}
    },
```

ANNEX 1 : *Javascript function that popups the window which allows the addition of the comment's metadata*

```
script_test = g.root.makeelement('script')
script_edit = g.root.makeelement('script')
g.root.body.append(script_test)
g.root.body.append(script_edit)
script_test.set("src", conf.APPURL[0] + "/server/ui.js")
script_test.set("type", "text/javascript")
script_edit.set("src", conf.APPURL[0] + "/server/wsgi/pageEditor.js")
script_edit.set("type","text/javascript")
script_config = g.root.makeelement('script')
g.root.body.append(script_config)
script_config.set("src", conf.APPURL[0] + "/server/config.js")
script_config.set("type", "text/javascript")


script_jq_mini = g.root.makeelement('script')
g.root.body.append(script_jq_mini)
script_jq_mini.set("src", conf.JQUERYURL[0] + "/jquery-1.7.min.js")
script_jq_mini.set("type", "text/javascript")


style_cust = g.root.makeelement('link')
style_cust.set("rel","stylesheet")
style_cust.set("type", "text/css")
style_cust.set("href", conf.JQUERYCSS[0] + "/jquery-ui.css")
g.root.body.append(style_cust)
```

ANNEX 2 : *Python scripts that are used to launch the application and the javascript code*

```
def post():
    dicts = []
    d = {}
    d['about'] = request.form['about']
    d['author'] = request.form['author']
    d['tag'] = request.form['tag']
    d['xpath'] = request.form['xpath']
    d['type'] = request.form['type']
    d['title'] = request.form['title']
    d['data'] = request.form['data']

    if(d['type'] == 'comment'):
        alipius = "tag:{0},type:{1},author:{2}".format(d['tag'],d['type'],d['author'])
        string ='<p about="{0}" alipius="{1}" xpath="{2}">{3}</p>'.format(d['about'],alipius
        try:
            title = d['title']
            d.pop('title')
        except KeyError:
            title = "Comment"
        dicts.append(d)

        blogEntry= ''
        blogger_service = service.GDataService(conf.EMAIL[0], conf.PASSWD[0])
        blogger_service.source = 'Servelots-alipi-1.0'
        blogger_service.service = 'blogger'
        blogger_service.account_type = 'GOOGLE'
        blogger_service.server = 'www.blogger.com'
        blogger_service.ProgrammaticLogin()
        query = service.Query()
        query.feed = '/feeds/default/blogs'
        feed = blogger_service.Get(query.ToUri())
        blog_id = " "
        if title == '':
            title = "Comment"
        for entry in feed.entry:
            if conf.BLOGURL[0] == entry.GetHtmlLink().href:
                blog_id = entry.GetSelfLink().href.split("/")[-1]
                blogEntry = CreatePublicPost(blogger_service, blog_id, title=title, content=
        rstr = storeCom(str(blogEntry.GetHtmlLink().href))
        response = make_response()
        response.data = 'ok'
        response.headers['Access-Control-Allow-Origin'] = '*'
        return response
```

ANNEX 3 : *Python function that allows the posting of the comment into the Alipi blog*

```
def storeCom(url):
    connection = pymongo.Connection('localhost', 27017)
    db = connection['dev_alipi']
    collection = db['post']
    root = lxml.html.parse(url).getroot()
    elements = root.xpath('//@alipius/..')
    store_list = []
    com_id = random.random()
    pat = re.compile('<.*?>')
    for element in elements:
        temp = {}
        for i in element.attrib['alipius'].split(','):
            try:
                temp[i.split(':')[0]] = i.split(':')[1]
            except  IndexError:
                pass
        if temp['type'] == 'comment':
            temp['about'] = element.attrib['about']
            temp['xpath'] = element.attrib['xpath']
            data =''
            ret = pat.search(lxml.html.tostring(element))
            data = lxml.html.tostring(element).partition(ret.group())[2]
            data = data.rpartition('</p>')[0]
            temp['data'] = data
            temp['blog'] = url
            temp['bxpath'] = makePath(element)
            temp['com_id'] = com_id
            store_list.append(temp)
    for z in store_list:
        collection.insert(z)
    connection.disconnect()
    return 'ok'
```

ANNEX 4 : *Python function that allows the storing into the MongoDB database*