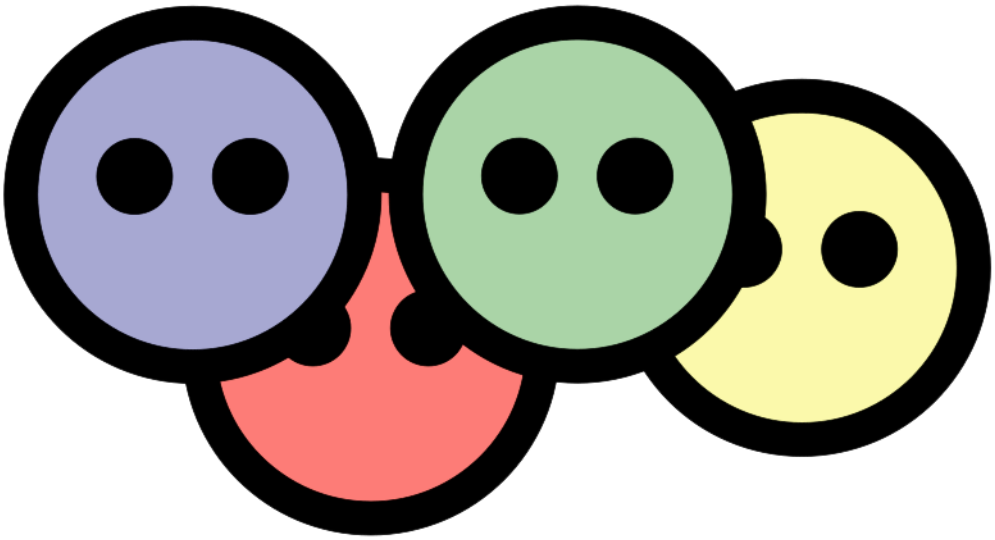
Projet Technologies du Web

FoaFinder & FoaFox
ou
Développement d'un WebService et d'un AddOn Firefox permettant de naviguer à travers des profils FOAF.



Sébastien Pontaillier
sebastien.pontaillier@gmail.com

Université d'Evry
MIAGE M2 Apprentissage 2009:2010

Frédéric Eterno
frederic.eterno@gmail.com

Index

Présentation

Introduction

Dans le cadre de l'enseignement Technologies du Web / SOA dispensé à l'université d'Evry, il nous a été proposé un projet destiné à nous permettre de mettre en application les connaissances acquises durant ce cours. Parmi le vaste choix de sujets proposés, composé de sujets définis par les deux professeurs responsables du projet, Henry Boccon-Gibod et Vincent Godefroy, notre intérêt s'est porté sur le développement d'une extension pour le navigateur Firefox permettant d'exploiter les fichiers au format FoaF. Voici l'intitulé du sujet tel qu'il nous a été proposé : **"En utilisant le standard FoaF, on imagine l'utilité d'un plugin pour FireFox, permettant d'exploiter les ressources disponibles que permet ce standard. On imagine aussi l'enrichissement de bases de connaissances de réseaux sociaux exprimés en FOAF, à partir de requêtes sur des systèmes de communication en vogue tels que Twitter ou Wikipédia"**.

Le projet a été découpé en plusieurs phases. Nous avons tout d'abord mis au point un cahier des charges, qui a été remis lors de la troisième séance de projet et validé par nos responsables. Le développement a ensuite pu démarrer et a constitué la partie la plus importante de notre projet en terme d'investissement et de charge de travail. La dernière étape du projet consiste en la rédaction du présent rapport.

Ce document, après avoir effectué un rappel du contexte ainsi qu'une présentation du format FoaF, présente les choix architecturaux qui ont été faits lors du développement, avant de décrire de manière précise les deux parties de l'application : le service, FoaFinder, permettant de réaliser diverses requêtes sur des fichiers FoaF, et le client, FoaFox, extension Firefox permettant d'interroger ce service et d'utiliser les résultats pour présenter des informations cohérentes et facilement exploitables à l'utilisateur.

Terminologie employée

Utilisateur : personne physique utilisant le plugin FireFox développé dans le cadre du projet

Administrateur : personne physique ayant à charge de gérer le service web développé

Ressource : fichier FoaF répertorié par le service et utilisé par le plugin

Dépôt : espace dans lequel sont regroupées des ressources FoaF pouvant prendre plusieurs formes (URL, FTP, base de données, espace disque...)

FoaFox : nom du plug-in FireFox développé

FoaFinder : nom du service web développé

Contexte

FoaF - Friend of a Friend - (http://www.foaf-project.org/) est une spécification XML basée sur le format RDF dont l'idée a été proposée par Dan Brickley and Libby Miller en 2000. Il permet de voir les personnes, les pages web et les informations disponibles sur le web comme un gigantesque réseau dont chaque nœud est lié aux autres, éventuellement par l'intermédiaire d'autres nœuds. En utilisant un tel format il est facile de naviguer d'une personne à l'autre en considérant leurs amis, leurs centres d'intérêts, ou toute autre information qu'ils désirent fournir.

Bien que cette technologie n'en soit qu'à un stade expérimental, plusieurs sites proposent déja la publication de profils FOAF. On compte parmi eux Advogato (http://www.advogato.org/), LiveJournal (http://www.livejournal.com/) ou encore CrazyLife (http://www.crazylife.org/). La plupart de ces sites donnent la possibilité aux utilisateurs de créer leur profil en ligne, qui donne par la suite lieu à la génération d'un profil FoaF basé sur les information saisies.

La structure des fichiers FoaF est fixée par une spécification officielle (http://xmlns.com/foaf/spec/), mais le caractère expérimental de cette technologie conduit chaque site à utiliser une syntaxe qui lui est plus ou moins propre, bien que la structure générale des fichiers soit similaire la même d'un site à l'autre. Voici un extrait de fichier FoaF rédigé selon les spécifications officielles :

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:admin="http://webns.net/mvcb/">
  <foaf:Person rdf:ID="me">
    <foaf:name>Sébastien Pontaillier</foaf:name>
    <foaf:title>Mr</foaf:title>
    <foaf:givenname>Sébastien</foaf:givenname>
    <foaf:family_name>Pontaillier</foaf:family_name>
    <foaf:nick>I-Cue</foaf:nick>
    <foaf:mbox_sha1sum>310e59568aa1ad673ee905b73c9a6e9536e2cabe</foaf:mbox_sha1sum>
    <foaf:homepage rdf:resource="http://blog.s-pontaillier.fr"/>
    <foaf:workplaceHomepage rdf:resource="http://www.altissemiconductor.com"/>
    <foaf:knows>
      <foaf:Person>
        <foaf:name>Frédéric Eterno</foaf:name>
        <foaf:mbox_sha1sum>c2d40c75e5981b456c84dc7f531e5fce82a45714</foaf:mbox_sha1sum>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
</rdf:RDF>
```

Il existe donc de nombreux fichiers FoaF prêts à être exploités sur la toile. C'est forts de cette constatation que nous avons commencé à établir les spécifications de notre application, qui sont présentées ci-dessous.

Status

Comme il a été dit précédemment, notre projet se découpe en deux parties distinctes : le service FoaFinder, qui fournit la possibilité d'exécuter des requêtes sur la base de données contenant les fichiers FoaF répertoriés, et le client, l'extension FoaFox, qui permet d'exploiter le service en lui passant des requêtes et en affichant les résultats à l'utilisateur de manière lisible et fonctionnelle. Il a été prévu dans le cahier des charges que les fonctionnalités suivantes soient disponibles dans la version finale de FoaFox (à chaque fonctionnalité correspond un nom indiqué entre parenthèses qui sera utilisé pour désigner la fonctionnalité par la suite) :

- recherche par mot-clé ("rechercher") : une recherche classique par saisie d'un mot clé, qui donne lieu à l'interrogation de la base de données du service puis à l'affichage des résultats dans la fenêtre FoaFox.
- lister les dépôts FoaF enregistrés dans la base de données dans la fenêtre FoaFox ("listerDepots"). Chaque dépôt est affiché sous la forme d'un lien vers le site correspondant.
- trouver des liens entre deux personnes à partir de leur profil FoaF ("trouverLiens") : à partir de la soumission de deux profils FoaF, afficher les liens qui apparaissent entre les deux personnes, si il en existe.
- soumettre un profil FoaF trouvé sur le web ("soumettreProfil") : possibilité pour l'utilisateur de soumettre l'URL d'un fichier FoaF pour qu'il soit ajouté dans la base de données du service.
- afficher une cartographie des ressources enregistrées dans la base de données ("carto") : il s'agit d'un graphe illustrant les dépendances entre les fichiers FoaF répertoriés par le service.
- mettre en évidence les liens vers les ressources FoaF trouvées sur une page web pendant la navigation ("signalerRessources") : lors de l'affichage d'une page web, activer à l'aide d'un bouton la recherche de liens vers des ressources FoaF, et les mettre en évidence s'il en existe. Cette fonctionnalité a été définie comme non-prioritaire.
- configurer FoaFox ("configurer") : spécifier les paramètres de la recherche (nombre de résultats affichés, etc...).

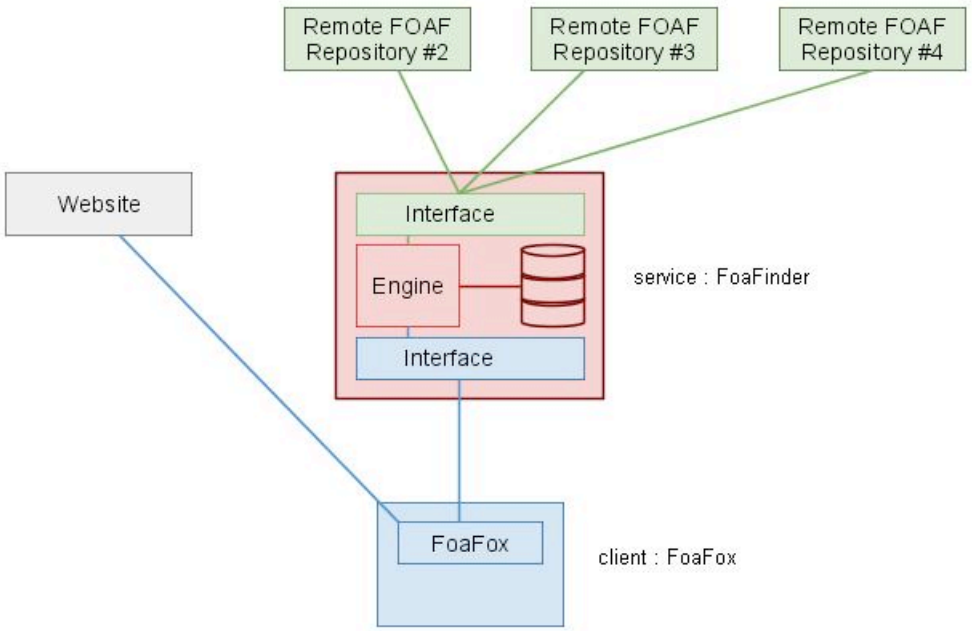
Le sujet qui nous a été proposé mentionne la possibilité d'enrichir la base de connaissance du WebService à partir de systèmes de communication tels que Twitter ou Wikipedia. Comme défini dans le périmètre du projet du cahier des charges initial, le projet ne fournit pas de solution permettant de combler ce besoin. Néanmoins, il nous apparait comme envisageable d'interfacer le WebService avec Twitter : un compte nommé FoaFinder pourrait par exemple recevoir les Twitts émis par d'autres utilisateurs. On imaginera un Twitt de la forme "@Foafinder ADD_RESOURCE http://example.com/user/foaf.rdf" qui déclenchera l'ajout en base de données du profil FOAF pointé par l'URI. Le WebService aura donc à charge de déployer un agent permanent qui 'attrapera' les Twitts.

Le tableau suivant fournit un état des lieux pour chaque fonctionnalité : rappel du nom, proportion du travail accompli par rapport aux prévisions du cahier des charges, observations et commentaires.

Fonctionnalité	**Avancement**	**Commentaires / Observations**
Rechercher	80%	Recherche par mot-clé entièrement opérationnelle, avec affichage des résultats et affichage possible d'une page de détails. Recherche avancée abandonnée, gain faible en terme de fonctionnalités par rapport au temps nécessaire pour son développement
ListerDepots	100%	Fonctionnalité entièrement opérationnelle, liste des dépôts accessibles d'un simple clique sur l'item correspondant dans le menu de l'extension. Affichage de statistiques concernant la proportion de la contribution de chaque dépôt au remplissage de la base de données
TrouverLiens	100%	Fonctionnalité entièrement opérationnelle : recherche de deux profils, sélection parmi les résultats de recherche et affichage des liens trouvés
SoumettreProfil	95%	Fonctionnalité entièrement opérationnelle côté service. Côté client, soumission possible par saisie d'une URL dans l'extension. Objectif pour la démonstration : afficher une icône dans la barre d'adresse de Firefox permettant de soumettre le fichier FoaF affiché (cette icône ne doit apparaître que si l'adresse saisie pointe sur un fichier foaf)
Cartographie	10%	Fonctionnalité non encore disponible côté serveur. Côté client, lien prévu dans la liste de résultats de la recherche. Objectif pour la démonstration : lors du clique sur le lien, envoyer une requête au serveur qui répond du code SVG permettant le tracé d'un graphe de dépendance du fichier FoaF sélectionné
SignalerRessources	0%	Fonctionnalité définie au départ comme non-prioritaire. Temps insuffisant pour permettre son implémentation
Configurer	100%	Fonctionnalité client uniquement. Permet de modifier les paramètres de la recherche, en particulier le nombre de résultats à afficher

Architecture

Ce schéma présente l'architecture globale de l'application FoaFinder, ainsi que son interaction avec le client FoaFox. Notez que le service étant ouvert, tout client ayant correctement implémenté les méthodes de l'API suivra le même schéma.



FoaFinder est donc capable de communiquer avec différents dépots de fichiers FOAF (boites vertes). Le moteur du WebService peut ainsi copier ces profils dans la base de données locale (boites rouges) afin d'optimiser et d'uniformiser les traitements et opérations du service. Enfin le WebService met ses services à disposition du public, l'add-on FoaFox pour Firefox 3.5+ dans ce document, via des composants d'interface client (boites bleues).

Le WebService : FoaFinder

Le serveur

FoaFinder est un WebService hébergé sur une machine Ubuntu-9.04, faisant tourner un serveur web Apache-2 compilé avec PHP-5.2. De plus, un moteur de base de données MySQL-5 est utilisé afin d'assurer la persistance des données.

Comme on le constatera rapidement, la totalité des technologies utilisées pour implémenter ce WebService est libre et OpenSource. Plus qu'annuler les coûts de mise en production, l'utilisation de technologies OpenSource offre (presque toujours) la garantie de bénéficier d'un support communautaire et d'une interopérabilité maximale.

De plus, la disponibilité d'un tel serveur web Ubuntu à notre domicile a conditionné positivement notre choix.

API

La description du WebService est effectuée dans le fichier WSDL disponible ici : http://x-home.hd.free.fr/admin/miage/foafinder/foafinder.xml

Search($keyword, $offset, $count)

La méthode Search retourne $count résultats à partir de l'$offset dont la chaine de caractère $keyword débute le nom ou le pseudonyme d'un profile FOAF.

INPUT:

- $keyword (string) : mot clé utilisé pour faire la recherche
- $offset (int) : point de départ des résultats sélectionnés
- $count (int) : nombre de résultats

```
<m:Search xmlns:m="urn:soa">
  <keyword xsi:type="xsd:string">$keyword</keyword>
  <offset xsi:type="xsd:integer">$offset</offset>
  <count xsi:type="xsd:integer">$count</count>
</m:Search>
```

OUTPUT:

- $nb (int) : nombre de résultats retournés (inférieur ou égal à $count)
- $id (int) : identifiant du résultat courant
- $name (string) : nom ou pseudonyme complet du profil FOAF
- $uri (string) : URI du profile FOAF

```
<resultSearch>
  <set nb=$nb>
    <foaf id=$id>
      <name>$name</name>
      <uri>$uri</uri>
    </foaf>
    ...
  </set>
</resultSearch>
```

ListRepositories()

La méthode ListRepositories retourne le listing des dépots FOAF utilisés par le service, ainsi qu'un lien vers un camembert statistique Google.

INPUT:

- *null*

```
<m:ListRepositories xmlns:m="urn:soa">
</m:ListRepositories>
```

OUTPUT:

- $count (int) : nombre de résultats retournés
- $img (string) : URI du camembert statistique Google correspondant aux resultats
- $id (int) : identifiant du résultat courant
- $uri (string) : URI du dépot courant

```
<resultListRepositories>
  <repositories count=$count img=$img>
    <repository id=$id>$uri</repository>
    ...
  </repositories>
</resultListRepositories>
```

AddRecord($uri)

Cette méthode permet de soumettre une ressource FOAF au WebService. Si la ressource est valide, le service essaiera de l'ajouter en base ou de la mettre à jour.

INPUT:

- $uri (string) : URI de la ressource FOAF à soumettre

```
<m:AddRecord xmlns:m="urn:soa">
  <uri xsi:type="xsd:string">$uri</uri>
</m:AddRecord>
```

OUTPUT:

- $status (int) : status de l'opération {3, 2, 1, 0}
  - 3 : profil FOAF ajouté en base
  - 2 : profil FOAF mis à jour en base
  - 1 : profil FOAF trouvé mais non mis à jour
  - 0 : profil FOAF non valide (URI non valide ou problème de structure XML)

```
<resultAddRecord>
  <status>$status</status>
</resultAddRecord>
```

FindLinks($uri1, $uri2)

Cette méthode renvoie les points communs (amis, projets, interets) des profils FOAF disponibles à $uri1 et $uri2.

INPUT:

- $uri1 (string) : URI de la ressource FOAF
- $uri2 (string) : URI de la ressource FOAF

```
<m:FindLinks xmlns:m="urn:soa">
  <foaf1 xsi:type="xsd:string">$uri1</foaf1>
  <foaf2 xsi:type="xsd:string">$uri2</foaf2>
</m:FindLinks>
```

OUTPUT:

- $count (int) : le nombre de similitudes trouvées dans la section
- $id (int) : l'identifiant de la similitude courante de la section courante

- $name (string) : nom de la similitude courante
- $uri (string) : URI de la similitude

```
<resultFindLinks>
  <status>
    <know count=$count>
      <friend id=$id>
        <name>$name</name>
        <uri>$uri</uri>
      </friend>
      ...
    </know>
    <projects count=$count>
      <project id=$id>
        <name>$name</name>
        <uri>$uri</uri>
      </project>
      ...
    </projects>
    <interests count=$count>
      <interest id=$id>
        <name>$name</name>
        <uri>$uri</uri>
      </interest>
      ...
    </interests>
  </status>
</resultFindLinks>
```

FullMap($uri, $deep)

La méthode FullMap permet d'obtenir un graphe de dépendance généré par le WebService pour la ressource FOAF disponible à $uri, pour une profondeur $deep.

INPUT:

- $uri (string) : URI de la ressource FOAF
- $deep (int) : Profondeur à parcourir pour la représentation en SVG

```
<m:FullMap xmlns:m="urn:soa">
  <uri xsi:type="xsd:string">$uri</keyword>
  <deep xsi:type="xsd:integer">$deep</offset>
</m:Search>
```

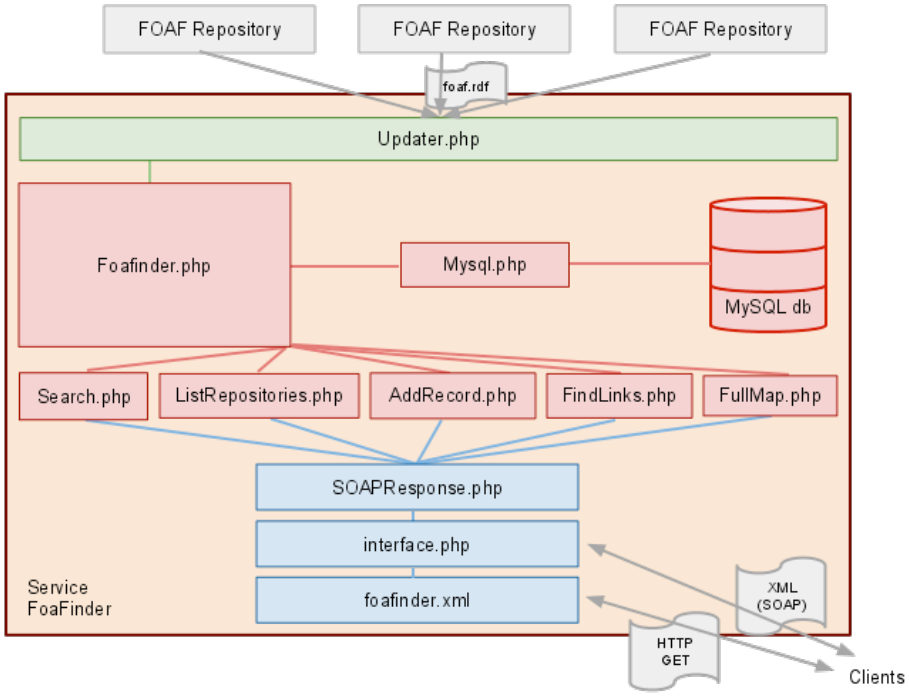
OUTPUT:

- not defined yet...

```
not defined yet...
```

Composants

Voici une vue éclatée des différents composants qui constituent le WebService, ainsi que les agents extérieurs avec qui le WebService est amené à communiquer.



Interface client

Le fichier de description du WebService est **foafinder.xml**. Il ne porte pas l'extension ***.wsdl** car il semblerait que notre Apache (le serveur web) ne fasse pas correspondre l'extension ***.wsdl** au MimeType XML. L'extension ***.xml** permet de forcer l'interprétation (et non le téléchargement) du fichier de description. Comme cela ne bloque et ne compromet pas l'intégrité du service, nous n'avons pas donné une priorité haute à résoudre ce point.

A l'aide de ce fichier descriptif, des clients peuvent découvrir quelles sont les méthodes disponibles, les arguments qu'elles nécessitent ainsi que leurs valeurs de retour. Le fichier **foafinder.xml** est aussi utilisé pour instancier le serveur SOAP, dans **interface.php**. Enfin, chaque requête adressée au serveur SOAP est transmise à la classe **SOAPResponse.php** qui agira comme un aiguilleur et transmettra la demande à une des classes du moteur (boites rouges).

Extrait du fichier de description du service, mettant en évidence la topologie des arguments à passer aux méthodes:

```
<!-- INPUT ######################################################## -->


<!-- Search -->
<message name="inputRequestSearch">
    <part name="keyword" type="xsd:string"/>
    <part name="offset" type="xsd:integer"/>
    <part name="count" type="xsd:integer"/>
</message>

<!-- ListRepositories -->
<message name="inputRequestListRepositories"></message>

<!-- AddRecord -->
<message name="inputRequestAddRecord">
    <part name="uri" type="xsd:string"/>
</message>

<!-- FindLinks -->
<message name="inputRequestFindLinks">
    <part name="foaf1" type="xsd:string"/>
    <part name="foaf2" type="xsd:string"/>
</message>

<!-- FullMap -->
<message name="inputRequestFullMap">
    <part name="uri" type="xsd:string"/>
    <part name="deep" type="xsd:int"/>
</message>
```

Moteur

Le moteur de FoaFinder est tout d'abord constitué des classes de requêtes **Search.php, ListRepositories.php, AddRecord.php, FindLinks.php et FullMap.php**. Ces classes permettent de récupérer les informations demandées par une requête, de les formatter et de les renvoyer.

Toutes ces classes embarquent des objets de la classe **FoaFinder.php** qui centralise la plupart des opérations et traitement du WebService. La classe **Mysql.php** fait partie du projet PHP MySQL Simple Library de Laurent Léonard (http://www.graphix-webdesign.com) et nous permet de complètement centraliser et déléguer les requetes au SGBD, MySQL en l'occurence.

Extrait de la classe **FoaFinder.php** : la méthode de recherche en base de données (Search) :

```
/**
* Search function (public) : Goes in database and looks at for the keyword
* @param string $keyword : Keyword to search in FOAF profiles
* @param integer $offset : Start return point
* @param integer $count : Number of results
* @return array() : array(int id, array(uri/foaf))
**/

public function search($keyword, $offset, $count) {

    $r = $this->db->Select(
              $table = 'profiles',
              $field = array('uri', 'foaf'),
              $where = "uri = '{$keyword}' OR nick LIKE '{$keyword}%' OR name LIKE '{$keyword}%'",
              $group = '',
              $order = '',
              $limit = "{$offset}, {$count}"
              );

    if (!empty($r))
      for ($i = 0; $i < count($r); $i++)
        $r[$i]['name'] = $this->extract('name', $r[$i]['foaf']);
    return (!empty($r)) ? $r : null;
}
```

Stockage des profiles FoaF

La persistance des profils en mémoire est prise en charge par une base de données MySQL, qui contient à ce jour plus de **180.000 profils FOAF** trouvés sur Internet. Cette base de données est de taille non négligeable (presque 5Go) et elle est régulièrement alimentée, nettoyée, mise à jour et sauvegardée. L'alimentation permanente se fait grâce à une tâche 'cron' qui tire un fichier FOAF de la base au hasard toutes les minutes, le scanne, et ajoute les liens FOAF valides qu'il contient dans la base. Le nettoyage consiste à supprimer les profils FOAF dont l'URI ne correspond plus à un document valide. Enfin, tous les matins à 7h30 un 'cron' sauvegarde la totalité de la base de données au format SQL, afin de prévenir toute défaillance.

Les lanceurs de taches cron :

```
# Alimentation permanante de la base de données
* * * * * root curl -sf http://x-home.hd.free.fr/admin/miage/foafinder/cron.php?action=update >> /var/log/foafinder/update.log

# Sauvegarde des bases du serveur
30 7 * * * root sh /var/scripts/save_sql.sh >> /var/log/mysql/backup.log
```

La sauvegarde des bases de données du serveur (/var/scripts/save_sql.sh) :

```
#!/bin/bash
#
## on se place dans le repertoire ou l'on veut sauvegarder les bases
#
cd /var/backups/u-serv/sql

for i in blog eventum foafinder glpi miage tekweb wiki; do

## Sauvegarde des bases de donnees en fichiers .sql
mysqldump -uroot -p********* $i > ${i}_`date +%D | sed 's;/;-;g'`.sql

## Compression des exports en tar.bz2
tar jcf ${i}_`date +%D | sed 's;/;-;g'`.sql.tar.bz2 ${i}_`date +%D | sed 's;/;-;g'`.sql

## Suppression des exports non compresses
rm ${i}_`date +%D | sed 's;/;-;g'`.sql

done
```

Indexation des profiles FoaF

C'est la classe **Updater.php** qui a pour role de rapattrier les profils FOAF trouvés çà et là sur le web. Le stockage consistait au début simplement en une copie dans la base MySQL. Mais des problèmes de performances se sont vite posés, lorsque nous avions à peine 40.000 profiles en base. En effet, les temps de réponse moyen à une requete Search dépassait généralement les 30 secondes.

Nous avons donc décidés d'ajouter les index 'nick' et 'name' à la table des profils : **Updater.php** remplit ces champs lorsqu'il essaye de sauvegarder de nouveaux profils, en parsant les fichiers FOAF. La base MySQL maintient un index sur ces champs afin d'optimiser les temps de réponses lors d'une recherche : de 30secondes nous sommes passés à moins d'une seconde lors d'une requête Search...

L'implémentation d'une fonction de recherche avancée nécessiterait certainement l'ajout d'autant de champs supplémentaires indéxés, si toutefois le SGBD restait relationnel...

Voici les informations de la base de données de FoaFinder, et notament un listing des index maintenus.

```
mysql> use foafinder;
Database changed
mysql> show tables;
```

Tables_in_foafinder
profiles
repositories

mysql> show columns from profiles;

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
uri	varchar(256)	NO	UNI	NULL	
foaf	text	NO		NULL	
nick	varchar(256)	NO	MUL	NULL	
name	varchar(256)	NO	MUL	NULL	
updated	timestamp	NO		CURRENT_TIMESTAMP	

mysql> show columns from repositories;

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
uri	varchar(256)	NO	PRI	NULL	
pattern	varchar(100)	NO		NULL	
count	int(11)	NO		NULL	

mysql> show index from profiles;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality
profiles	0	PRIMARY	1	id	A	182262
profiles	0	uri	1	uri	A	182262
profiles	1	nick	1	nick	A	182262
profiles	1	name	1	name	A	182262

mysql> show index from repositories;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality
repositories	0	PRIMARY	1	id	A	NUL
repositories	0	PRIMARY	2	uri	A	10

Le client : extension FoaFox pour Firefox

Le sujet de notre projet nous a imposé l'utilisation des technologies Mozilla pour le développement du client (extension Firefox). Dans le cadre du Master 2 Miage, nous avons pu bénéficier d'une semaine d'enseignement intensif dispensé par des intervenants membres de la communauté Mozilla, au cours de laquelle nous avons découvert le développement d'extensions pour le navigateur Firefox. Un projet dédié à également été lancé pour valider les connaissances acquises. Nous avons donc saisi cette occasion de parfaire nos compétences dans les technologies Mozilla.

Architecture

Le développement d'une extension pour Firefox impose la mise en place d'une arborescence précise, qui est décrite succinctement ci-dessous. Il s'agit de l'arborescence la plus simple pouvant être utilisée pour la mise en place d'une extension. Pour des raisons évidentes, nous ne pouvons décrire tous les aspects de la mise en place d'une extension. Pour de plus amples informations sur le sujet, il est possible de se reporter au Mozilla Developer Center (https://developer.mozilla.org/En)

chrome.manifest

Ce fichier définit les différentes URL chrome (de la forme /chrome/content par exemple) qui permettent à l'extension de trouver les fichiers dont elle a besoin pour fonctionner. Il définit également les fichiers d'overlay à utiliser.

install.rdf

Ce fichier fournit les informations de version et compatibilité de l'extension. Il regroupe entre autres l'id de l'extension, sa version, son auteur, ainsi que les logiciels avec lesquels elle est compatible (dans notre cas, une seule application : Firefox 3.0+).

chrome/content/

Ce répertoire contient tous les fichiers utilisés par l'extension. Il regroupe des fichiers aux formats XUL, JavaScript et CSS.

Solutions techniques

Comme il a été dit, le développement d'une extension implique d'utiliser des technologies précises. Voici une liste de ces technologies accompagnée d'un descriptif et des avantages qu'elles ont présentées pour la réalisation du client :

XUL

Dérivé du XML, ce langage permet de dessiner des interfaces à la manière de pages HTML. Il s'appuie donc sur un système de balises permettant la mise en forme et l'agencement des éléments, mise en forme pouvant être complétée à l'aide feuilles de style CSS. Il permet entre autre la réalisation de

formulaires, de tableaux, de listes, ou de tout autre élément d'interface fournit par des langages comme Java. Les spécifications du langage XUL sont disponibles à l'adresse suivante : https://developer.mozilla.org/en/XUL. Grâce à la formation que nous avons suivi avec Mozilla, nous avons pu économiser le temps de familiarisation et de prise en main du langage. Sa grande flexibilité nous a en outre permis de réaliser précisément nos interfaces utilisateurs tel que nous l'avions prévu.

JavaScript

La réalisation des traitements dans une extension Firefox est obligatoirement assurée par le langage JavaScript. Comme pour le XUL, la formation dispensée par Mozilla nous a permis d'être à l'aise pour la mise en place les interactions avec les interfaces. De plus, le fait d'avoir plusieurs années de pratique de JavaScript derrière nous à constitué un avantage certain lors de son utilisation. Du fait des nombreuses requêtes basées sur la technologie AJAX émises par FoaFox, nous aurions pu être handicapée par le fait que les navigateurs actuels bloquent ce type de communication entre deux serveurs distincts. Fort heureusement, cette restriction n'est pas appliquée dans les extensions Firefox. De plus amples détails sur cet aspect du projet seront fournis plus loin dans ce rapport.

CSS

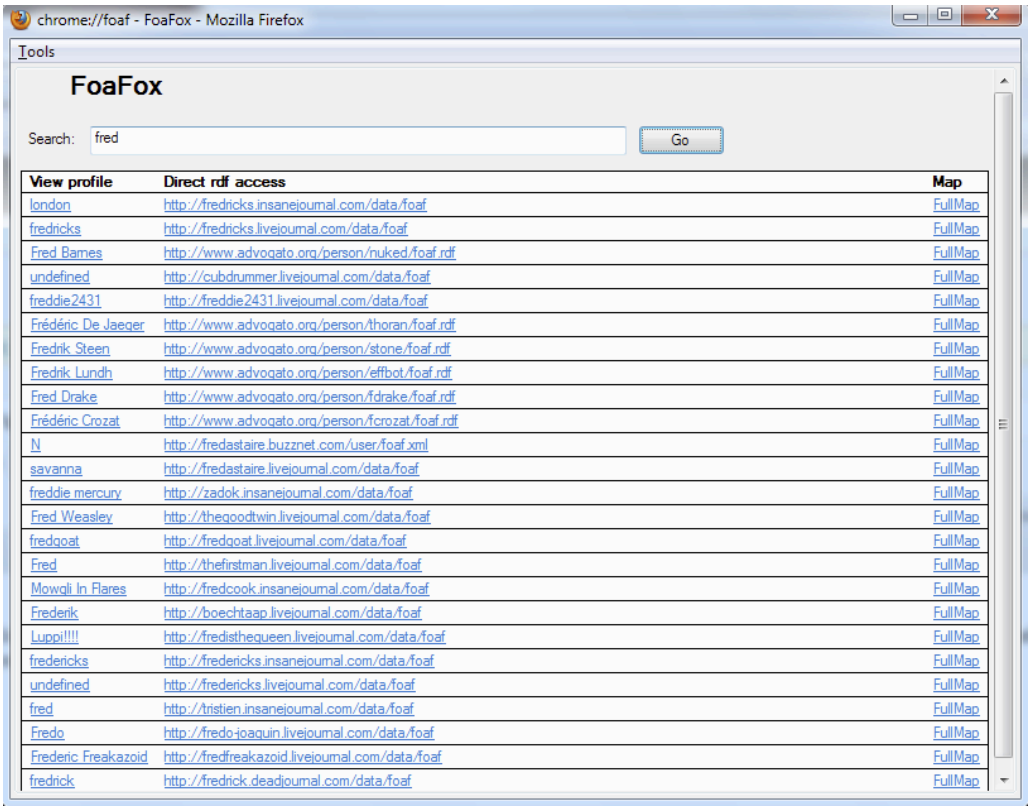
L'un des avantages de XUL est comme nous l'avons vu précédemment sa compatibilité avec les feuilles de style CSS. Nous avons donc pu "styliser" nos interfaces afin d'obtenir le rendu exact que nous avions prévu. Comme pour JavaScript, nos quelques années de pratique nous ont permis d'utiliser CSS à bon escient et sans rencontrer de problème majeur.

Interface utilisateur

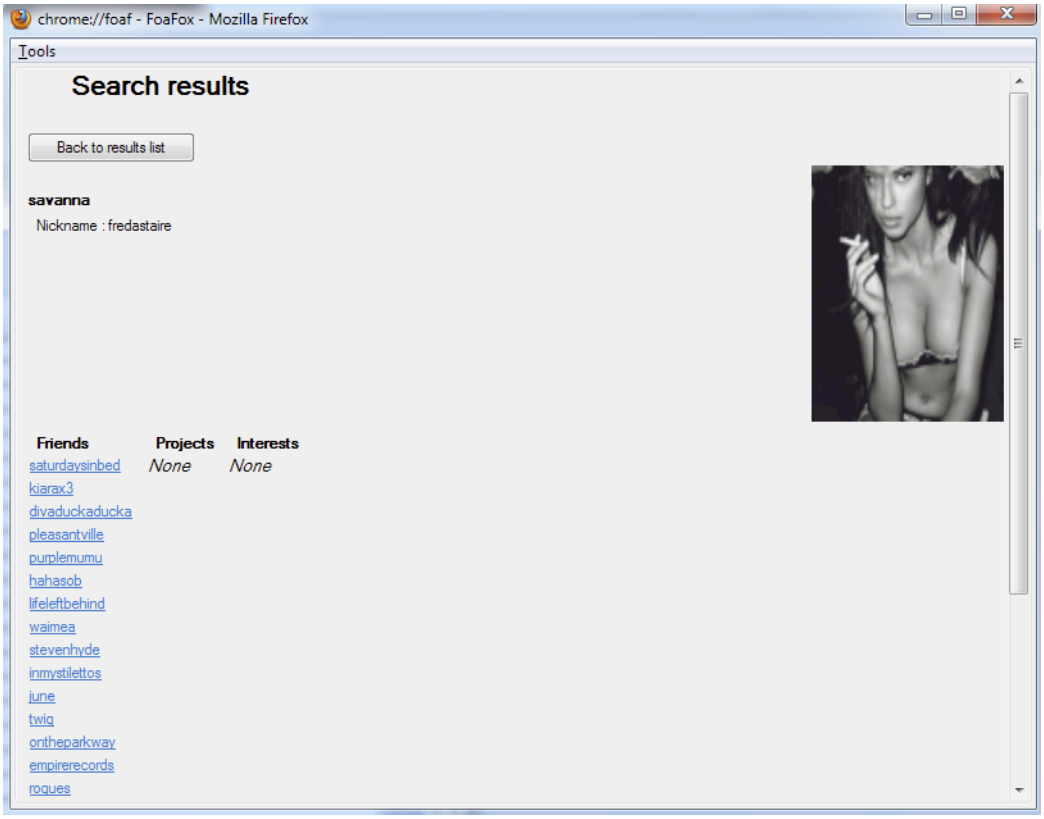
Cette section est destinée à présenter l'ensemble des interfaces disponibles dans FoaFox ainsi que leur fonctionnement. Cette présentation sera découpée par fonctionnalité afin de la rendre plus lisible. Les différentes fonctionnalités sont accessibles via le menu "tools" disponible dans l'extension :

Rechercher

Cette fonctionnalité est affichée dès l'ouverture de FoaFox. Il s'agit d'un système de recherche classique qui permet la saisie d'un mot-clé et la soumission de la recherche. Lors de la réception de la réponse envoyée par le service, les résultats sont affichés comme suit.



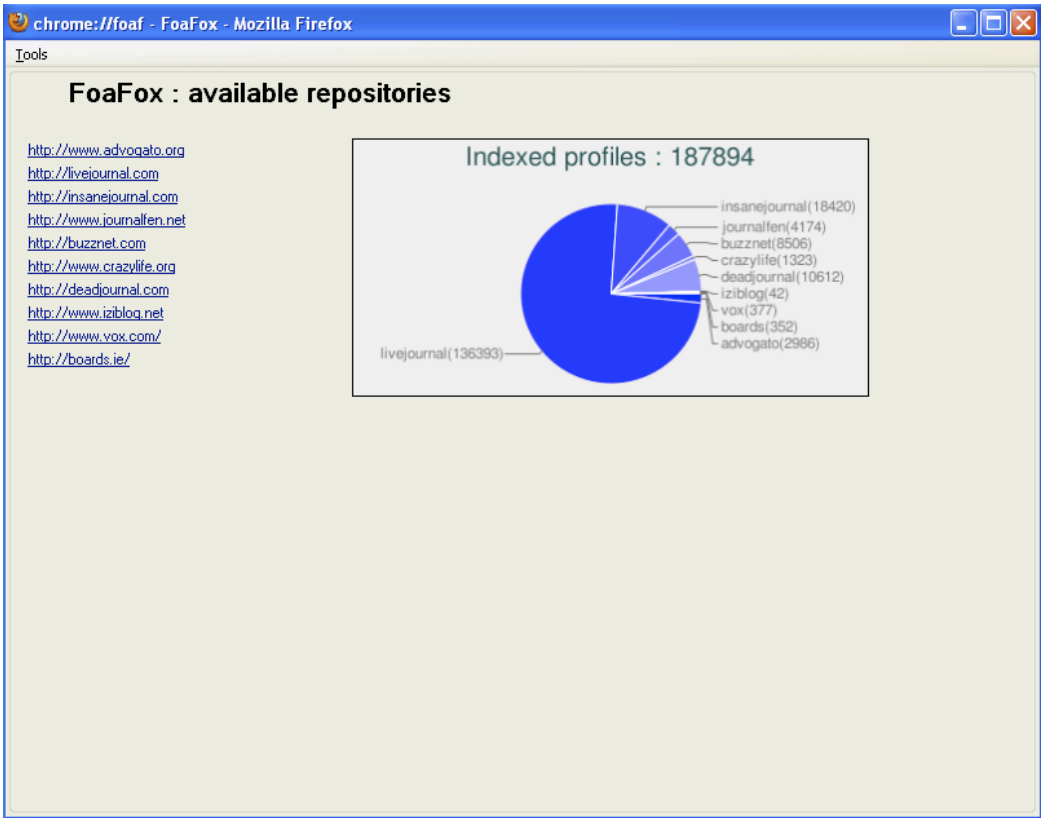
Pour chaque profil trouvé, l'utilisateur peut au choix accéder au fichier FoaF intégral en cliquant sur son URL, ou afficher une page de détails en cliquant sur son nom de profil. Un lien est également prévu pour permettre l'affichage d'une cartographie basée sur le profil sélectionné (voir "cahier des charges et spécifications"). Un exemple d'une page de détails est disponible ci-dessous :



Pour chaque ami listé dans la page de profil, il est possible d'afficher le profil détaillé en cliquant sur son nom. A tout moment, l'utilisateur peut revenir aux résultats de la recherche en cliquant sur le bouton "back to results list".

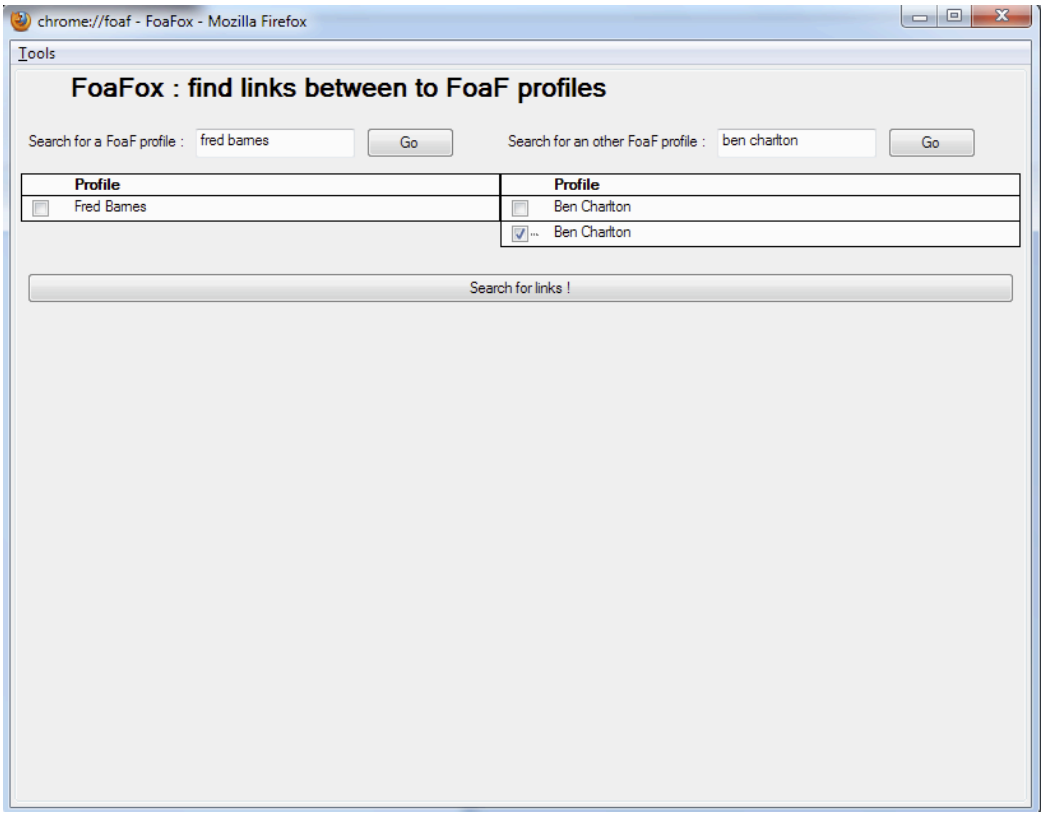
ListerDepots

Cette fonctionnalité très simple permet d'afficher la liste des dépôts utilisés pour peupler la base de données du service. Elle permet aussi de connaître la proportion de chaque dépôt à ce remplissage via l'affichage d'un graphique :

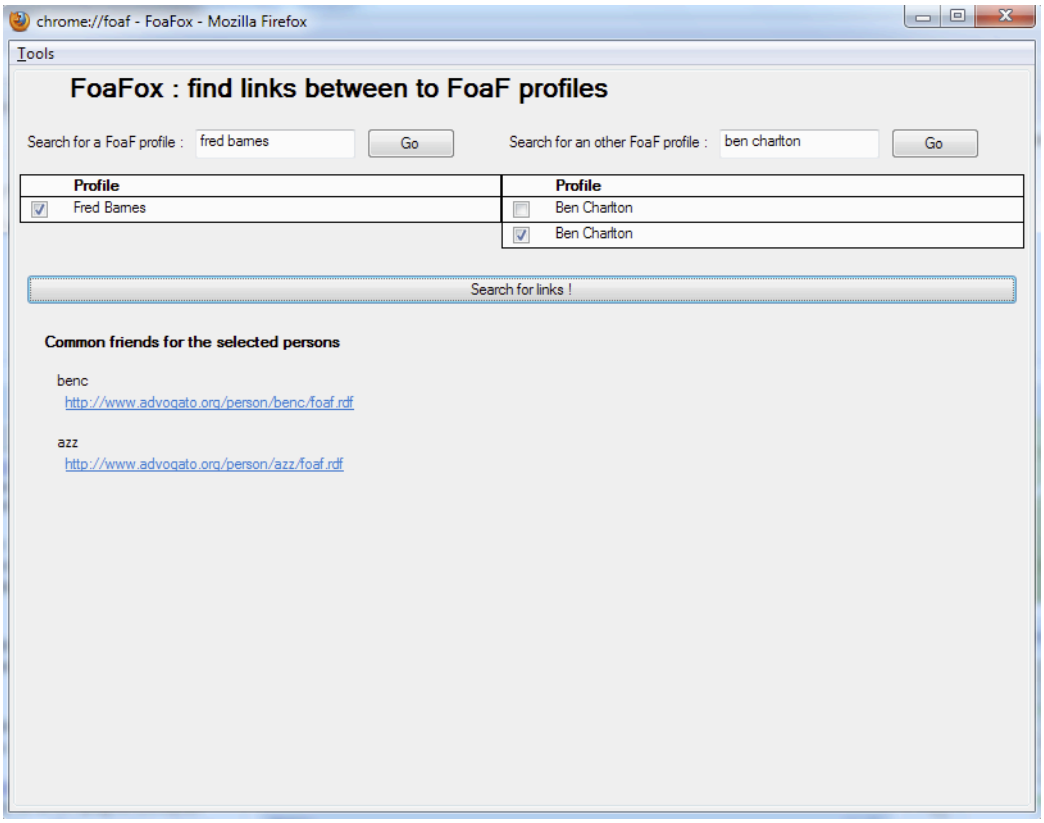


TrouverLien

Cette fonctionnalité est réalisée en deux étapes. Dans un premier temps, il convient de rechercher deux profils afin que l'utilisateur choisisse ceux qu'il veut comparer. Cela est rendu possible par la ré-utilisation du module de recherche :



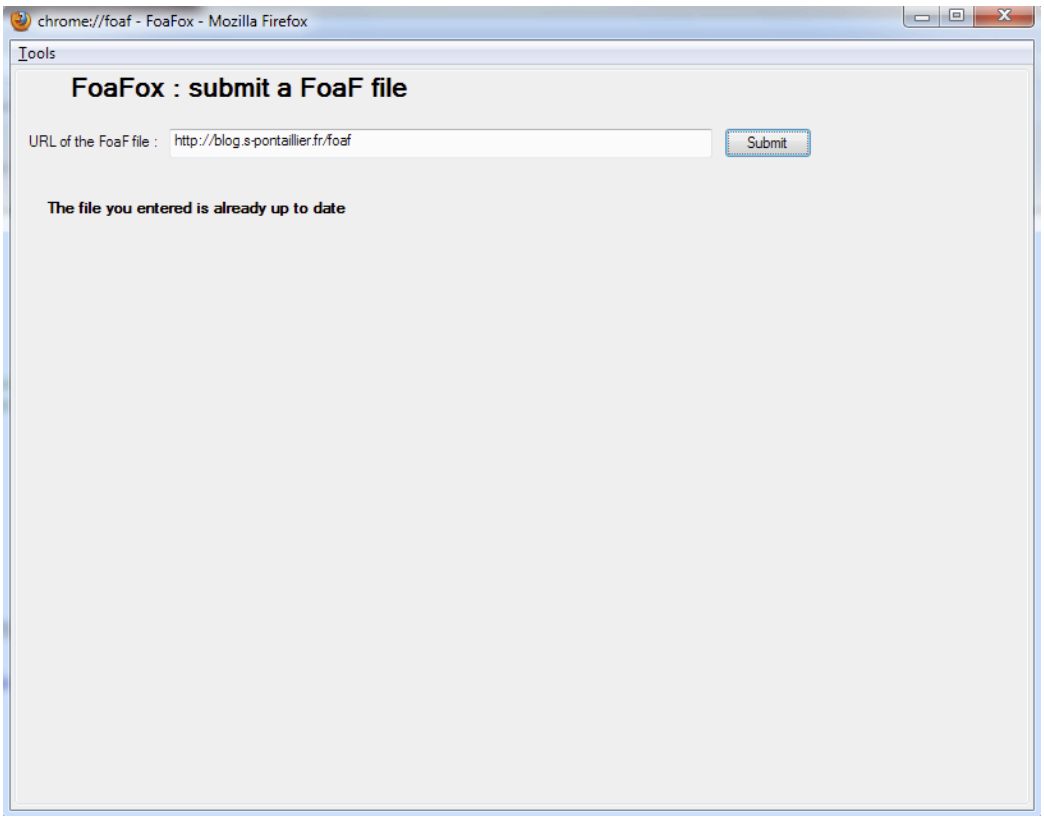
Une fois les résultats affichés, l'utilisateur sélectionne les profils qu'il veut comparer et envoie la requête. Les informations communes au deux profils sont affichés dès réception de la réponse. Si aucun point commun n'a été trouvé entre les deux profils, l'extension le signale à l'utilisateur.



SoumettreProfil

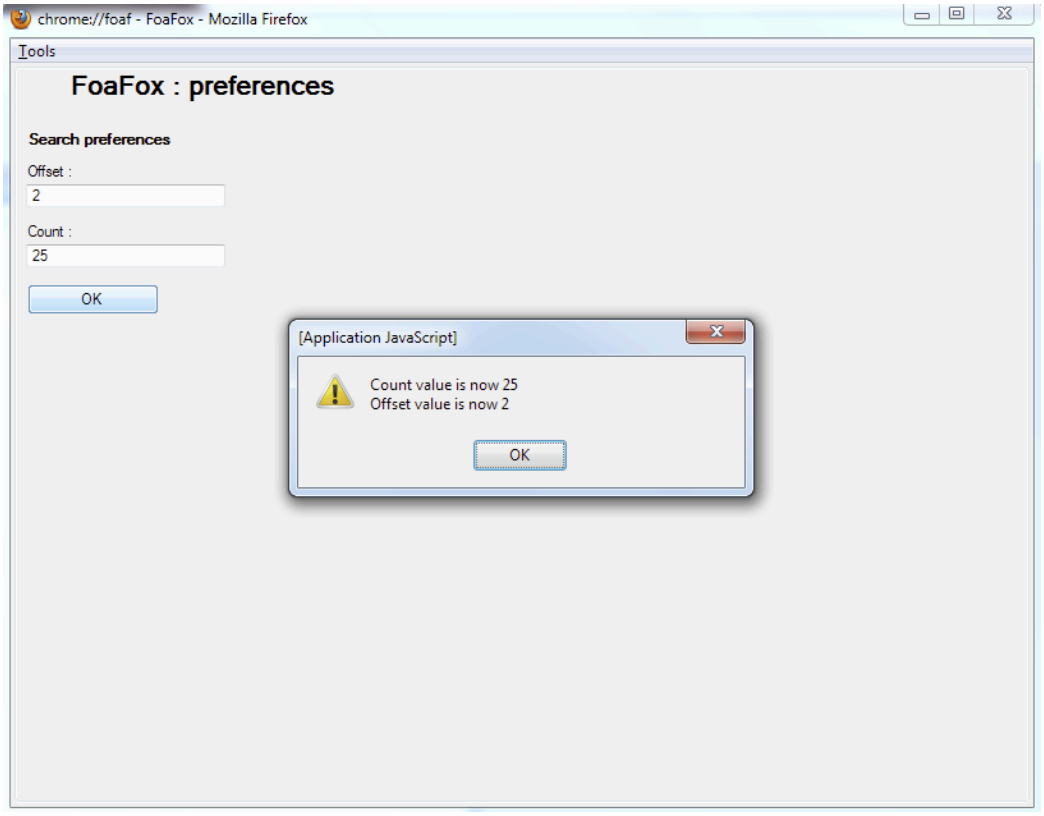
La soumission d'un profile FoaF à la base de données de FoaFinder peut actuellement être effectuée via le formulaire correspondant intégré à l'extension. Après envoi de la requête et réception de la réponse, FoaFox informe l'utilisateur sur la manière dont à été traitée la ressource proposée. Il est prévu d'ajouter un lien à la barre d'adresse de Firefox pour rendre possible la soumission de profiles directement lors de la navigation (voir "cahier des charges et

spécifications").



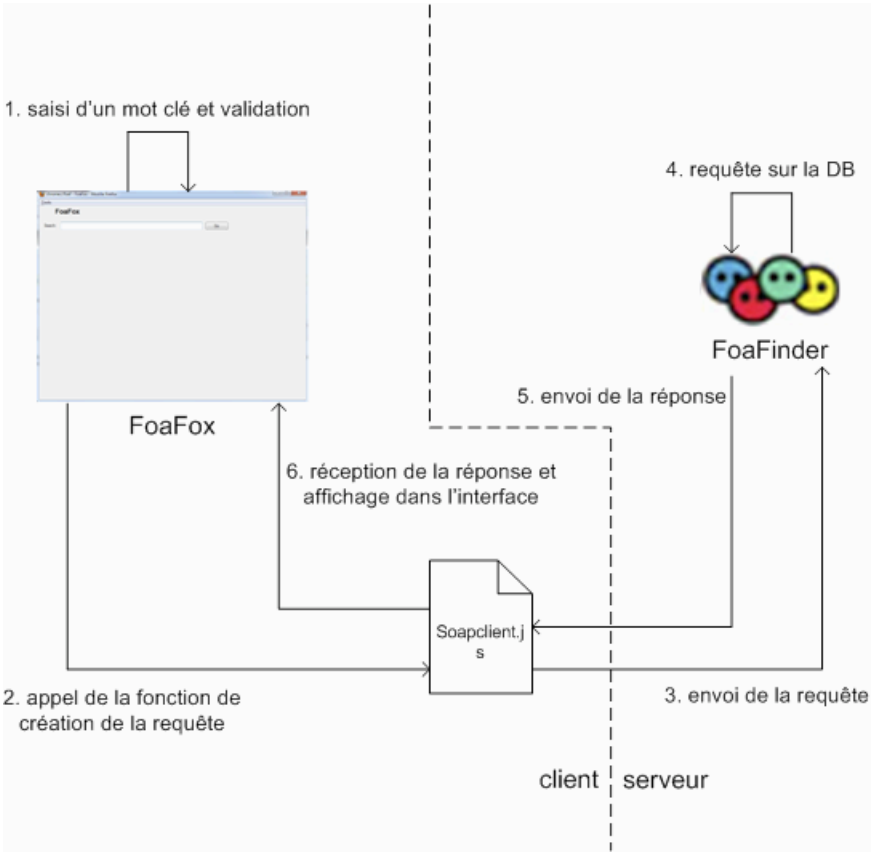
Configurer

Un petit écran de configuration des paramètres de la recherche est également disponible. Il permet de fixer le nombre de résultats à afficher lors d'une recherche. Le paramètre "offset" permet quand à lui de décaler l'index du premier résultat à envoyer. Par exemple, si on définit "count=40" et "offset=10", le service renverra 30 résultats, à partir du dixième résultat trouvé.



Requêtes au WebService

Le coeur de notre projet a consisté à la mise en place d'un web service. Chaque fonctionnalité du client (excepté sa configuration) est donc réalisée via l'envoi de la requête correspondante à FoaFinder. Pour établir une communication avec le service, nous avons utilisé une librairie JavaScript mise à disposition par Matteo Casati, qui est disponible à l'adresse http://www.guru4.net/. L'utilisation de cette librairie, sobrement nommée "soapclient.js", nous a permis d'économiser le temps nécessaire au développement d'une librairie "maison", mais nous a aussi posé quelques problèmes lors de son exploitation (voir "difficultés rencontrées"). Son principe de fonctionnement est néanmoins assez simple : appel d'une fonction pour envoyer une requête SOAP, dans laquelle on spécifie la fonction JavaScript à appeler lors de la réception de la réponse. Le formatage et l'envoi de la requête ainsi que la réception et l'acheminement de la réponse sont entièrement gérés par la librairie soapclient.js. La mise en page de la réponse est quand à elle assurée par FoaFox. Le schéma suivant décrit ce fonctionnement de manière visuelle à travers l'exemple de la recherche :



Comme l'envoi d'une requête via la librairie soapclient.js est fait de la même manière pour chaque fonctionnalité, nous allons conserver l'exemple de la recherche pour passer en revue l'implémentation de l'envoi d'une requête et de la réception de sa réponse :

```
/* fonction appelée lors du clic sur le bouton "go" dans la fenêtre de recherche */
function doSearch() {

  //récupération du mot-clé entré par l'utilisateur
  criterion = document.getElementById("search").value;

  //création de l'objet permettant de passer des paramètres avec la requête
  var pl = new SOAPClientParameters();

  //ajout des différents paramètres à la requête : mot-clé, nombre de résultats à afficher, et offset
  pl.add("keyword",criterion);
  pl.add("offset",search_offset);
  pl.add("count",search_count);

  //autres options
  var wsdl = "http://x-home.hd.free.fr/admin/miage/foafinder/foafinder.xml";
  var method = "Search";
  var async = true;
  var callback = answerSearch;

  //appel de la fonction de soapclient.js permettant de construire et d'envoyer la requête SOAP
  SOAPClient.invoke(wsdl,method,pl,async,callback);
}


//fonction appelée lors de la réception de la réponse à la requête prenant en paramètre l'objet réponse
function answerSearch(r) {

  //utilisation de l'objet réponse pour déterminer le nombre de résultats reçus
  nb = r.documentElement.getElementsByTagName('set')[0].getAttribute('nb');

  //extraction du contenu désiré de l'objet réponse (stockage en variable globale)
  foafs = r.documentElement.getElementsByTagName('foaf');

  //fonction utilisée pour afficher les résultats en utilisant l'objet réponse
  displayResultsList();
}
```

Dans le cadre de la recherche, nous avons également fait appel au langage de lecture XML E4X (spécifications et exemples : https://developer.mozilla.org/en/E4X_Tutorial) permettant d'accéder facilement aux attributs et au contenu des balises contenus dans le fichier XML lu. Un exemple d'utilisation de cette

technologie est disponible ci-dessous.

```
//définition d'un espace de noms utilisé par la suite par les commandes E4X
var ns = new Namespace("rdf","http://www.w3.org/1999/02/22-rdf-syntax-ns#");

//instanciation d'un objet permettant de récupérer la réponse dans le bon format
var s = new XMLSerializer();

//récupération de la réponse
var st = s.serializeToString(xhr.responseXML);

//instanciation d'un objet JavaScript XML sur lequel il est possible d'utiliser la syntaxe E4X
var x = new XML(st);

//récupération du contenu de la balise "name" elle-même contenue dans la balise "person"
var name = x.*::Person.*::name;

//récupération du contenu de la balise "nick" elle-même contenue dans la balise "person"
var nick = x.*::Person.*::nick;

//récupération de la balise "resource" dont l'espace de nom est défini dans la variable "ns"
//et qui est contenue dans la balise "img", elle même contenue dans la balise "person"
var depiction = x.*::Person.*::img.@ns::resource;
```

Problèmes rencontrés et solutions apportées

Bien que l'utilisation de la librairie soapclient.js semble au premier abord très simple, il s'est en premier lieu avéré très difficile d'implémenter son utilisation et l'appel de ses fonctions. Lors des premiers tests que nous avons effectué, nous nous sommes basés sur les exemples que le développeur a mis à disposition sur la page de sa librairie (http://www.guru4.net/articoli/javascript-soap-client/demo/en.aspx). Les premières implémentations se sont avérées ne pas fonctionner. Après quelques recherches, nous avons découvert que la communication via la technologie AJAX entre domaines distants était nativement bloquée par la plupart des navigateurs. En suivant cette piste, nous avons modifié certains paramètres de Firefox pour tenter de contourner cette sécurité. Ayant constaté que ces modifications n'avaient pas résolu notre problème, nous avons par la suite appris que cette restriction ne s'appliquait pas aux extensions : notre problème se trouvait ailleurs. Nous avons alors étudié de manière plus poussée la librairie, ce qui nous a conduit à constater un bug dans le code qui conduisait à l'envoi d'une requête pointant sur une service inexistant. En effet, l'URL passée lors de l'appel de la fonction (voir section précédente) n'était pas acheminée correctement par la librairie. Nous nous sommes donc vus forcés de passer l'URL en dur directement dans la librairie, avant l'envoi de la requête. Le service étant alors normalement détecté, l'envoi des requêtes et la réception des réponses a pu se faire de manière normale.

Nous avons été confrontés à un autre problème, concernant le traitement des réponses aux requêtes. La manière dont nous avons implémenté le service entraîne le renvoi d'une réponse sous la forme d'un objet XMLDocument. La lecture de cet objet a été une tâche fastidieuse. Nous avons d'abord tenté de l'utiliser pour instancier un objet JavaScript XML sur lequel il est possible d'utiliser la syntaxe E4X (voir section précédente), mais cela s'est avéré impossible : l'objet récupéré était systématiquement vide. Après avoir effectué quelques recherches, la solution que nous avons adoptée est l'utilisation de la propriété "documentElement" de l'objet réponse, qui permet une fois utilisée de parser celle-ci comme un objet DOM standard. L'exemple ci-dessous exprime bien l'implémentation que nous avons choisi :

```
nb = r.documentElement.getElementsByTagName('set')[0].getAttribute('nb');
```

Bilan et ouverture

La réalisation de ce projet nous a avant tout permis de mieux comprendre les interactions ayant lieu entre un client et un WebService. Nous avons entre autres pu appréhender les contraintes et les restrictions pouvant s'appliquer à l'utilisation d'un web service par un client JavaScript : debug difficile, accès peu aisé au service, mais aussi difficulté d'appropriation de la librairie utilisée. En revanche, nous avons également pu constater et utiliser toute la puissance des web services, qui permettent la réalisation de traitements distants sans surcharger le client. Le seul problème de ce type d'architecture peut être le temps de réponse à une requête, qui est déterminé par l'environnement extérieur du client (bande passante, serveur...).

Concernant le client, nous sommes assez satisfaits du travail accompli, puisque la quasi totalité du cahier des charges a pu être respectée. On peut néanmoins imaginer quelques améliorations, comme la mise en place d'une interface de configuration avancée, ou encore la transformation des interfaces pour leur donner une touche plus conviviale et unique.

Le WebService quant à lui, pourrait être amélioré à plusieurs niveaux : tout d'abord, le requêtage SQL n'est peut-être pas le plus approprié sur le type de données stockées (des fragments XML essentiellement), le SPARQL pourrait apporter d'énormes gains en terme de performance. Il existe un framework Java appelé Jena (http://jena.sourceforge.net/index.html) capable de traduire des requetes RDF en SPARQL. Un projet de portage de cette librairie vers PHP semble avoir débuté, il s'agit de Pena (http://www.semwebcentral.org/projects/pena/), qui pourrait s'avérer très utile dans le projet FoaFinder, sans même provoquer de gros changement dans l'architecture existante (la conservation des interfaces du WebService, du moteur et de la base de données MySQL semble possible).

Exemple d'une requête SPARQL (source: http://fr.wikipedia.org/wiki/SPARQL) :

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?nom ?image ?description
WHERE {
  ?personne rdf:type foaf:Person .
  ?personne foaf:name ?nom .
  ?image rdf:type foaf:Image .
  ?personne foaf:img ?image .
  ?image dc:description ?description
}
```

Exemple d'un résultat de requête SPARQL (source: http://fr.wikipedia.org/wiki/SPARQL) :

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
```

```
    <variable name="nom"/>
    <variable name="image"/>
    <variable name="description"/>
  </head>
  <results ordered="false" distinct="true">
    <result>
      <binding name="nom">
        <literal>Pierre Dumoulin</literal>
      </binding>
      <binding name="image">
        <uri>http://example.net/Pierre_Dumoulin.jpg</uri>
      </binding>
      <binding name="description">
        <literal>Photo d'identité de Pierre Dumoulin</literal>
      </binding>
    </result>
    <result>
      <binding name="nom">
        <literal>Paul Dupont</literal>
      </binding>
      <binding name="image">
        <uri>http://example.net/Paul_Dupont.jpg</uri>
      </binding>
      <binding name="description">
        <literal>Photo d'identité de Paul Dupont</literal>
      </binding>
    </result>
  </results>
</sparql>
```

Enfin, nous imaginons proposer ce projet à la communauté Open Source, peut-être via Sourceforge ou autre SVN-like, afin de prolonger et de perfectionner ces outils.

Pensez à l'environnement avant d'imprimer ce document.
Document xHTML mis en forme avec CSS et optimisé pour des navigateurs respectant les standards du web.
Dernière mise à jour : 09/02/2010
Contact : Frédéric Eterno (frederic.eterno@gmail.com) / Sébastien Pontaillier (sebastien.pontaillier@gmail.com)