Alipi Project

Internship report

Salma El Mohib

Tutor : M. Julien Allali
Internship supervisor : T.B Dinesh

November 8, 2012

Contents

addcontentslinetocsectionIntroduction

As a student working toward a degree in engineering, I had to do at the last of my second year an internship of three to four months, that take place in the period from june to september.
Since most of the time at university the theory part of the course studied is learnt, the internship allows to learn the practical part, moreover it is an opportunity to acquire more skills and experience.
This internship has for objective to introduce the student to a formal professional framework, to give him favorable circumstances to apply his knowledge in a team of professionals, by working on a project relying on some specifications imposed by his supervisor. It is a preview of his future job. As far as I was concerned, I fixed to myself one objective, which was accomplishing it in an english speaking country, since it is an opportunity to improve my english on the one side, and to discover a new way of working on the other side.
So in order to find my second year internship I attended to the forums organized by the school, which were all about compagnies in France, and I knew then there I had to proceed differently to achieve my goal.
I first applied to some colleges in foreign countries which all were laboratories, then considering the importance of working in a compagny, I dared to aim even higher, fully confident in myself, I wanted to work in a compagny in a foreign country. So I started contacting some teachers to get some help, until Mr Sebastian Fourestier accepted to help me find this internship in Servelots, implanted in Bangalore, India.

0.1 The Company

Servelots is a company founded and directed by Mr TB Dinesh, it provides free and open sources solutions for non-profit and non-governmental organisations since 2002. They help small non-profit organisations working on designing web-sites, configuring news-filters, helping them migrate to open source solutions, localisation and Indian language issues support, geographic information collection, and comprehensive or modular open source software development.

Between the projects that are being developed here, I chose alipi, a project they started working on last year. I made that choice for both the web semantics they are using I am interested into learning it, and for its social dimension.

Indeed, in India several languages are spoken (23 languages), only two are official : hindi and english, the rest of them are are local languages spoken by local people depending on the geographic area they live in. Actually, like in all developing countries, a lot of these people are not sufficiently literate to understand the official documents that are mainly in english or hindi, in addition to the visually handicapped. These two facts leads to web-exclusion of these people and keeping them from reaching the available aknowledgement on the web.

So they came up with the concept of alipi, which has the aim of including these people by helping them access to web pages they can be interested in.

The main idea of alipi is to "re-narrate" the pages, by giving to the users the opportunity of translating any web page targetting a certain community, in words they can understand, in addition to add audio-renarrations for the visually handicapped, or the illiterate people.

When I first came here, I was surprised to find out about the company which is composed of a team of 7 people, in addition to my internship supervisor TB Dinesh, he supervises the work of everyone, and helps with his suggestions. But I soon learnt that many people work for this company in different cities in India, and moreover out of India such as Turkey, France. During the period I have stayed in India, we went to an it university in Hyderabad to meet other people working on the same projects and work all together.

In Servelots many projects are being developed, however they maily work on the open-source api "alipi", and "Hampi" another api allowing one to get to explore in great detail the murals, carvings, architecture and videos of cultural heritage.

They are mainly using javascript and python-libraries for the development.

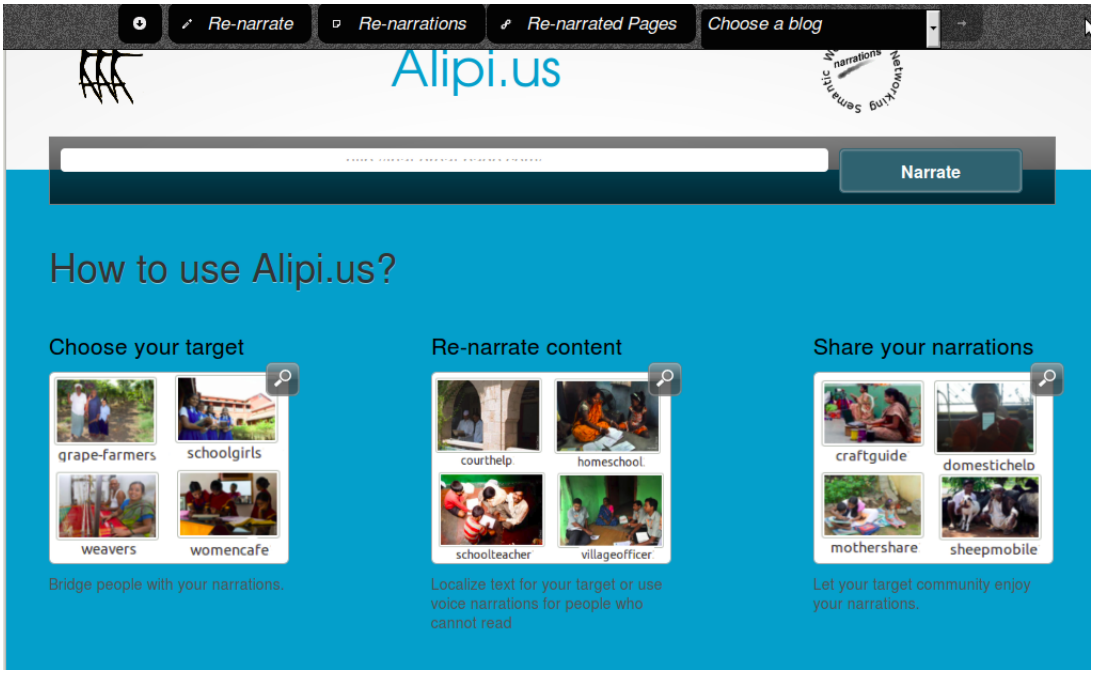
We are three interns coming the three of us from Enseirb-Matmeca. Arvind is the one charged of introducing the project to us, he had to answer to all the questions I wondered about.

First, I downloaded the source code, and I had to configure some files (as apache) to make the application work on localhost. We assisted to our first meeting with the other engineers about Selenium, a software testing framework for web applications, that some of us should work on later, applying it on the alipi api. But soon I learned that my task wouldn't be based on the existing code, but I had to start something new from the beginning, and it is the web-semantic. I needed some time to read the documentation about the web-semantic, the ontologies and the RDF, and learn how it works.

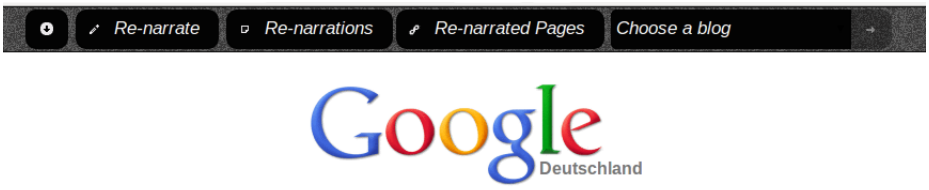
0.2 Alipi

0.2.1 Alipi project

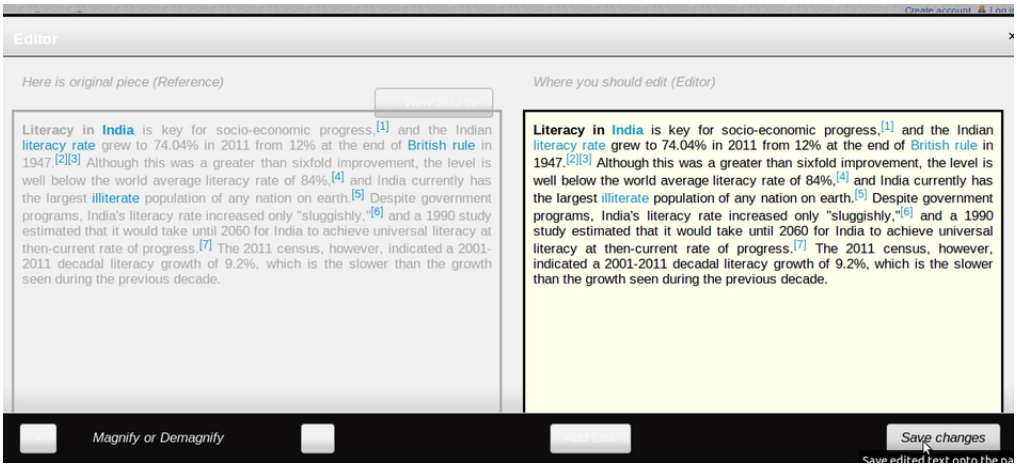
Alipi works by writing the URL of the site that is aimed to re-narrate on the website alipi.us, Then some buttons are proposed, so as to see the re-narrations already done for the web-site, to re-narrate, or to choose the blog where the re-narration will be posted, as seen above :



for example if the URL is www.google.com :



By choosing the option "renarrate" every field of text is available to re-narrate by clicking on it, a window then comes out with fields to fill, the html and javascript is already done, it only changes the texts to the languages the user wants.



0.2.2 Alipi using the MongoDB

The application Alipi is using a relational database, MongoDB.
Actually, once the text fields while renarrating are saved, there are some informations that the user needs to fill as his name, the language the renarration is written, the location of the people he is targetting, etc. Once these informations,that are defining the renarration, are saved, they are posted as feeds in a specific URL.
A web feed is a data format used for providing users with frequently updated content. The attributes in the feeds are later stored in MongoDB database by a python script.

Specifically, the process of storing the data in MongoDb is as the following explains.
When a user starts renarrating a web page, a function is responsible of posting the specific information about this renarration, in a **json** format, which are availabe on the url : http://y.a11y.in/web/feed.
The choice of json format for the feeds is due to its quality of being **human readable**, the data structure are called objects. Even if it is derived for the Javascript language, it is an independant language with parsers available for many languages.

Then from the json feeds the attributes of the renarration are posted as tweets in the url : http://y.a11y.in/web/feeds as the following :

@Arvind has narrated http://alipi.us//*[@id='content-middle']/H1

at http://alipi123.blogspot.com/2012/06/alipius$_7$919.*html*// ∗

@id='post-body-5826464676505885926'
/*p*

forindiainenglish

The function named "doScrape(url)" in the code passes the url to be crawled. Alipi attributes will be indexed from the given url. It is than responsible of extracting the information from the json feed and storing it in the MongoDB database.

Specifically, since the information about the web re-narration are generated in a feed format, then stored in the MongoDB database, which is a relational database, my task is to figure out a way to store this data instead in a database written in a specific format, which is a representation of Resource Description Framework(RDF) models.

0.3 Ontologies and web semantic

0.3.1 Ontologies

My role then, was to figure out a way to move to the semantic web,using rdf and ontologies, instead of the relational database.

Tim berners Lee the founder of the World Wide Web, at the first World Wide Web Conference in Geneva, in May 1994 : "To a computer, the Web is a flat, boring world, devoid of meaning. This is a pity, as in fact documents on the Web describe real objects and imaginary concepts, and give particular relationships between them. For example, a document might describe a person. The title document to a house describes a house and also the ownership relation with a person. Adding semantics to the Web involves

two things: **allowing documents which have information in machine-readable forms**, and **allowing links to be created with relationship values**. Only when we have this extra level of semantics will we be able to use computer power to help us exploit the information to a greater extent than our own reading.

A formal definition of the semantic web would be : it is an effort to enhance current web so that computers can process the information presented on WWW, interpret and connect it, to help humans to find required knowledge.
So as to have computers or artificial intelligences to connect and communicate between them without the interfering of human being, they should have the right description of domains.

I will start introducing the ontologies by stating an example. We suppose that we have two artificial intelligences, represented by agents A and B. If A wants to communicate with B, each one needs to know the other, and should dispose of the same references in a specific domain.

Ontology is what describes this domain, and should be shared between the two items in order to understand while communicating.

Ontologies enable to capture, process, reuse, and communicate the knowledge. It is an inner body of knowledge.
Unlike relational database it focuses more on the meaning and the links that bind the different components in a domain than the data.

Main differences between ontologies and relational database schema :

Database schema	Ontology
Focuses on data	Focuses on the meaning , the shared understanding, and the semantics.
The main purpose is to structure instances for the storage and the querying => the meaning is lost, because there is a minimum focus on the semantics	The same purpose of the Database schema, plus Human communication, interoperability, allows the semantic search. => Saves the meaning, strong focus on semantics.
The diagrams are in the form of Entity-Relationship. The schema is expressed by entities - attributes, relations – constraints.	The diagram just follows logic. We have Classes, Properties ans Axioms
Standard rules	No standard rules.
The instances are central.	The instances are optional.
Concerning SQL engines : the primary focus is querying, data integrity.	The primary focus is : derive new information from existing data, consistency integrity.
Rarely reused to build new ones, and stanrdized on SQL.	Very often reused, and standardization.

Some examples of ontologies very often used :

- foaf : or friend of a friend is used to link people, based on the web, it can be used in the social networks as facebook or twitter. As soon as a person publishes in a foaf document, the machine will be able to use this information, and if it conatins some links like "see also", leading to other documents, the machine , this way it is going to be a web of documents, that machines can handle, without the interfering of computer programmers, or just simple users.
- DC : DublinCore is a metadata used to describe documents

The Web Ontology Language OWL, is a family of languages that reprent knowledge, they are characterised by formal semantics and RDF/XML-based serializations for the Semantic Web.

0.3.2 RDF

The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web. RDF is a set of rules for encoding documents in a triple format: subject-object- predicate, expressions called triples, it is used as a common referential, it can be considered as a dictionnary which is a referential for people speaking the same language, except for the RDF, instead of the dictionnary, we get URI which can define in the RDF vocabulary other RDF files as ontologies or sometimes just web pages. RDF is designed to standardize the definition and use of metadata- descriptions.

One step back, before moving forward, as we all know, database is used to store data, and to retrieve the information via a query language. One question can cross our mind, why change from relational to triple store database, using the RDF.Actually, relational database and triple store work similarily with the same principle, but unlike the relational database, in the triple store the information is stored as triples, a fact that optimizes the storage, and makes the retrieval easier.

Moreover, the amount of data that is availabe in the web nowadays is huge, we can find any information we need in any topic, if we type the right query. The point is the researches on search motors as google, work grammatically, meaning that if we type for instance some key words, the fetch is based upon the grammar, not the sense or the link between the words.
So in order to organize the web and make the data more accessible, the World Wide Web Consortium (W3C) came with the idea of converting all the non well structured documents available on the web to one uniform format taking to account all the data related to one domain, transforming the web to a "web of data" as they call it, and enabling its enrichment.
According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.

As a start, I established an RDF file about re-narration, using the **ontology** DublinCore and it was validated, (you will find it in the picture below).

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://alipi.us#//*[@id='content-middle']/H1	http://purl.org/dc/elements/1.1/title	"Re-narration"
2	http://alipi.us#//*[@id='content-middle']/H1	http://purl.org/dc/elements/1.1/author	"Arvind"
3	http://alipi.us#//*[@id='content-middle']/H1	http://purl.org/dc/elements/1.1/language	"in"
4	http://alipi.us#//*[@id='content-middle']/H1	http://purl.org/dc/elements/1.1/location	"for"
5	http://alipi.us#//*[@id='content-middle']/H1	http://purl.org/dc/elements/1.1/identifier	"xpath page renarrated"

The original RDF/XML document

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:   xmlns:dc="http://purl.org/dc/elements/1.1/">
4:   <rdf:Description rdf:about="http://alipi.us#//*[@id='content-middle']/H1">
5:     <dc:title>Re-narration</dc:title>
6:     <dc:author>Arvind</dc:author>
7:     <dc:language>in</dc:language>
8:     <dc:location>for</dc:location>
9:     <dc:identifier>xpath page renarrated</dc:identifier>
10:
11:   </rdf:Description>
12: </rdf:RDF>
13:
```

Then I decided to create my own schema about the renarration and instead of using the dublin core metadata (as for dc:location), I would define my own **ontology** about renarration.

Rdf is a metadata (data about data) to describe information resources. So in order to describe information about renarration, I made up an rdf schema about renarration, containing the properties of each one (lang, title, author, location, blog, xpath, about, id). An example of "renarration" in the rdf schema is like the following :

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

<rdf:Description rdf:about="http://www.w3.org/2000/01/rdf-schema#Renarration">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>Renarration</rdfs:label>
  <rdfs:comment>The class renarration.</rdfs:comment>
</rdf:Description>
```

Properties definition

- title a name given to the resource.
- lang The language of the resource.
- locaion The location of the resource.
- author An entity primarily responsible for making the resource.
- bxpath The xpath of the blog where the resource is posted.
- blog The blog where the resource is posted.
- about The URL of the page aimed to renarrate.
- id The id of the author renarrating the page.

Once I knew how to use rdf, and create an rdf schema for renarration. I had to incorporate it to the project, by using the data from the json feeds for every and each renarration made, relying on the rdf schema established for the renarration.

0.3.3 RDFLIB

RDFlib is a Python library for working with RDF, and a powerful package, it contains different plugins, and contains :

- RDF/XML parser/serializer
- Persistent Graph backends, its nodes can be these instances : URIRef, Literal, BNode, Variable, QuotedGraph.
- Namespaces by giving the Base url of an ontology, and using just the attribute then
- Manages the triple files, adding, modifyng, removing
- Rdfstore where the rdf files are stored to be queried
- Sparql querying

In order to figure out how it works, and how to store the data in the database which is a triple store, I started by installing it. Once the file parsed and stored in the rdf triples, as rdf files stored in an rdf store, I had to generate the queries based on those rdf files. For that, I proceeded to get familiar with SPARQL that is a RDF query language to generate the queries on those triples.

0.3.4 Parsing Json Feeds

The first part of the project is situated right here : moving from the MongoDB to RDF based on the Json feeds.
So I had to figure out a way to parse the feeds generated, every renarration of a web page, to store them then in rdf files.
Each feed contains specific information about a renarration, as :

- The language of the renarrated page
- The xpath
- The author
- The location
- The id of the renarration
- The blog
- The blog xpath

The Json feeds in the url : http://y.a11y.in/web/feed are in this format :

```
{
  "0": {
    "lang": "English",
    "xpath": "//*[@id='content_page']/DIV/P",
    "about": "http://www.freelancefirm.nl/",
    "elementtype": "text",
    "author": "Eddy",
    "style": "Fun",
```

```
    "blog": "http://alipi123.blogspot.com/
2012/07/freelancefirm-een-
netwerk-van.html",
    "location": "Dutch",
    "ren_id": 0.11343031344870236,
    "_id": "5001c1f1eff0bd3d70000000",
    "data": "De juiste kennis en kunde op de juiste plek.
Dat maakt het verschil.
              Zeker in de online wereld, waar specifieke
kennis essentieel is en
die heeft u niet altijd
               ddfghh we verstand van. Meer dan 400 zorgvuldig
geselecteerde en
gemotiveerde online professionals
        staan voor u klaar. Snel en eenvoudig direct in
uw organisatie ingezet.
 Via freelancefirm beschikt u over:D",
    "bxpath": "//*[@id='post-body-9221501882154576185']/p"
  },
```


The python script to parse the json feed, opens the url in which the feed is contained, loads the json_data (the objects) into a variable "data", than data contains the object parsed and can be reused later to be stored in the rdf triples :

```
# Parse the JSON feed from the url
json_data=urllib2.urlopen('http://y.a11y.in/web/feed')
data = json.load(json_data)
json_data.close()
```

0.3.5 RDF files and rdfstore

For generating the rdf files, With the persistent Graphs that Rdflib contains, the triples can be created by calling the function graph in a python script after importing rdflib, and the graph from rdf lib :

```
from rdflib.graph import Graph
from rdflib import URIRef, Literal, BNode, Namespace
from rdflib import RDF

store = Graph()
```

The graph then can store the triples : subject-object-predicate, by calling the function "store.add(, ,)" it takes these three arguments.

```
store.add((donna, RDF.type, ren["Renarration"]))
```

The following example demonstrates the use of some of the Renarration schema properties in an RDF document:
The rdf file of renarration is generated as following :

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ren="http://localhost/ren/"
>
  <rdf:Description rdf:nodeID="KRlVDNuo3614">
    <ren:lang rdf:resource="Hindi"/>
    <ren:about rdf:resource=
"http://team.servelots.com/my/pradeep/Sanchaya/success.html"/>
    <ren:aboutXpath>/HTML/BODY/DIV[2]/UL[2]/LI[4]</ren:aboutXpath>
    <ren:author rdf:resource="http://alipi.us/users/
 4ff10913eff0bd0989004458/rdf"/>
    <ren:blogXpath>//*[@id='post-body-60']/p[4]</ren:blogXpath>
    <rdf:type rdf:resource="http://localhost/ren/Renarration"/>
    <ren:location rdf:resource="Karnataka"/>
  </rdf:Description>
```

As we can notice all the attributes are from "ren", ren is the url where the rdf schema defined before is located. So instead of having "foaf" or "dc" the renarration rdf schema is used.

Semantic Description of a Person :
A person is someone doing a renarration. A person interested in renarrating has some information when using Alipi:

- name
- socialid
- blog address (if any)

Alipi is using lastuser service, thus each person has a:
unique lastuser id
Using all this information, we create a FOAF file for each renarrator. This FOAF file will be stored in Alipi RDF Store and can be reached via http://localhost/users/lastuserid/rdf. Python code is as the following:

```
from rdflib.graph import Graph
from rdflib import Literal, BNode, Namespace
from rdflib import RDF


def generateFOAF(person_unique_id, person_name,
 person_social_id, alipi_ns, person_weblog=None):
g = Graph()

# Bind a few prefix, namespace pairs.
g.bind("dc", "http://http://purl.org/dc/elements/1.1/")
g.bind("foaf", "http://xmlns.com/foaf/0.1/")
g.bind("owl", "http://www.w3.org/2002/07/owl#")
g.bind("ren", "http://localhost/ren")

# Create FOAF namespace
FOAF = Namespace("http://xmlns.com/foaf/0.1/")
OWL = Namespace("http://www.w3.org/2002/07/owl#")
RDF = Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
```

```
ALIPI = Namespace(ren)

# Identifier for person
person = BNode()

# Add triples
g.add((person, RDF.type, FOAF["Person"]))
g.add((person, ALIPI["lastUserId"], Literal(person_unique_id)))
g.add((person, FOAF["name"], Literal(person_name)))
g.add((person, OWL["sameAs"], person_social_id))
if person_weblog is not None:
g.add((person, FOAF["weblog"], Literal(person_weblog)))

print g.serialize()

return g
```

An example query is:

```
generateFOAF("1234","Salma El Mohib",
"http://www.facebook.com/agacho", "http://alipi.us/",
 "http://agacho.blogspot.com")
```

where 1234 is lastuserid, Salma El Mohib is name, http://www.facebook.com/agacho is social id used for login process, http://alipi.us/ is alipi namespace, and http://http://agacho.blogspot.com is blog address where renarration wil be published.

Result of the query is as the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:alipi="http://alipi.us/"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
```

```
  <rdf:Description rdf:nodeID="_8d413f4b-7a1e-4340-87d2-7aa6c0568ca0">
    <owl:sameAs rdf:resource="http://www.facebook.com/agacho"/>
    <alipi:lastUserId>1234</alipi:lastUserId>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <foaf:name>Nadin Kokciyan</foaf:name>
    <foaf:weblog>http://agacho.blogspot.com</foaf:weblog>
  </rdf:Description>
</rdf:RDF>
```

It is a valid FOAF file where four namespaces are used: alipi, foaf, owl, and rdf.
How to use this method?
After each renarration, this rdf file will be generated using generateFOAF method. Resulted RDF file will be pushed into the Alipi RDF Store.
Each renarration is represented in a RDF file. In this file, the renarrator should be described via FOAF of this person using "rdfs:seeAlso" property where rdfs is rdf schema namespace.

Example usage:
In RDF file of a renarration, a renarrator will be described as:

```
<alipi:renarrator>
<rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
<rdfs:seeAlso rdf:resource="http://alipi.us/users/1234/rdf">

</alipi:renarrator>
```

0.3.6 SPARQL queries

Sparql is the language used to query the rdf files stored in the rdf store. In order to perform SPARQL queries, the installation of the companion rdfextras package is nedded, it includes a SPARQL plugin implementation:

In order to use the SPARQL plugin in the code, the plugin had first to be registered. This binds the the imported SPARQL query processor implementation to the rdflib.graph.Graph.query() method, which can then be passed a SPARQL query (a string). When called, the query() method returns a SPARQLQuery object whose result attribute is a list of results.

```
plugin.register(
'sparql', rdflib.query.Processor,
'rdfextras.sparql.processor', 'Processor')
plugin.register(
'sparql', rdflib.query.Result,
'rdfextras.sparql.query', 'SPARQLQueryResult')
```

Then to start querying, we need first to put the PREFIX : ren for the rdf schema stablished for rennaration, rdf, rdfs for rdf schema, owl, for ontologies, xsd for xml schema.

```
PREFIX ren: ¡http://localhost/ren# ¿
PREFIX rdf :< http://www.w3.org/1999/02/22 − rdf − syntax −
ns# >
PREFIX rdfs :< http://www.w3.org/2000/01/rdf − schema# >
PREFIX owl :< http://www.w3.org/2002/07/owl# >
PREFIX xsd :< http://www.w3.org/2001/XMLSchema# >
```

For the query : who is the author of the renarration with this id = 1 ?

```
SELECT ?x ?author
WHERE
?x ren:id "1" .
?x ren:author ?author .
```

The result is

```
x author
ns:1 "Arvind"
```

0.4 Recommendations

Today, we are lucky to live in an area that allows us to have information at our fingertips with all the medias, the internet, and the web services that

provide much more information that we can imagine, from huge databases. Unfortunately some of us do not have the enough knowledge or literacy to access it, that is the main reason I was more thant proud and happy to work on the alipi project. Because taking aside, the web semantics, and the computer science skills I acquired during these three months in India, I could help people access the web pages in languages they can understand, and that is something wonderful to see that I won't be the only one taking advantage of this internship.

Spending three months in Bangalore, known also as the Indian silicon valley, working on this project, in Servelots company, taught me a lot. It was weird for me at the begining, to work on a project without a request for proposal document with specific deadlines, but that was one thing that encouraged me to take control of my project, and fix my expectations from it on my own, it gave me more confidence by making my own decisions for what I judged was good for my work.

Moreover, the relationship with the advisor encouraged the motivation, that special link that I could have with the engineers that have much more experience, and still look at me as their equal, and ask for my advice, helped me improve myself in different ways professionally speaking.

They treat each other in a way they let you all be equal, like if we all don't know the answers to some new issue, like no one knows everything, so we all have to look for answers, helping each others, and proposing our point of views.
I was supposed to be just a regular intern, but they gave me this extra respect that gave me more confidence. And I can assure that this collaborate process is much more fruitfull, students become more confidents, they acquire sense of honorship, listen to their peers, and try the best they can. My supervisor was more like a guide, and I think that was something that was really important that helped me achieve the internship in the best way possible.

I would like to thank mr Sebastian Fourestier for giving me this opportunity to have my chance here, and mr TB Dinesh for allowing me work on his project that I value.