Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de diplôme 2008

Filière Informatique de gestion

Web 3.0 : le Web comme base de données

Etudiant : Antoine Darbellay
Professeur : Anne Le Calvé

PRÉFACE

Actuellement, Internet représente sans conteste la plus grande source d'informations disponibles pour chacun. Néanmoins, tout le monde s'accorde à dire que cette énorme masse d'informations n'ai pas aisément exploitable. Une bonne maitrise des outils de recherche ainsi qu'une certaine dose de patience sont souvent nécessaire pour trouver l'explication que l'on souhaite.

Pour répondre à cette problématique, Tim Berners-Lee, co-inventeur avec Robert Cailliau du World Wide Web, a eu l'idée géniale d'ajouter une description à l'information, la rendant ainsi plus aisément accessible à une personne ou une machine. Il s'agit d'un vaste projet qui vise à donner du sens à l'information ou en d'autres termes, à apporter une signification à un document.

Le projet Memoria-Mea et son moteur Onto-Mea s'inscrivent dans cette direction en cherchant à nous donner les moyens de classer et d'interroger sémantiquement tous les documents multimédia résidents sur notre ordinateur.

Ainsi, en partant d'une idée révolutionnaire et avec le soutient d'une communauté de développeurs passionnés, il est aujourd'hui possible de stocker et de retrouver intelligemment de l'information, et même d'en construire une nouvelle en se basant sur une logique de déduction.

TABLES DES MATIÈRES

SIGNIFICATION DES COULEURS ET POLICES

- Idées importantes : **le Web « intelligent »**
- Partie de code: echo « bienvenue dans le Web sémantique » ;
- Bloc de code : `echo « bienvenue dans le Web sémantique » ;`
- Citation : « *Avec l'ère des machines, beaucoup d'esprits se croient robots.* »
- Source du document: *(Source : www.wiki.com)*
- Liens à voir pour plus d'informations*: (Voir : www.wiki.com)*
- Note sur un sujet : 
- Remarque sur le projet : 

1. PRÉSENTATION DU TRAVAIL

1.1 L'INFORMATION UTILE

Si l'on regarde la situation actuelle, seuls l'esprit humain est capables de comprendre l'information que l'on trouve sur Internet et c'est lui qui décide en quoi cela se rapporte à ce qu'ils cherchent vraiment.

Bien sur, nous pouvons compter sur des moteurs de recherche très sophistiqués, cependant leurs limites ont déjà été atteintes et la profusion de pages Web indexées dans le monde ne fait qu'accroitre ce travail de sélection de l'information pour l'utilisateur. On pourrait résumer les capacités d'un moteur de recherche en ceci: 'Quelles sont les pages contenant les termes *X* ?' et 'Quelles sont les pages les plus populaires au sujet de *Y* ?'.

L'objectif du Web sémantique est d'arriver à **un Web 'intelligent'**, où les informations ne seraient plus stockées mais 'comprises' par les ordinateurs afin d'apporter à l'utilisateur ce qu'il cherche vraiment.

En 2000, l'inventeur du Web, Tim Berners-Lee, donnait la définition suivante: « *Le Web sémantique est une extension du Web classique où l'information reçoit une signification bien définie améliorant les possibilités de travail collaboratif entre les ordinateurs et les machines.* » En d'autres termes, si le Web est une grande bibliothèque, il est plus facile de consulter son catalogue que de parcourir toutes ses allées.

Ainsi, le Web Sémantique, cherche à fournir les moyens de développer des outils qui dépasse le simple affichage d'information, mais permettent au contraire d'automatiser des requêtes complexes à travers diverses applications. **Le but est de transformer la masse ingérable des pages Web en un gigantesque index hiérarchisé**. Les moteurs de recherche seront alors en mesure de répondre à des demandes précises, comme "Trouve-moi un hôtel en Valais qui accepte les chiens et que se situe dans une région propice aux ballades en familles.

En effet, l'idée du Web Sémantique n'est pas de faire en sorte que les ordinateurs puissent comprendre le langage humain ou fonctionner en langage naturel; il ne s'agit pas d'une intelligence artificielle permettant au Web de réfléchir, mais simplement de **regrouper l'information de manière utile**, comme pour une gigantesque base de données, où tout est écrit en langage structuré.

(Texte inspiré de l'article:
http://www.journaldunet.com/developpeur/tutoriel/xml/021115xml_websemantique1a.shtml et
http://www.clever-age.com/actualites/dernieres-actualites/dans-la-presse/decision-informatique/web-semantique-la-toile-prend-tout-son-sens.html)

1.2 DÉROULEMENT DU PROJET

Le but de ce travail de diplôme est de démontrer l'intérêt des nouvelles technologies du Web sémantique dans l'application du tourisme en Valais. L'objectif étant de montrer aux acteurs du monde du tourisme, que cette 'sémantisation' de l'information, et plus particulièrement des informations descriptives de leur régions devraient être mis-à-jour pour permettre la création de nouveaux outils de recherche et de planification de vacances. La solution apporterai ainsi une réelle plus value par rapport aux offres traditionnelles.

Pour parvenir à ce résultat, je devrai dans un premier temps étudier les solutions actuellement disponibles et ainsi me forger une opinion sur l'état d'avancement de d'imbrication des technologies du Web sémantique dans l'Internet d'aujourd'hui.

Suite à cette étude de l'art, je vais devoir concevoir un démonstrateur pour mettre en avant cette technologie. Donc je vais devoir commencer par récolter un échantillon de données. Il est évident que cet échantillon peut prendre plusieurs formes, aussi je vais surtout m'intéresser à la logique qui permet de transformer une exportation de base de données au format XML.

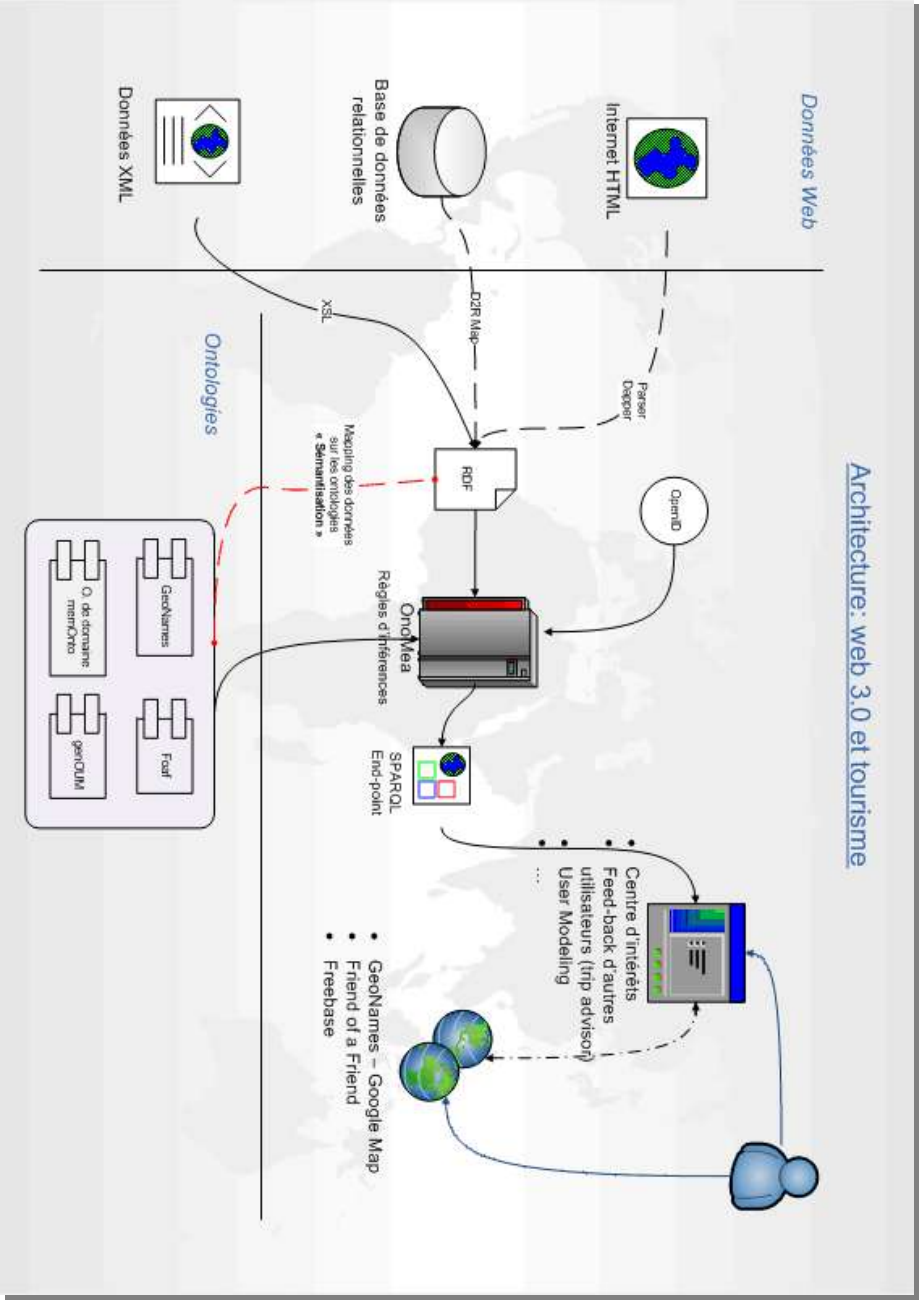
Ensuite, il va falloir sémantiser ces données grâce aux nombreuses technologies permettant de mieux gérer l'information. Pour répondre à ce besoin je vais m'appuyer sur le langage RDF et OWL.
RDF, pour Resource Description Framework, définit la logique et les balises permettant de décrire les métadonnées.
OWL, pour Ontology Web Language, pour représenter les ontologies. Des ontologies qui ne sont rien d'autre que des sortes d'index étendus où les liens entre les concepts sont précisés : synonymes, équivalents, liens vers unconcept de sens plus large, plus précis, etc.
Ainsi, ma solution va combiner les URI, HTTP, RDF(S) et OWL pour bâtir un système de gestion intelligente de l'information.

Une fois ces données sémantisées, il faudra les stocker, heureusement la Hevs dispose déjà d'un moteur répondant en grande partie à mes attentes. De plus, OntoMea dispose d'un raisonneur qui va aussi me permettre de déduire de nouvelles données.

Finalement, il faudra concevoir un démonstrateur qui va s'appuyer sur le langage SPARQL pour l'interrogation de mes données et, présenter ces données désormais interconnectées ainsi que mes nouvelles déductions.

1.3 ARCHITECTURE GÉNÉRAL DU SYSTÈME

Sur ce schéma, on voit à droite les sources de données provenant d'Internet ou d'une base de données.
Sur le fond, on voit certaines des ontologies chargées dans le moteur OntoMea.
Au centre, le cœur de système de traitement, toutes les données sont transformées au format RDF puis poussées dans le moteur OntoMea.
Ensuite le front-end SPARQL permet d'accéder aux données et une interface Web présente les informations avec d'éventuelles liaisons sur d'autres sites.



Je vais essayer dans les chapitres suivants de vous présenter au mieux un aperçu de la situation actuelle ainsi que les différentes techniques, technologies et approches que j'ai suivies tout au long de ce travail.

2. ETAT DE L'ART

2.1 SITUATION ACTUELLE

Suite à mes recherches sur les différents projets liés au Web sémantique et plus précisément à l'application de cette technologie dans le monde du tourisme, je constate que bien que le Web 3.0 a déjà atteint une certaine maturité, surtout au niveau structurel (RDF, OWL, SPARQL…) les projets concrets ne foisonnent pas.

« L'initiative du Web sémantique reste encore aujourd'hui marginale face aux problématiques courantes du Web, comme par exemple la publication de contenu. Pourtant, le Web sémantique est déjà riche de nombreux langages et outils qui peuvent trouver leur place au sein de l'entreprise, et améliorer la manière dont celle-ci traite, organise et publie ses données. »
(Source : http://www.clever-age.com/veille/clever-link/le-web-semantique-en-entreprise-comment-et-a-quels-niveaux.html)

Il existe de nombreuses applications pratiques basées sur cette technologie, mais rien de révolutionnaire dans notre utilisation quotidienne d'Internet.
Voici quelques applications qui s'annoncent très prometteuses :

- Freebase, Twine: un répertoire global d'informations structurées, intelligibles et exploitables autant par des machines que par des humains
- ClearForest: outils d'analyse de texte qui permettent de sémantiser une page, par exemple, signaler tous les termes qui relève de noms de société
- Powerset, TrueKnowledge, Hakia: les moteurs de recherche en langage naturel
- Spock: le moteur de recherche de personnes
- Adaptive Blue: outils de raccourcis intelligents

Avis d'Alex Iskold

Pour reprendre les propos d'Alex Iskold (journaliste pour Read/WriteWeb et fondateur et CEO de la société AdaptiveBlue)

« Il écarte rapidement la compréhension du langage humain, qui, tout en demeurant le saint Graal du Web sémantique, ne sera pas accessible avant longtemps. Tout comme le fait que le Web sémantique puisse demain permettre aux ordinateurs de résoudre des problèmes complexes, comme trouver un lieu de vacances qui vous corresponde, car comprendre ce type d'information nécessite un processus complexe (Où avez-vous déjà été ? Avec qui comptez-vous partir ? Qu'aimez-vous faire ? Quel est votre budget ?…), qui demande de multiples itérations et de la mémoire et pas seulement un profil couplé à des champs de données. »
(Source :http://www.readwriteweb.com/archives/semantic_web_what_is_the_killer_app.php)

Selon Alex Iskold, les objectifs atteignables à court et moyen terme reposent sur les bases de données de connaissances sémantisées, à la Freebase ou à la Twine. Le premier construisant un Wikipédia sémantisé, le second permettant de construire ses propres bases de données sémantisées.

Bien que ces propos ne soient pas très encourageants, je reste certain que la sémantisation d'informations touristiques pourrait être un vrai plus dans un processus de planification de vacance.

De plus, même si l'on n'arrive pas aujourd'hui à crée un véritable moteur commerciale qui va proposer une voyage sur mesure à tous les touristes de notre région, on peut déjà imaginer sémantiser un certain nombre de données et fournir un service intelligent sur certain aspect pratique de la recherche d'informations sur Internet.

Sans prétendre révolutionner le monde des agences de voyage, je vais quand même essayer de créer un démonstrateur simple pour prouver l'utilité du Web sémantique appliqué au domaine du tourisme en Valais. Voici un bref aperçu des projets actuels.

2.2 PROJETS SÉMANTIQUES LIÉS AU TOURISME

Eiffel

« L'objet du projet EIFFEL est de disposer d'une solution technique pour la mise en œuvre de moteurs de recherche spécialisés tourisme/territoire, accessibles via Internet, donnant une information riche, précise, contextualisée aux utilisateurs et permettant au territoire de valoriser leur offre et de conduire leur politique de marketing.

Eiffel est donc un projet de mise en œuvre d'une plateforme de collecte, d'analyse, de consolidation, de classement et de mise en relations d'informations dans le domaine du tourisme pour un territoire donné afin permettre la valorisation de l'offre au niveau du territoire, et de faciliter l'accessibilité et la compréhension de l'offre du coté des utilisateurs. »
(Source: http://panini.u-paris10.fr/jlm/?u_s=0&u_a=25&sid=)

Bien que ce projet réponde exactement à ce que nous recherchons, le projet est actuellement en cours de développement et nous n'avons accès à aucunes informations.
(Voir : http://www.rntl.org/projet/resume2005/eiffel.html)

Tripit

« Ce service Web est un guichet unique qui vous permet de rassembler toutes les informations relatives à l'organisation de votre voyage. Que ce soit une réservation d'hôtel, de billet d'avion, de carte géographique ou de restaurant, Tripit vous permet de centraliser toutes vos informations à un seul endroit et de les partager avec des collègues ou amis. »
(Source : http://descary.com/tripit-loutil-ultime-du-voyageur/)

Pour avoir un peu testé le produit, j'ai été très déçu… ce site ne fait que rassembler et présenter des documents que l'on doit fournir sans y ajouter aucune vrai service supplémentaire.
(Voir : www.tripit.com)

Hi-Touch

Délivrer un outil d'assistance pour les agences de voyages basé sur des nouvelles technologies (XML, Java, Flash, base de données ontologique et descripteurs sémantiques) pour permettre au client de choisir efficacement et rapidement un voyage qui répond à tous leurs souhaits.
(Voir : http://www.ist-world.org/ProjectDetails.aspx?ProjectId=218cef0541804546bccdc372588d3d70)

CapeLing

« CapeLink permet d'intervenir sur l'intégralité de la chaîne de valeur de l'information touristique, c'est-à-dire sur l'ensemble des activités qui s'enchaînent pour aboutir à la vente de vos produits ou services. »

Bien que cette solution ne semble par reposer sur des technologies liées au Web 3.0, cette solution mérite quand même d'être mentionnée vu son nombre conséquents de fonctionnalité ainsi que sa grande clientèle.
(Voir : http://www.mondeca.com/capelink/fr/index.html)

2.3 PROJETS AJOUTANTS UNE COUCHE SÉMANTIQUE À L'INFORMATION

OpenCalais

OpenCalsis est un service Web étonnant qui a été récemment rendu public par l'agence Reuters. En résumé, OpenCalais prend arbitrairement du texte ou du contenu HTML en entrée et essaie d'en extraire les entités Web sémantique.

Le service Web Calais vous permet d'annoter automatiquement du contenu avec des métadonnées sémantique, y compris des entités comme des personnes, des entreprises, des événements et des faits comme des acquisitions et des réformes de gestion.

- Relations: Acquisition, CompanyInvestment, PersonProfessionalPast
- Organization: Palo Alto Research Center
- IndustryTerm: broader search development effort, text search, text analytics software
- Company: Time Warner Inc.,Reuters, Pitango Venture Capital, ClearForest Ltd
- Person: Antoine Darbellay
- Country: United States, Suisse
- City: New-York, Sierre, Genève

Il existe un stater-kit écrit en Java que l'on peut télécharger sur leur site. Il fournit le client nécessaire pour appeler un Web service. Il suffit de spécifier la source d' entrée en paramètre, source qui n'est rien du plus qu'un fichier au format texte.

Je l'ai essayée avec un texte anglais du site valaistourism.ch (il ne supporte que l'anglais actuellement) et les résultats sont assez moyennement intéressant et difficilement exploitables.
Sur un texte complet de description de la région d'Anzère, le service ne 'tag' seulement que le mot 'Alps' et 'Mediterranean' comme région…

En revanche si je prends le texte fournit en exemple ou un texte du genre : « My name is Antoine Darbellay and I live Verbier in Switzerland » le résultat est nettement plus probant.

Ci-dessous, une impression d'écran du site http://autotagger.opensynapse.net/ qui permet de passer directement sur la page du site un texte à analyser.

Etant donnée les résultats, on peut imager que cette analyse du texte doit être reposé sur une analyse syntaxique du contenu via un processus relativement complexe. Mais il n'y rien de magique là-dessous, si je mets un nom propre sans contexte, je n'ai pas de résultat…

(Voir: http://www.opencalais.com/)

Nepomuk

« IBM, SAP, HP, Mandriva, pour ne citer qu'eux, viennent de s'engager dans le projet Nepomuk.
Un acronyme pour Networked Environment for Personalized, Ontologybased Management of Unified Knowledge. Planifié sur trois ans, ce projet veut définir le poste de travail sémantique de demain. L'objectif est de définir les standards et de développer des API pour l'indexation, la recherche, le partage et la visualisation des métadonnées.

Le cœur des logiciels Nepomuk intégrera un index de toutes les informations textuelles: mails, documents bureautiques, agenda, carnet d'adresses, bookmarks, liste des tâches... Mais cet index ira au-delà de la prise en compte des mots inscrits dans ces documents, comme c'est le cas des technologies actuelles. Sur le modèle de la déduction humaine, il cherchera les liens existants entre les mots, à commencer par le sens des phrases. Le poste de travail sémantique est communément défini comme l'ensemble des technologies qui rendent les ordinateurs capables de recueillir des informations à la manière du cerveau humain. »

« Nepomuk vise aussi à assister les utilisateurs dans la création des documents en leur fournissant des informations contextuelles liées, par exemple, au texte qu'ils sont en train de taper. Une assistance qui se présentera sous la forme de zones spécifiques dans les applications, où l'utilisateur se verra proposer des liens vers d'autres documents sur le même sujet. »
Il s'agit d'un vaste projet très intéressant et disponible depuis peu sous java sous le nom de NightlyBuilds. Je n'ai malheureusement pas pu tester cette solution car elle n'était malheureusement pas sorti quand j'ai effectué ce tour d'horizon.
(Source : http://www.clever-age.com/actualites/dernieres-actualites/dans-la-presse/decision-informatique/web-semantique-la-toile-prend-tout-son-sens.html et http://www.zdnet.fr/actualites/informatique/0,39040745,39312523,00.htm)
(Voir: http://nepomuk.semanticdesktop.org/xwiki/bin/view/Main1/)

Aperture

Aperture est une bibliothèque libre et un framework pour l'exploration et l'indexation de sources d'informations telles que les systèmes de fichiers, sites Web ou des mails boxes. Aperture prend en charge un certain nombre de formats de documents et fournit des moyens pour en implémenter d'autres.

Le code d'Aperture se compose d'un certain nombre de pièces indépendantes:

- L'exploration des sources d'information: les systèmes de fichiers, sites Web, mail box
- Identification de Type MIME
- Extraction depuis un texte et ses métadonnées sur divers formats

Pour chacune partie, un API a été élaboré et un certain nombre d'implémentations sont fournies. Aperture met fortement l'accent sur les données sémantiques. Par exemple, beaucoup d'efforts sont déployés afin d'extraire le plus de métadonnées contenues dans les formats de fichiers (titres, auteurs, commentaires, ...) et de combiner cela avec des métadonnées spécifiques à une source (l'emplacement, la date de dernière modification, ...).

Il s'agit d'un framework très riche qui est d'ailleurs utilisé dans de nombreux projets d'envergure: Nepomuk Social Semantic Desktop, Virtuoso Server by OpenLink Software, Aduna Autofocus, Waves.
(Voir : http://aperture.sourceforge.net/)

Web content

Produire une plate-forme flexible et générique pour la gestion de contenus et l'intégration des technologies du Web Sémantique dans le but de montrer leur utilité sur des applications réelles à fort impact économique ou sociétal.

La plate-forme vise initialement un groupe d'applications autour de la veille technologique. Plus particulièrement:

- Veille économique dans le domaine aéronautique ;
- Veille stratégique ;
- Risque microbiologique et chimique dans le domaine alimentaire ;

(Source: http://www.webcontent.fr/scripts/home/publigen/content/templates/show.asp?L=EN&P=55&vticker=alleza)

E-wok-hub

Le projet e-Wok Hub a pour objectif de tirer bénéfice des travaux entrepris sur le Web sémantique pour développer des systèmes opérationnels autorisant la coopération sur internet entre différentes organisations (entreprises, instituts, ...) impliqués dans un workflow d'ingénierie.
Objectif : dans le domaine des géosciences, vise à gérer la mémoire de plusieurs projets sur la capture et le stockage de CO2, tout en exploitant les résultats de la veille technologique sur le domaine.
(Source: http://www-sop.inria.fr/edelweiss/projects/ewok/)

SEMTAG d'ibm

SEMTAG est une application de balisage sémantique automatique commercialisée par IBM depuis l'automne 2003 dans le cadre de son produit Webfountain. SEMTAG s'appuie sur l'ontologie TAP [3] développée par l'université de Stanford
SEMTAG récupère les documents bruts ramenés par le crawler de Webfountain (Seeker), et les analyse, en essayant d'identifier les occurrences de chacune des 72000 étiquettes présentes au sein de TAP. Chaque élément ainsi étiqueté est sauvegardé avec une fenêtre de 10 mots (afin de permettre une désambiguation ultérieure).
(Source : http://www.webmaster-hub.com/publication/L-autre-semantique-Le-Web,161.html)

Synomia

Synomia détecte et organise automatiquement les milliers d'expressions clés qui structurent votre site Internet, vous permettant d'optimiser et d'automatiser vos

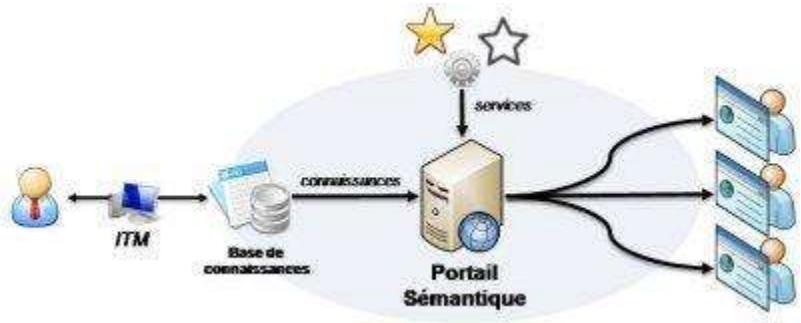
applications de gestion de contenu, de recherche d'information et d'interaction client, en toute simplicité.
Une démonstration de ces outils se trouve sur le site du journal Libération.fr.
(Source : http://www.synomia.fr/site/)

Triplify

Triplify est un plugin pour les applications Web qui permet d'exposer une base de données de manière sémantique. Pour implémenter cet outil, il suffit d'écrire quelques requêtes SQL afin de sélectionner les informations que l'on veut dévoiler sous forme RDF ou JSON. Ce plugin est actuellement en développement (beta test) et n'applique qu'à des sites construits en PHP.
(Voir : http://triplify.org/Overview)

Mondeca : portail sémantique

Pour exploiter les bases de connaissances et valoriser leur contenu, Mondeca propose une solution innovante de portail sémantique.
Concrètement, un portail sémantique est un site Internet qui offre une porte d'entrée unique sur un large éventail de ressources et de services centrés sur une base de connaissances.
Ainsi les utilisateurs peuvent évoluer, recherche et naviguer dans leur espace informationnel en exploitant d'une part la sémantique de leurs bases de connaissances et d'autre part utilisant un ensemble de services de haut niveau.



La solution de portail sémantique repose sur une bibliothèque de widgets : des éléments graphiques, à ajouter dans un portail, qui offrent des fonctionnalités avancées.

- Saisie d'une requête en langage naturelle : des zones de saisies pour aider l'utilisateur à soumettre sa requête au service de recherche.
- Affichage et navigation au sein des résultats : différentes approches pour accéder aux résultats de recherches et pour naviguer entre eux.
- Navigation par facettes : visualisation du nombre de résultats par facettes où chaque valeur de facette permet d'affiner la recherche. Cette solution permet de guider les utilisateurs dans leurs recherches.
- Cartographie des résultats : des widgets capables de visualiser les résultats dans le temps, sur des cartes, ou en fonction de leur sémantique
- Cartographie des relations entre les sujets de la base

Gamme de produit extrêmement intéressant chez Mondeca mais payant et prise de contact nécessaire.
(Source : http://www.mondeca.com/index.php/fr/intelligent_topic_manager/applications/semantic_portal_semantic_widgets)

Moteurs de recherche sémantique

- CORESE, développé à l'INRIA
- KartOO
- Ujiko
- Lingway KM est une plateforme linguistique et sémantique multilingue permettant le développement de moteurs de recherche spécialisés
- Seek de Verticrawl, logiciel de recherche en mode webservice (www.verticrawl.com)
- Pertimm
- Sinequa CS de Sinequa
- Zoom, d'Acetic

(Source: http://fr.wikipedia.org/wiki/Moteur_de_recherche)

2.4 AUTRES SERVICES ET OUTILS

Je présente ci-dessous différents solutions qui, même si elles ne sont pas directement en relation avec le Web sémantique ou ne permettent pas directement d'ajouter du sens à l'information, présentent néanmoins un intérêt dans le cadre de ce travail.

Dapper

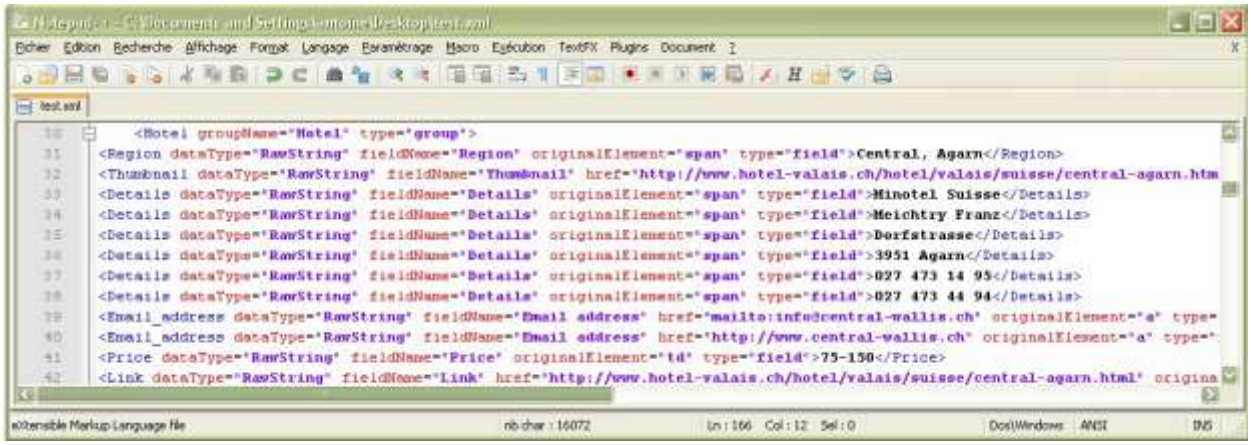
Dapper est un service Web gratuit qui permet aux utilisateurs de créer facilement un flux d'informations à partir de n'importe quel site Web.
J'ai creusé un peu cette solution pour essayer d'en extraire des données depuis le site de valaistourism.ch et hôtel-valais.ch.

Suite à un échange assez riche avec leur support, j'ai pu créer un 'Dapp' très intéressant pour hôtel-valais.

Dans un premier temps, leur browser virtuel ne supportait pas les dorp-down menu. *"POST data can generally be used by Dapper as a variable input, but Dapper does not yet support drop-down menus as variable input".*
Finalement après leur avoir soumis le problème, ils ont amélioré leur service pour prendre en compte ce cas de figure

Ci-joint une image du rendu HTML que j'ai pu générer avec leur service en ligne. J'ai choisi ici un affichage HTML pour avoir un ainsi les images. Mais il est certain que si l'on devait poursuivre en avant avec cette technique, on utiliserait un rendu XML.

Et le code source de la page qui serait facilement exploitable. Cette partie est à mettre en relation avec l'étape suivante de mon travail qui consistait à produire un échantillon de donnée depuis différentes source.



```
<Hotel groupName="Hotel" type="group">
<Region dataType="RawString" fieldName="Region" originalElement="span" type="field">Central, Agarn</Region>
<Thumbnail dataType="RawString" fieldName="Thumbnail" href="http://www.hotel-valais.ch/hotel/valais/suisse/central-agarn.htm
<Details dataType="RawString" fieldName="Details" originalElement="span" type="field">Minotel Suisse</Details>
<Details dataType="RawString" fieldName="Details" originalElement="span" type="field">Meichtry Franz</Details>
<Details dataType="RawString" fieldName="Details" originalElement="span" type="field">Dorfstrasse</Details>
<Details dataType="RawString" fieldName="Details" originalElement="span" type="field">3951 Agarn</Details>
<Details dataType="RawString" fieldName="Details" originalElement="span" type="field">027 473 14 95</Details>
<Details dataType="RawString" fieldName="Details" originalElement="span" type="field">027 473 44 94</Details>
<Email_address dataType="RawString" fieldName="Email address" href="mailto:info@central-wallis.ch" originalElement="a" type=
<Email_address dataType="RawString" fieldName="Email address" href="http://www.central-wallis.ch" originalElement="a" type="
<Price dataType="RawString" fieldName="Price" originalElement="td" type="field">75-150</Price>
<Link dataType="RawString" fieldName="Link" href="http://www.hotel-valais.ch/hotel/valais/suisse/central-agarn.html" origina
```

Pour valaistourism.ch, j'ai utilisé à peu près la même technique sans avoir à gérer les inputs variables. Il suffit dans l'affichage des résultats de modifier l'URL

Page 1 :
http://www.matterhornstate.com/fr/Valais_list-WinterVS-SkiVS.html

Page 2 :
http://www.matterhornstate.com/fr/navpage_list.cfm?listall=yes&category=WinterVS&subcat=SkiVS&start=13

Et ainsi de suite pour toutes les pages qui suivent.
(Voir: http://www.dapper.net/)

TopBraid Composer

Le seul problème avec ce fantastique outil de développement qui est écrit par la même personne qui a écrit protégé, c'est qu'il est payant et que je n'ai pas voulu passez du temps sur la version d'évaluation (30 jours) sans pouvoir l'utiliser par la suite.
(Voir: http://www.topquadrant.com/topbraid/composer/index.html)

RDF gravity

Outil en java pour visualiser les graphes RDF/OWL/Ontologies.
RDF Gravity est implémenté en utilisant la librairie JUNG Graph API et Jena semantic web toolkit. Il permet ainsi de se représenter visuellement les différents liens et concepts au sens d'un fichier RDF/OWL.
(Voir : http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html)

Xml2owl

XML2OWL est un outil qui permet comme son nom l'indique de créer un modèle OWL à partir d'un document XML. Il s'agit d'un site Web complet écrit en PHP que l'on peut télécharger et installer en local. Il va chercher un fichier XML ou XSD en entrée et va parser ce fichier pour en sortir un modèle OWL assez générique. J'ai essayé cette solution mais le résultat avec mon fichier XML me paraissait difficilement exploitable.
(Voir : http://xml2owl.sourceforge.net/)

Bots ou spiders Web

J'ai aussi testé deux robots OpenSource : JSpider et Httrack. L'objectif étant de savoir si un robot serait à même de rapatrier certaines données.

JSpider est écrit complètement en Java est permet de principalement de vérifier l'intégrité d'un site.

- Vérifiez votre site pour les erreurs (erreur interne du serveur, ...)
- Vérification des liens entrants ou sortants

- Analysez la structure de votre site (création d'un plan du site, ...)
- Téléchargement complet des sites Web

Httrack permet de télécharger un site Internet sur votre disque dur, en construisant récursivement tous les répertoires, récupérant HTML, images et fichiers du serveur vers votre ordinateur.

Ces outils permettent ainsi de copier un site en local, qui simplifie la suite de tu travail pour l'analyse des données Web. Il est intéressant de notre aussi que certains bots prennent en paramètres des fichiers de configuration, et il semble possible d'y intégrer des outils de recherche textuelle avancé comme Lucene mais néanmoins il semble assez difficile d'extraire une information ciblée sur un sujet.
(Voir : http://j-spider.sourceforge.net/) et
(http://www.httrack.com/page/2/fr/index.html)

3. PHASE D'ANALYSE DES TECHNOLOGIES

Je vais vous présenter les différents langages et technologies que j'ai étudiés dans le cadre de ce travail. Je tiens à préciser que je n'ai malheureusement pas suivi de cours de formation sur le langage XML ainsi XSLT. C'est pourquoi, je leurs accorde une grande place dans la présentation qui suit. Pour les autres langages je vais juste décrire brièvement leur fonctionnalité ainsi que certains aspects intéressants qui recèlent.

3.1 LES DIFFÉRENTS LANGAGES DU WEB SÉMANTIQUE

RDF

Le format RDF permet de définir des métas-données afin de préciser les caractéristiques d'une information. C'est un langage à la base du Web sémantique qui modélise sous forme de triplet. Chaque affirmation est décomposé en trois parties: (sujet, prédicat, objet) par exemple (tutorial.php, auteur, "Antoine").

Le **sujet** est l'URI identifiant une ressource. Le **prédicat** est une relation identifié par une URI. L'objet ou le **complément** est soit un URI correspondant à une ressource soit une valeur littérale. Chaque triple peut être considéré comme un arc et on obtient ainsi un graphique qui décrit les ressources identifiées (URI) et leurs relations.

```
<hotel:Hotel rdf:about="CH_1920_motel_des_sports">
                <hotel:hasName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Motel des
Sports</hotel:hasName>
                <hotel:hasManager
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Jean-Marc
Habersaat</hotel:hasManager>
                <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Piscine_exterieure"/>
```

```
                <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Bar"/>
                <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Situation_calme"/>
                <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Chien_accepte"/>
                <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Situation_centrale"/>
        </hotel:Hotel>
```

Un exemple de donnée RDF pour un hôtel

RDFS

RDFS est une extension basé dur RDF qui permet de créer un schéma. C'est un ensemble de primitives pour décrire des ontologies légères. Il permet ajoutant des notions de classe et de sous-classe, de propriété et sous-propriété, d'héritage et de domaine.

A noter que le langage RDF Schéma n'est qu'un des langages expérimentaux de logique actuellement mis au point. Les autres, moins avancés ou reconnus, sont DAML+OIL (une extension de XML et RDF dans un but Sémantique), swap/log.

OWL

OWL permet de définir des **ontologies complexes**. Elle regroupe les concepts de RDF et RDFS et ajoute les notions de classes équivalentes, de propriétés équivalentes, d'égalité et d'inégalité des ressources, de symétrie et de cardinalité.

Avec des ontologies précises et complètes, des ordinateurs peuvent agir comme s'ils comprenaient les informations qu'ils transmettent.

```
<owl:ObjectProperty
rdf:about="http://www.websemantique.ch/onto/hotel/hasFeature">
   <rdfs:domain rdf:resource="http://www.websemantique.ch/onto/hotel/Hotel"/>
   <rdfs:range rdf:resource="http://www.websemantique.ch/onto/hotel/Feature"/>
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
   >Définit la relation entre un hotel et une liste complète de ces
caratéristiques</rdfs:comment>
</owl:ObjectProperty>
```

Un exemple sur la portée des concepts dans OWL

OWL comprend trois sous-langages différents :

- OWL/Lite est de la version la moins expressive de OWL. Il est destiné à représenter des hiérarchies de concepts simples.
- OWL/DL est une version plus complète. Il est fondé sur la logique descriptive d'ou son nom (OWL Description Logics). Il est adapté pour faire des raisonnements, et il garantit la complétude des raisonnements et leurs décidabilités. Il s'agit, en général, de la version utilisée par les raisonneurs tels que Racer ou Pellet.
- OWL/Full est la version la plus complète, destiné aux situations ou il est important d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie.

(Voir : http://www.yoyodesign.org/doc/w3c/owl-features-20040210/)

Je reviendrai plus profondément sur le langage OWL et ses possibilités dans le chapitre protege 3.4, ainsi que sur les détails de mes créations d'ontologies dans les chapitres Les ontologies développées.

SPARQL

SPARQL est un langage de requête pour graphes RDF. Inspiré du langage SQL, il se base sur les triplets contenus dans un graphe.

Dans sa forme habituelle il utilise la clause SELECT x WHERE () qui ressemble au classique SQL. Cependant, j'ai pu constater que ces fonctionnalité vont nettement moins loin que SQL. Ainsi, sa syntaxe se limite à une dizaine de mot clé. Un des grands intérêts de SPARQL est qu'il permet de construire dynamiquement des données avec le mot clé CONSTRUCT qui pourrait s'apparenter à un INSERT.
A noter aussi que l'on peut filtrer des requêtes au moyen des clauses FILTER et OPTIONAL, et que la fonction BOUND permet de savoir dans quelle mesure une URI est lié à une variable précédemment défini.

```
PREFIX tdprofile: <http://www.websemantique/TDprofile#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * WHERE{
?personne foaf:accountName ?name.
?personne tdprofile:likeToHaveHotel ?cara.
}
```

Exemple simple de requête SPARQL

Explication : en se basant sur les ontologies TDprofile et FOAF, trouve-moi les personnes qui ont un nom de login et qui aimeraient avoir certaines caractéristiques particulières dans un hôtel.

```
PREFIX hotel: <http://www.websemantique.ch/onto/hotel/>
PREFIX tdprofile: <http://www.websemantique/TDprofile#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT * WHERE{
?hotel a hotel:Hotel.
?hotel hotel:hasFeature <http://www.websemantique.ch/onto/hotel/Animaux_autorises>.
<http://www.websemantique.ch/onto/hotel/Situation_calme>.
OPTIONAL{?personne tdprofile:hasBeenHotel ?hotel .
?personne foaf:accountName "linda"} .
FILTER(!BOUND(?personne))
}
```

Exemple d'une requête complexe

Explication : en se basant sur les ontologies hotel, tdprofile, foaf, et rdf(pas obligatoire ici), trouve-moi les hôtels qui sont défini par le concept 'hotel' et qui ont comme caractéristiques d'accepter les animaux et d'avoir un situation calme.
Trouve-moi aussi en option les personnes qui ont été dans ces hôtels et qui un comme nom de compte 'Linda '. De plus je veux filtrer la sélection pour ne faire ressortir que les hôtels dans lequel ladite personne n'a pas encore été.

Cette requête me sera très utile dans mon démonstrateur pour présenter quelques suggestions vis-à-vis d'une personne.

XML

XML est un acronyme pour "eXtensible Markup Language" soit langage de balisage extensible. Il s'agit d'un standard défini par le W3C, qui permet de créer des structurations de données spécialisés et personnalisés. Le XML est ainsi ce qu'on appelle un "méta langage", soit un langage qui définit une information, mais il est aussi question du format de fichiers.
Les langages basés sur XML permettent de traiter et de communiquer toutes sortes de données

```
<object id="TDS00020010000148892" dataClassName="IrsSwissBudgetServiceProvider">
        <attribute name="CompanyName1" id="WBX00020010000000026"
        dataType="0" comment="Name of the company (first line)">
                        <value>Motel des Sports</value>
        </attribute>
        <features>
        …
        </features>
</object>
```

Exemple d'une partie de mon fichier XML

XSD

XSD est un standard approuvé par le W3C permettant de décrire la structure d'un document XML. Il définit de façon structurée le type de contenu et la syntaxe d'un document XML. Il est également utilisé pour valider un document XML, c'est à dire vérifier si le document XML respecte les règles décrites dans le document XSD.

D'une manière générale, un document définit les éléments et les attributs constituant un fichier XML. Un élément peut être vu comme une classe, elle définit alors une structure, et un attribut peut être envisagé comme un complément d'information rattaché à un élément.

```
<xs:element name="object">
  <xs:complexType>
   <xs:sequence>
      <xs:element ref="features"/>
      <xs:element ref="multimedia"/>
      <xs:element ref="attribute"/>
      <xs:element ref="descriptions"/>
   </xs:sequence>
   <xs:attribute name="dataClassID" use="required" type="xs:NCName"/>
   <xs:attribute name="dataClassName" use="required" type="xs:NCName"/>
  </xs:complexType>
 </xs:element>
```

Exemple d'une partie de mon fichier XSD

Je reviendrai dans le chapitre suivant sur l'utilité des ces fichiers. De même, je consacre un chapitre au langage XSLT et l'utilisation que j'en ai eue pour transformer mon fichier XML en données RDF.

4. PHASE DE COLLECTE D'INFORMATIONS

4.1 LES SOURCES D'INFORMATIONS

Pour réaliser mon démonstrateur sur le domaine appliqué du tourisme en Valais, je devais tout d'abord commencer par collecter un échantillon de données représentatives. Si je reprends la partie gauche du graphique concernant l'architecture général du système *(voir architecture du système)*, on voit que les sources de données possibles se présentent sous 3 formes :

- **Une base de données**

  C'est le meilleur scénario possible, car ainsi on a un accès direct à l'ensemble des données. Il existe actuellement des solutions permettent d'exporter une base de données relationnelle directement vers le format RDF. Pour réaliser une telle manipulation, je citerai juste le langage D2R.
  D2R est un langage déclaratif pour codifier les relations entre une base de données relationnelles et une ontologie OWL/RDFS. On peut ainsi grâce à ce 'mapping' exporter toutes nos données dans un format RDF valide.

- **Internet**

  Cette solution offre des nombreux avantages, notamment la richesse et la multitude informations mais elles posent aussi de nombreux problèmes, dont la fiabilité des données. De plus, il n'existe actuellement aucun (à ma connaissance) outil permettant de parcourir le Web et de collecter des informations relatives à un sujet bien précis et ensuite, de structurer et de consolider l'ensemble de ces informations.
  Pour réaliser un tel outil, il faudra avoir accès à des outils d'intelligence artificielle pour analyser le contenu des articles du Web. Il ne s'agit pas uniquement de

créer un 'bot' basé sur un algorithme qui calcule les occurrences d'un mot et tri les pages en fonction, mais bien d'un outil capable d'analyser le contenu d'un article et de comprendre la nature de ce qui décrit.

En revanche, et c'est le choix que nous avons fait, nous pouvons réaliser un 'parseur' qui se focalise uniquement sur un site et analyse ses données en fonction de la structure des balises.
Evidemment, nous analysons à l'avance la structure des pages Web fournit par le site et ainsi on peut se basé sur des balises types.

Exemple :
```
<div class="topLists">
<mon tableau de donnée>
</div>
```

On peut aussi se servir d'outil de 'mapping' tels Dapper. Cet outil permet déjà de filtrer une partie du contenu des pages HTML. De plus, il comprend plus-ou-moins la structure d'un tableau HTML et semble être en mesure de faire des itérations dessus.
(Voir explications dans Etat de l'art)

Note : Le site de Dapper propose depuis peu, en version beta, un service de sémantisation pour les sites codés en PHP. Les informations RDF ne sont alors plus stockées dans la page mais renvoyer par Dapper au moment où un moteur de recherche procède à une requête sur votre site. Cette idée est assez intéressante sur l'aspect de la décentralisation des données RDF mais il faut bien être conscient qu'on place alors une grande confiance dans les informations gérer par Dapper.
(Voire : http://www.dapper.net/semantify/)

Remarque: Je pensais dans un premier temps écrire un programme (parseur) en Java qui collecterai ces données. Mais finalement, cet aspect a été réalisé par Fabian sur le site de TripAdvisor.

- **Données XML**

  Cette solution représente un niveau intermédiaire entre des données puisées sur internet et une exportation typée depuis une base de données.
  Explications : pour ma part et grâce au partenariat d'Anne avec Valais tourisme, nous avons obtenu un fichier XML assez volumineux (38Mo) qui correspond au dump d'une partie de leur base de données. Aussi les données correspondent exactement aux champs de la base de données.

  Mais XML c'est aussi le meilleur moyen pour un designer Web de fournir et d'échanger de l'information sur le Web. C'est pourquoi j'ai écrit un script en XSLT pour transcrire ces données au format RDF.

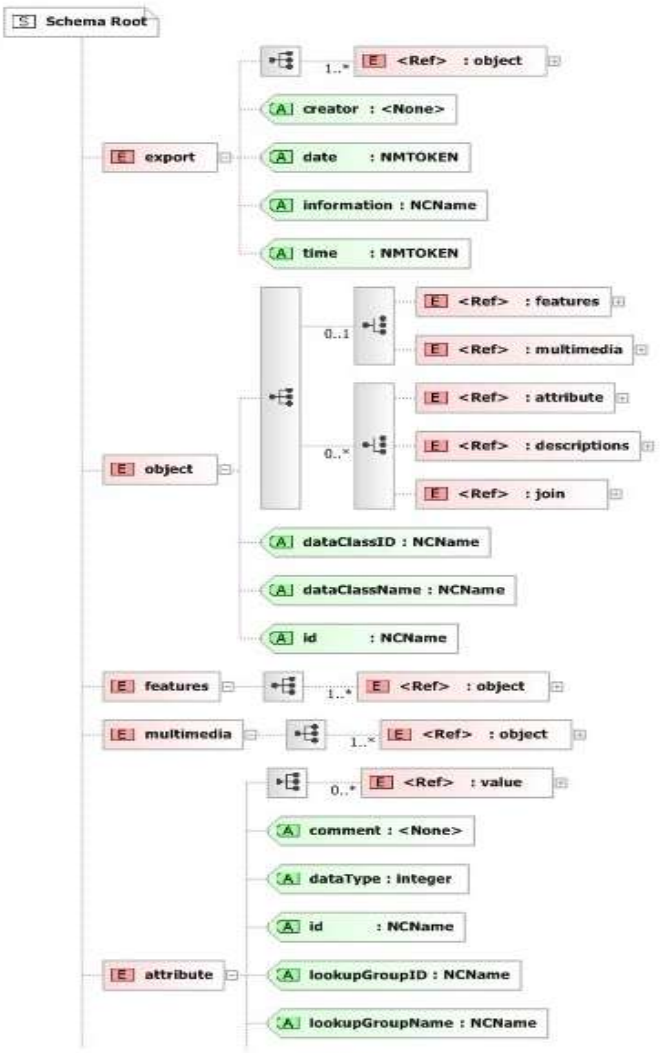
  Bien sur, il existe des solutions comme GRRDL qui font réaliser cette transformation, d'ailleurs GRRDL n'est rien de plus qu'un moyen d'automatiser la génération de fichiers XSLT. Mais j'ai préférer apprendre les concepts liés au XML, XSD, XSLT et ainsi maitriser de nouveaux langages. J'explique dans le chapitre suivant les différentes étapes pour transformer ces données.

  Je tiens aussi à préciser que les données Tomas doivent restées confidentielles.

4.2 PROBLÈMES LIÉS AU FICHIER XML

Bien que dans la théorie, les concepts d'XML et XSD soient assez simples, j'ai quand même rencontré quelques problèmes avec le dump fournit par Valaistourisme.ch.

Premièrement, la 'mauvaise' structure du document XML. Comme vous pouvez le voir sur le graphique ci-dessous qui représente graphiquement la structure XSD. Le créateur du fichier à nommé certains de ces éléments *attribute* et il les a complété avec d'autres attributs. Je pense qu'il s'agit ici d'un outil d'exportation automatique qui doit reprendre certain nom de table… De plus, il n'est pas aisé de naviguer dans cette structure relativement complexe, surtout pour un néophyte comme moi.



Une représention graphique du schéma XSD

Deuxièmement, le fichier original faisait 38 MB, ce qui est largement suffisant pour faire planter la plupart des éditeurs XML. La raison d'une telle taille pour un export de 157 hôtels, la balise CDATA.
Cette balise (Character Data soit données de caractère) figure dans le fichier XML sous la forme ![CDATA[était utilisé pour insérer des images.

Explication : la section CDATA n'est pas analysée par les navigateurs : elle sert donc de plus ou plus en plus souvent de fourre-tout pour certains développeurs qui ne prennent pas soin de valider leur XML et préfère simplement cette section si pratique. Elle est donc utilisée parfois à tort ou à travers afin d'éviter qu'un caractère inattendu ne casse la validité du XML, et donc ne le rende illisible.

4.3 XSLT

Dans les chapitres, je présente une partie importante de mon travail qui à consister à transformer mes données en RDF avec un script XSLT.
Pour comprendre la portée du langage XSLT, il faut d'abord s'intéresser à Xpath.

Xpath est un langage pour identifier, sélectionner et manipuler les nœuds dans une structure XML. On peut le comparer aux différentes méthodes qui permettent de se déplacer sur un arbre. Il se base sur le chemin de localisation qui dépend :

- D'un **axe** définissant le sens de la relation entre le nœud courant et le jeu de nœuds à localiser
  - child : contient les enfants directs du nœud contextuel.
  - descendant : contient les descendants du nœud contextuel. Un descendant peut être un petits-enfants...
  - parent : contient le parent du nœud contextuel.
  - ancestor : contient les ancêtres du nœud contextuel. Cela comprend son père, le père de son père... Cet axe contient toujours le nœud racine.

- 0 à n **prédicats** qui permettent d'affiner la recherche sur le jeu de nœuds à récupérer suivants certaines conditions entre crochets
  - test sur le contenu d'un élément, sur son nom, ses attributs ou sa position dans l'arbre.
  - expressions numériques, d'égalités, relationnelles et booléennes supportées

- De plus Xpath dispose d'une petite bibliothèque de fonctions prédéfinies pour :
  - chaînes de caractères (string),
  - nombres (number),
  - booléens (boolean),
  - nœuds (node)

XSLT est un langage de transformation procédurale, il permet notamment de filtrer des données, de les trier, les restructurer. Voici les principales règles susceptibles d'être appliqué à un nœud spécifique.

- xsl:template : Définir les règles de transformations à réaliser sur un ensemble de nœuds
- xsl:value-of : Sélectionner les éléments XML à afficher
- xsl:copy-of : Copie le nœud en entier
- xsl:for-each : Sélectionner toutes les valeurs des éléments XML à afficher
- xsl:sort : Permet de trier les données
- xsl:if : Test simple de type booléen
- xsl:text : Inclure du texte littéral
- xsl:choose, xsl:when, xsl:otherwise : Choix Conditionnels
- xsl:element et xsl:attribute : Permet d'insérer un élément ou un attribut
- xsl:variable et xsl:param et xsl:with-param : Permet de déclarer des variable locales
- Pour accéder à un attribut on utile la syntaxe '@monattribut'

Comme touts langages modernes, il est possible d'appeler un script XSLT depuis un autre langage via une librairie appropriée.

Ci-dessous un exemple en PHP et un autre en JavaScript / ActiveX

```php
<?php
// Nouvelle instance
$xslt = new XSLTProcessor();

// Chargement du fichier XML
$xml = new domDocument();
$xml -> load('hotel_test.xml');

// Chargement du fichier XSL
$xsl = new domDocument();
$xsl -> load('rdf.xsl');

// Import de la feuille XSL
$xslt -> importStylesheet($xsl);

// Transformation et affichage du résultat
echo $xslt -> transformToXml($xml);
?>
```

```javascript
<script type="text/javascript">

// Chargement du fichier XML
var xml = new
ActiveXObject("Microsoft.XMLDOM")
xml.async = false
xml.load("stc_export_formated_min.xml")

// Chargement du fichier XSL
var xsl = new
ActiveXObject("Microsoft.XMLDOM")
 xsl.async = false
 xsl.load("feature.xsl")

// Transformation en Html
document.write(xml.transformNode(xsl))
</script>
```

4.4 MON SCRIPT XSLT

Dans le chapitre suivant, j'explique en détail la création du script XSLT par rapport à mon fichier XML.
Pour créer mes données RDF depuis un script XSLT, je dois d'abord mettre l'entête:

- <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  Contient des éléments de mise en forme et nous permet d'avoir accès à des fonctions évoluées d'XSLT.

- <xsl:output>
  Permet de spécifier des options concernant l'arbre de sortie

- <xsl:template>
  Définit un modèle à appliquer à un nœud

- <xml:base>
  Spécifier le 'base' auquel seront concaténés les identifiants pour créer des URI complètes

- <xmlns :named_space>
  Définit les named space auquel les données font référence

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" indent="yes" encoding="UTF-8"/>
	<xsl:template match="/">
		<rdf:RDF xml:base="http://www.websemantique.ch/onto/hotel/"
			xmlns="http://www.websemantique.ch/onto/hotel/"
			xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
			xmlns:genoum="http://www.websemantique.ch/onto/genoum/"
			xmlns:memo="http://www.memoria-mea.ch/memonto.owl#"
			xmlns:foaf="http://xmlns.com/foaf/0.1/"
			xmlns:owl="http://www.w3.org/2002/07/owl#"
			xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
			xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
			xmlns:hotel="http://www.websemantique.ch/onto/hotel/">
```

L'en-tête du script

Ensuite, je fais une boucle pour chaque enregistrement:

- <hotel:Hotel rdf:about="…">
  Je crée un élément xml pour chaque ressource: une instance du concept Hotel de l'ontologie hôtel (le named graph hotel valable dans ce fichier et spécifié ci-dessus dans l'entête).

Donc <hotel:Hotel rdf:about="…"> identifie de manière unique la ressource. Le < nxml:base> sera rajouté par OntoMea pour créer une URI complète lors de l'import.
Par exemple :
rdf:about="http://www.websemantique.ch/onto/hotel/CH_7550_Gasthaus_Mayor

De plus, vous pouvez voir que j'ai du utiliser la notion de variable XSLT pour concaténer deux valeurs. Bien que peu permissif, le langage XSLT possède quand même quelques attributs propre à un langage de programmation comme les boucles, les structure de contrôle et les variables.

- <hotel:hasnom>, <hotel:hasFax>, <hotel:hasTelephone>, <memo:locatedIn:>
  Des data property, comme par exemple le nom
  <hotel:hasNom> Motel des Sports</hotel:hasNom>
  Et des object property comme ici les liens GeoNamess avec un test 'choose'
  <memo:locatedIn rdf:resource="http://sws.geonames.org/2658434/"/>

- <hotel:hasFeature>
  Je boucle sur toutes les caractéristiques d'un hôtel et je les teste une-à-une.
  Pour ce test, j'effectue une comparaison de string.
  <hotel:hasFeature
  rdf:resource="http://www.websemantique.ch/onto/hotel/Bar"/>

```
<xsl:for-each select="export/object">
        <xsl:variable name="a1" select="attribute[@name='ZipCode']/value"/>
        <xsl:variable name="a2" select="attribute[@name='CompanyName1']/value"/>
        <hotel:Hotel rdf:about='CH_{$a1}_{translate($a2, " ", "_")}'>

                <hotel:hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                        <xsl:value-of select="attribute[@name='CompanyName1']/value"/>
                </hotel:hasName>

                <hotel:hasFax rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                        <xsl:value-of select="attribute[@name='CompanyFax']/value"/>
                </hotel:hasFax>
```

```
                <hotel:hasTelephone
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                        <xsl:value-of select="attribute[@name='CompanyPhone']/value"/>
                </hotel:hasTelephone>
                <!-- suive les autres tests de caractéristiques... -->

                <xsl:choose>
                        <xsl:when test="join/object/attribute/value[4] = 'Arolla'">
                                <memo:locatedIn
rdf:resource="http://sws.geonames.org/2661713/"/>
                        </xsl:when>
                        <!-- suive les autres tests de villes... -->

                        <xsl:otherwise>
                                <memo:locatedIn rdf:resource="Pas_en_valais"/>
                        </xsl:otherwise>
                </xsl:choose>

                <xsl:if test="starts-with(attribute[@name='Street']/value, '')">
                        <hotel:hasStreet
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                                <xsl:value-of select="attribute[@name='Street']/value"/>
                        </hotel:hasStreet>
                </xsl:if>

                <xsl:for-each select="features/object">
                        <xsl:choose>
                                <xsl:when test="attribute/value = 'Piscine ext?rieure'">
                                        <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Piscine_exterieure"/>
                                </xsl:when>
                                <xsl:when test="attribute/value = 'Bar'">
                                        <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Bar"/>
                                </xsl:when>
                                <xsl:when test="attribute/value = 'Chambres calmes'">
                                        <hotel:hasFeature
rdf:resource="http://www.websemantique.ch/onto/hotel/Chambres_calmes"/>
```

```
                                </xsl:when>
                            </xsl:choose>
                        </xsl:for-each>
                    </hotel:Hotel>
</xsl:for-each>
```

Le 'coeur' du script

Après la définition de tous les hôtels, je donne encore une information sur les ressources utilisées, comme les lieux GeoNamess. Car ici, on fait référence à http://sws.geonames.org/2658434/ mais on ne sait pas ce que c'est. Donc pour faciliter le traitement par la suite, j'ajoute l'information <rdf:type>.

Ainsi, le triple store saura de quoi il s'agit, et pourra par exemple aller sur GeoNamess chercher le <rdf :about> concernant ce lieu.
OntoMea effectuera pour nous cette opération, en appelant un Web service sur le site de GéoName. *(Voir : http://www.geonames.org/export/ws-overview.html)*

```
<rdf:Description rdf:about="http://sws.geonames.org/2658434/">
        <rdf:type rdf:resource="http://www.geonames.org/ontology#Feature"/>
</rdf:Description>
```

Description d'un ID geoname

Vous pourrez remarquez en observant le fichier XML qu'il y a un gros problème d'accents du au dump de la base de donnée. Bien que ce problème ne soit pas insurmontable, il pose néanmoins certains problèmes et permet aussi de voir rapidement les limitations du langage XSLT. J'ai trouvé sur internet une fonction propre à XSLT pour transformer les majuscules en minuscules, mais elle ne fonctionne pas dans mon script et ne semble d'ailleurs pas implémenté dans Oxygen XML Editor!

Finalement, il ne reste plus qu'à fermer toutes les balises d'en-tête

```
</rdf:RDF>
</xsl:template>
</xsl:stylesheet>
```

Fermeture des balises

4.5 OUTILS ET RESULTATS

Pour réaliser ces différents scripts, j'ai testé et utilisé différents software dont :

XTrans : un petit éditeur gratuit et très utile pour le langage XSLT. Il possède une simple fonction d'insertion avec une liste exhaustive des balises et des fonctions XSL. De plus, même si cette fonction est triviale, il permet de mettre en forme des scripts XML et surtout il supportait mon fichier de 38 MB !

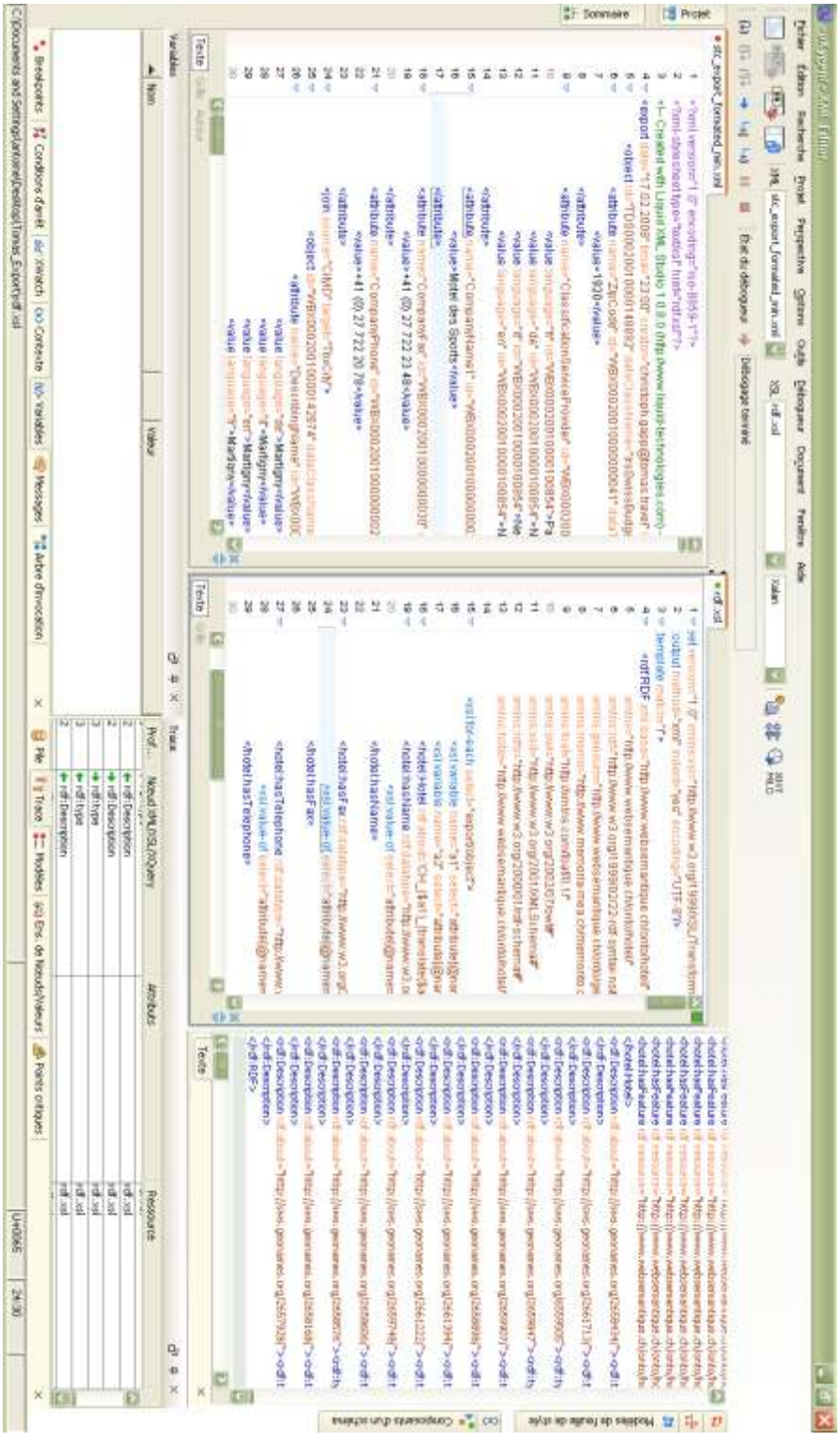
Liquid XML Studio (freeware) : éditeur complet supportant le langage XML, XSLT, XSD ainsi que WSDL. Cet éditeur propose notamment des fonctions des générations de code automatique à partir d'un schéma XSD ainsi qu'un service pour appeler et modifier des requêtes sur un Web Service.

Oxygen XML Editor 9.2 (en version d'évolution de 30 jours) : ce fantastique outil de développement est riche, très riche. J'avoue ne pas maîtriser la moitié des outils qu'il fournit et des technologies qu'il regroupe. Cependant, j'ai principalement utilisé ce programme pour le développement de mon script car il dispose d'une 'intelligent Content Completion' soit une forme d'intelligence pour la génération de code via des suggestions de bout de code adéquat.

De plus, il fournit une vue de débogage pour les scripts XSLT, avec notamment les possibilités classiques d'un tels outil comme, savoir la valeur d'un variable et rentrer et tester des conditions.

Ci-dessous un vue en mode débogage de :

- Mon fichier source XML -> 1er écran
- Mon script XSLT -> 2ème écran.
- Sa sortie au format RDF -> 3ème écran.
- Des informations sur l'état des variables en bas de page.



5. PHASE DE MODÉLISATION DES DONNÉES

En philosophie, selon Aristote une ontologie désigne un « *discours sur l'être en tant qu'être* », c'est à dire « *l'étude des propriétés générales de ce qui existe* ».

Dans le cadre du Web sémantique, une ontologie désigne :

- Un ensemble structuré de savoirs dans un domaine particulier de la connaissance ou
- Un ensemble de concepts organisés en **graphe** dont les relations peuvent être sémantiques ou de composition et d'héritage.

Une ontologie permet donc d'organiser des informations ou concepts pour construire de la connaissance.

5.1 PROTEGE 3.4

Protégé est un logiciel gratuit et open-source qui fournit une suite d'outils pour construire des ontologies. À sa base, Protégé met en œuvre un ensemble de connaissances et des mesures en vue de soutenir la création, la visualisation et la manipulation de ces ontologies. Protégé fournit 2 outils :

- Protégé-Frames fournit un éditeur graphique à part entière et un serveur de connaissances pour aider les utilisateurs à la construction et le stockage d'ontologies de domaine ainsi que la personnalisation de la saisie des données via des formulaires.

- Protégé-OWL est une extension de Protégé qui supporte le langage OWL. Approuvé par le World Wide Web Consortium (W3C) afin de promouvoir la vision

du Web sémantique OWL est le logiciel le plus élaboré dans la norme ontologie langues. Il permet de décrire précisément des classes, des propriétés et des instances. Il offre également la possibilité d'être interroger par un raisonneur externe via une interface DIG afin de contrôler l'intégrité d'un schéma et de créer des inférences. *(Voir : http://protege.stanford.edu/)*

Ce chapitre a pour but d'expliquer les principes de base d'une ontologie à travers le logiciel Protégé. Cette outil Open Source permet, grâce à son l'interface graphique de créer simplement des ontologies. Elle reprend en grande partie les chapitres ainsi que certaines images du fameux tutorial sur les pizzas.

Cette présentation regroupe des notions essentielles quant à la création d'une ontologie. De plus, je mettrai en avant certains aspects qui ont important lors de la création de mes ontologies *(Voir : les ontologies développées).*

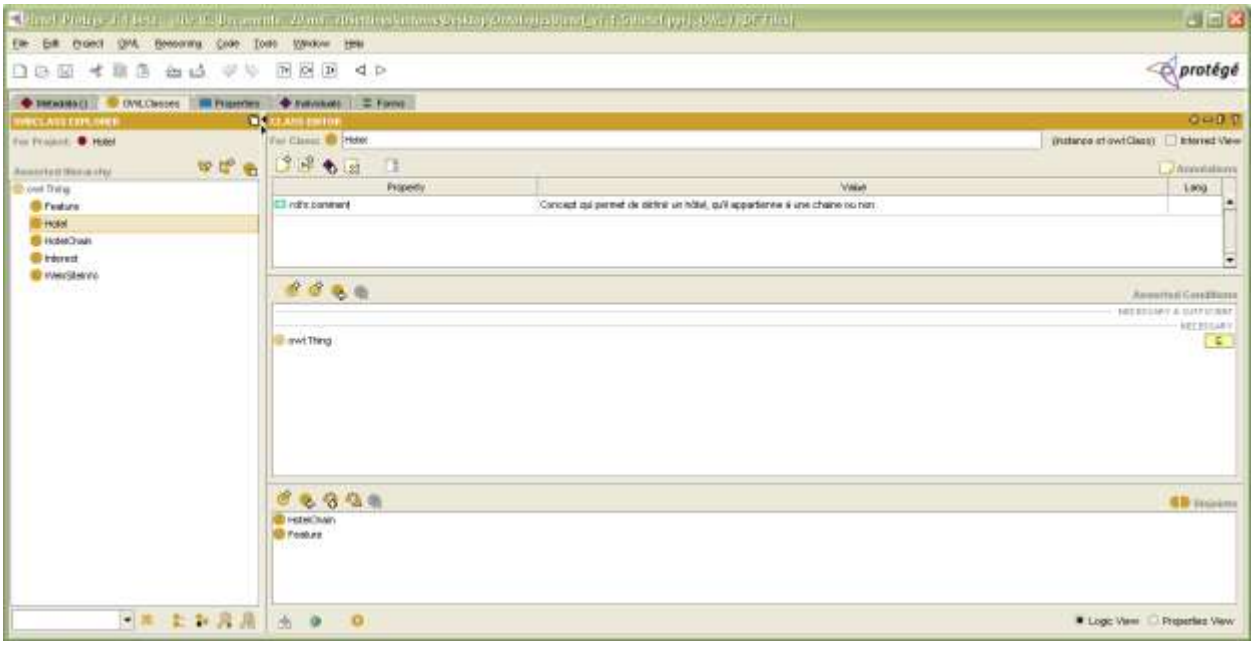
Classes

La première étape, lors de la création d'une ontologie OWL, est d'organiser les différents éléments nécessaires en classes.
La hiérarchie des classes est extrêmement importante les classes filles hériteront des propriétés des classes parentes.
Pour décrire cette hiérarchie, les propriétés RDFS rdfs:subClassOf et rdf:type sont utilisées. Il suffira donc d'un raisonneur RDFS pour effectuer des déductions sur ces propriétés.

RDFS permet également de définir certaines informations de descriptions sur une classe:

- rdfs:label indique le nom de la classe
- rdfs:comment permet d'ajouter certains commentaires notamment la description

Vous pourrez remarquer sur mes ontologies que j'ai systématiquement utilisé un label pour indiquer une petite explication sur une classe ou une propriété.

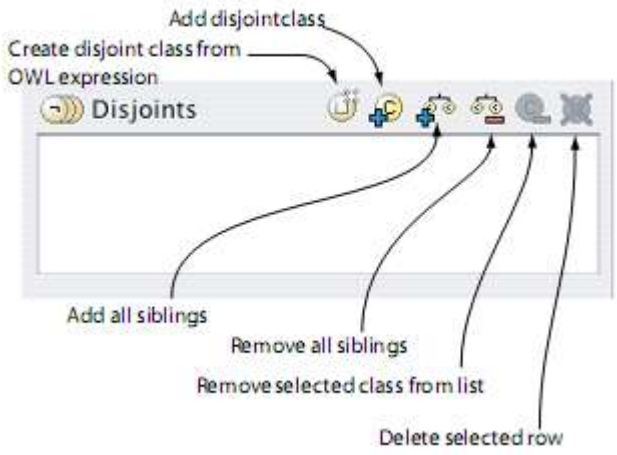


Printscreen de la classe Hotel

Classes disjointes

La méthode owl:disjointWith définit qu'une instance ne peut pas appartenir à deux classes disjointes. Il s'agit ici d'une propriété OWL. Pour pouvoir créer des inférences à partir d'un telle modèle, il faut utiliser un raisonneur capable d'interpréter le langage OWL.

Dans Protégé, la création de classes disjointes peut être automatisée. Il est possible de définir une classe fille comme disjointe de toutes les autres classes filles de la même classe parente.

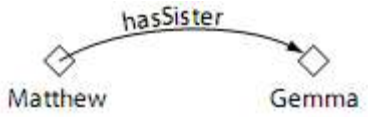


Cette propriété permet de mettre en évidence une autre fonctionnalité intéressante du raisonneur. Si par exemple, une classe définie a comme restriction d'appartenir à deux classes disjointes simultanément, le raisonneur permet de ressortir cette incohérence et définit le schéma comme non-valide.

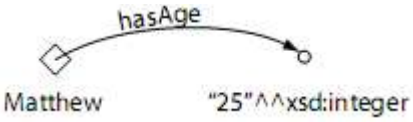
Les propriétés OWL

Une propriété représente une relation entre deux entités owl:ObjectProprety. Il existe trois types de propriétés. L'interface de gestion des propriétés OWL est similaire à celle de la gestion des classes. Il est également possible d'utiliser l'héritage en créant une hiérarchie grâce la propriété rdfs:subPropertyOf.

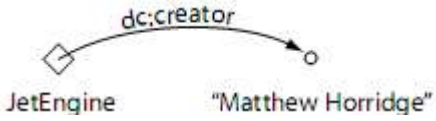
- Objects properties : définit la relation qui existe entre deux entités



- Datatype properties : définit la relation entre une entité et une valeur littérale (XML schema datatype)



- Annotations properties : permet d'ajouter une information à l'information (metadata)

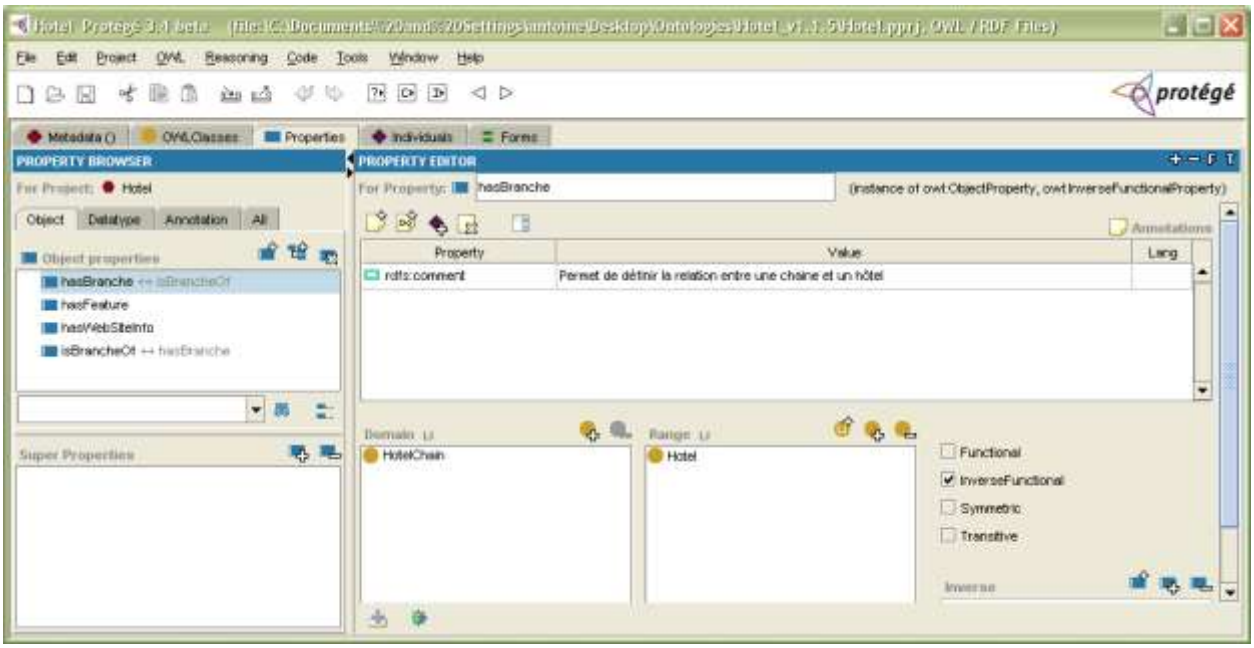


De plus, chaque propriété peut avoir les caractéristiques suivantes :

La propriété InverseOf

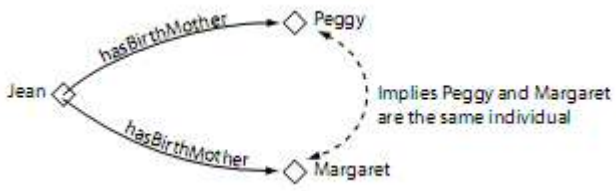
Cette propriété owl:inverseOf permet de définir que si une entité A à une relation x envers une autre entité B (A x B) alors il existe une propriété inverse qui définit la relation de B vers A (B y A). Dans Protégé, toutes ces fonctionnalités sont accessibles directement depuis l'interface dédiée aux propriétés.

Remarque : J'ai utilisé cette propriété dans l'ontologie hôtel pour déterminer que si un hôtel appartenait à une chaine d'hôtel, alors cette chaine possédait l'hôtel en question.



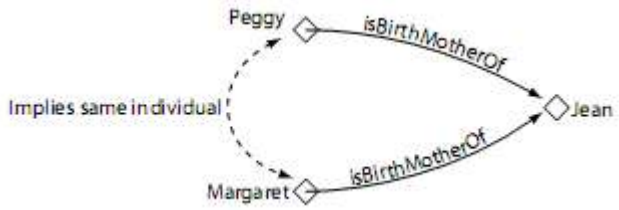
La propriété Functional

owl:FunctionalPropriety définit que l'ensemble des objets liés à un sujet donné par cette propriété font référence à un objet unique.



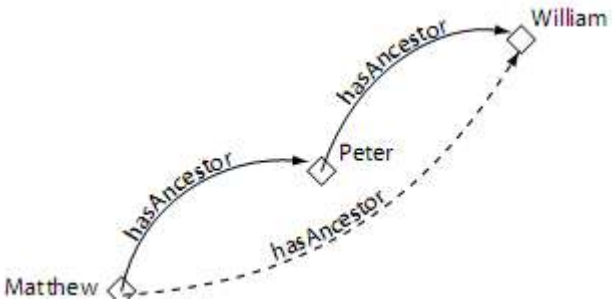
La propriété InverseFunctional

owl:InverseFunctionalPropriety définit que l'ensemble des sujets liés à un objet par cette propriété font référence à un sujet unique.



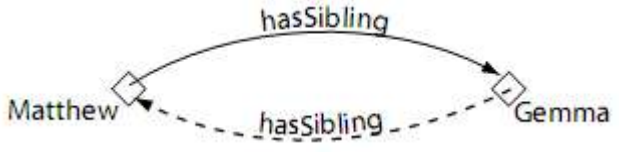
La propriété Transitive

owl:TransitivePropriety définit que si A -> B et B -> C alors A -> C



La propriété Symmetric

owl:SymmetricProprety définit que si un A est lié à un B par la propriété x alors B est lié à A par la propriété x.

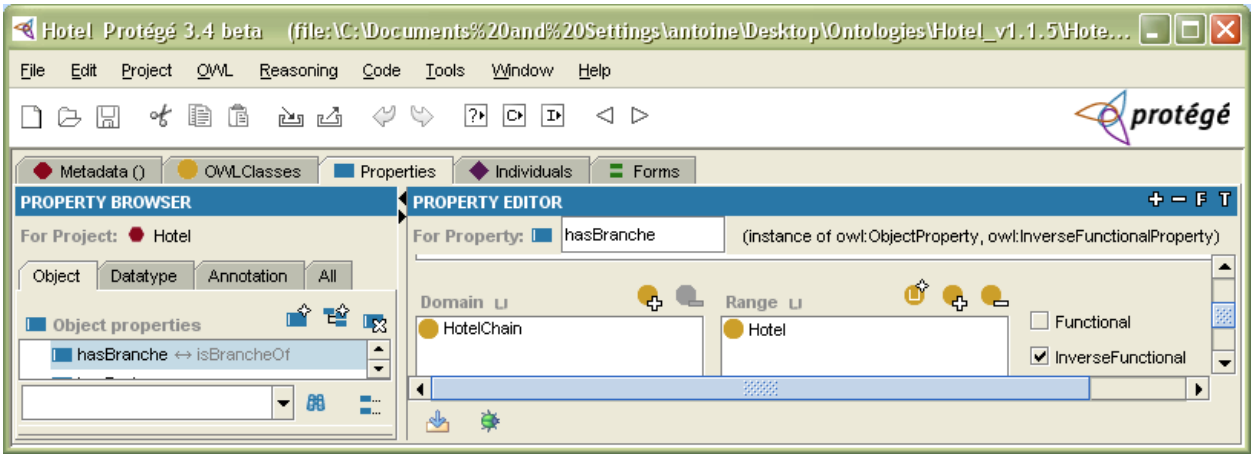


Evidemment, la création d'inférences basées sur ces propriétés et ces classes nécessite l'utilisation d'un raisonneur OWL.

Domain et range

Ils permettent de définir la portée des deux entités rattachées par une propriété.
Le domaine domain permet de déterminer de quel type de classes le sujet d'une propriété peut être.
La portée range détermine de quel type de classes l'objet doit appartenir.

Remarque : j'ai utilisé cette restriction plusieurs fois dans mes ontologies. Ci-dessous un exemple où il s'agit du domaine et de la portée de la propriété hasBranche. Elle s'applique à un sujet du type *HotelChain* et a un objet du type *Hotel*. Si cette propriété n'est pas respectée, le raisonneur indique l'incohérence.



Les restrictions

Dans le langage OWL, les propriétés peuvent être utilisées afin de créer des restrictions. Les restrictions permettent de définir si une instance appartient à une classe donnée. En OWL, il existe trois types de restrictions :

- Les restrictions de quantificateurs
- Les restrictions de cardinalité
- Les restrictions de valeur

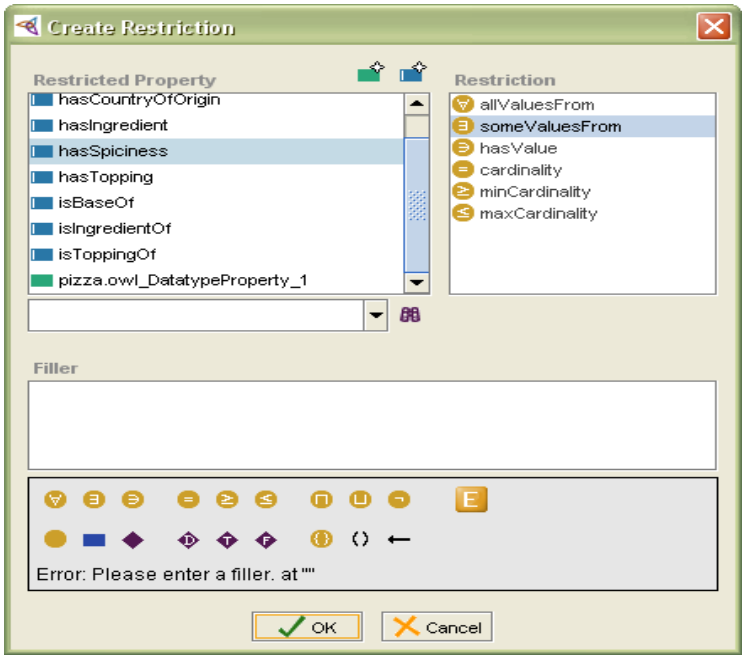
Les **restrictions de quantificateurs** s'appliquent à une classe. On peut utiliser les deux propriétés suivantes :

- quantificateur existentiel owl:someValuesFrom qui signifie «il existe au moins un ».
  Par exemple, pour dire qu'une *Pizza* doit posséder une *PizzaBase*, on utilise la restriction « hasBase owl:someValuesFrom PizzaBase »

- le quantificateur universel owl:allValuesFrom signifie est « pour toutes les valeurs ».
  La restriction « hasTopping owl:allValuesFrom VegetarianTopping » signifie que toutes les valeurs de *hasTopping* sont du type *VegetarianTopping* .

Les **restrictions de cardinalité** owl:minCardinality, owl:maxCardinality et owl:cardinality permettent de définir la cardinalité d'une propriété. Protégé-OWL utilise les symboles mathématiques <, > et = pour représenter ces rapports.

Les **restrictions de valeur** précisent la valeur que doit être adopté par une propriété. Elles sont représentées par la propriété owl:hasValue.

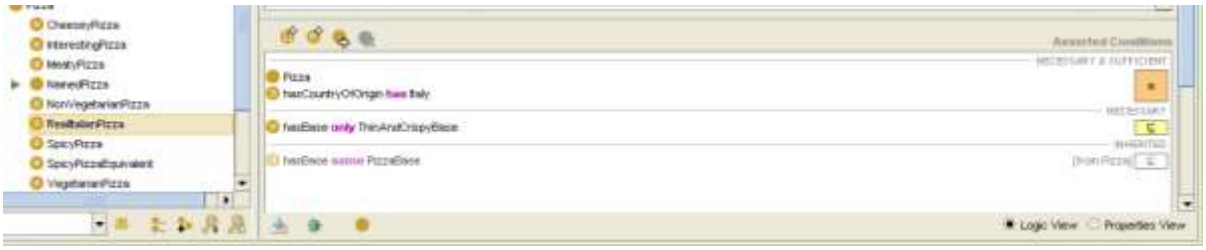


Je n'ai utilisé ces restrictions sur mes ontologies, car leur rôle est vraiment de restreindre les valeurs acceptables. Dans mon travail au contraire, nous avons cherché Fabian et moi à créer des ontologies le plus générique possible, donc sans restrictions.

Catégories de restrictions

Les restrictions peuvent êtres classées dans trois catégories:

- Les restrictions héritées des classes parentes.
- Les restrictions nécessaires, qui déterminent que tous les objets appartenant à une classe donnée possèdent ces propriétés.
- Les restrictions nécessaires et suffisantes permettent d'affirmer qu'un objet correspondant aux restrictions définies appartient forcément à la classe.



L'interface de Progété répertorie les différentes restrictions par types et catégories. Elle offre également d'un éditeur graphique de restrictions.

Il est intéressant de notre aussi que Protégé permet d'importer grâce à owl:import une ontologie dans une autre et ainsi de disposer des tous les classes et propriétés qu'elle définisse.

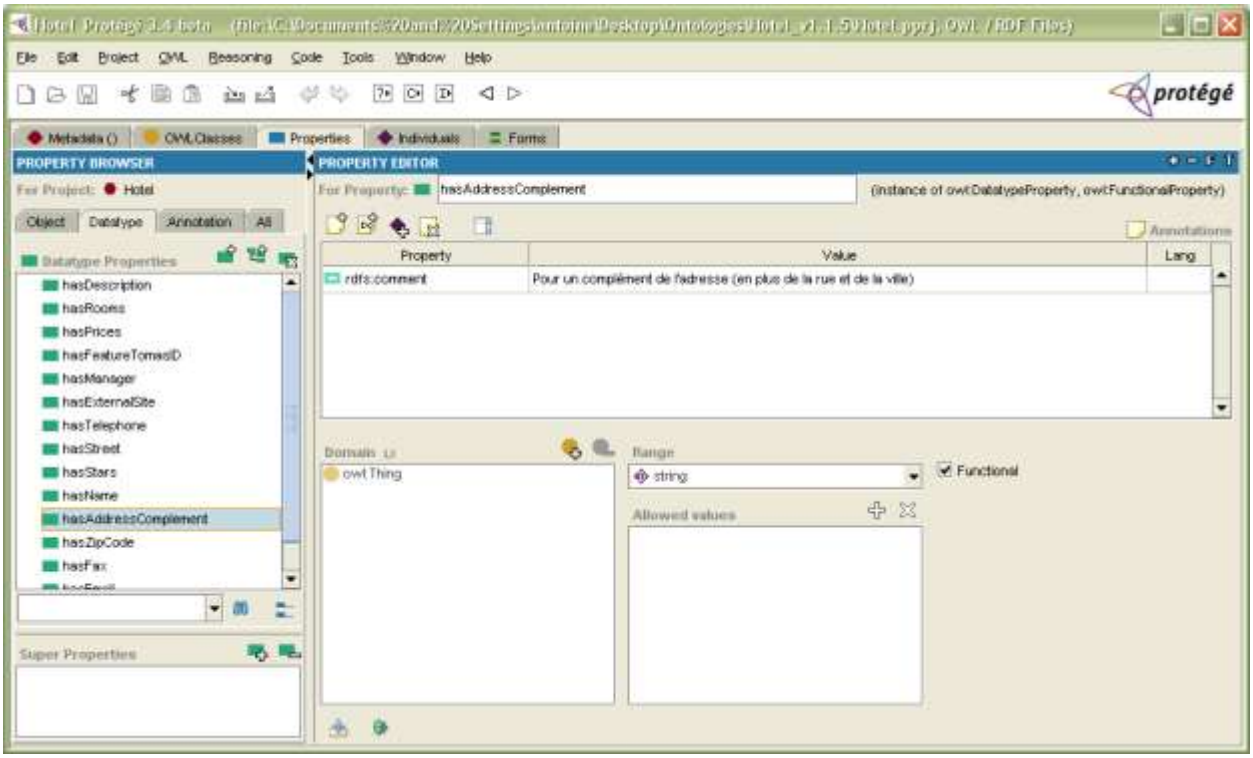
5.2 LES ONTOLOGIES DEVELOPPÉES

Dans ce chapitre, je vais brièvement présenter les ontologies que j'ai développées avec Fabian et revenir sur certains points qui me semblent intéressants.

Hotel.owl

Cette ontologies a pour bu de donner la structure nécessaire à mon exportation de données ainsi que celle de Fabian. Elle définit notamment les concepts d'Hôtel, de chaine d'hôtel, de caractéristique, de centre d'intérêt. Elle reprend évidemment la majorité des informations disponibles dans le fichier XML.
De plus elle définit un concept particulier, lié à la source de donnée que Fabian à utilisé, le site de Tripadvisor, nommé la classe *WebSiteInfo*.

Il s'agit d'un concept qui permet de définir une source de donnée (ex: TripAdvisor) pour certaines informations. On peut ainsi indiquer que surle site de TripAdvisor, il est noté que l'hôtel a 40 chambres, est un hôtel 3 étoiles... Ce classe permet donc de rattacher tous les renseignements d'un hôtel à un site, et pas directement à l'hôtel lui-même, ce qui nous permettrait de faire ensuite des contrôles de la mise à jour des informations. Pour mes données XML, je n'ai pas utilisé cette classe, car mes informations sont considérées comme sures.



Vous pouvez voire sur ce printscreen qu'elle définit un certain nombre de propriété littérale. Ces mêmes propriétés que j'utiliser mon script XSLT, tout comme la classe *Hotel* qui sera avec un rdf:about **l'identifiant unique d'une ressource**. De plus, je vais réutiliser une partie des ces propriété dans mes ontologies suivantes.

En ouvrant l'ontologie dans Protégé, vous pourrez remarquer dans l'onglet *Individual* que j'ai créé certain valeurs de test à la main, mais aussi que j'ai créé manuellement les caractéristiques que peut recevoir un hôtel.
Ce choix s'explique car il m'était plus facile de lister puis de créer manuellement ces caractéristiques que d'automatiser cette tâche. Le fichier XML comportant notamment un problème d'accent, donc il m'aura été impossible de savoir si le ' ?' remplaçait un 'é' ou 'è' ou 'à' ou 'ç' …

<value language="fr" id="WBX00020010000000119">Piscine ext?rieure</value>

Pour automatiser l'indexation du travail j'ai créé un petit script XSLT pour lister l'ensemble des caractéristiques.

```
<xsl:for-each select="export/object">
    <xsl:for-each select="features/object">
        <xsl:value-of select="attribute[4]/value[0]"></xsl:value-of>
        ;
        <xsl:value-of select="attribute[4]/value[0]/@id"></xsl:value-of>
    </xsl:for-each>
```

Et ainsi obtenir un fichier texte avec la liste de tous caractéristiques pour tous les hôtels.

```
Sp?cialis? pour familles ; WBX00020010000111676
Garage priv? ; WBX00020010000111685
Places de stationnement de l'h?tel ; TDS00020010000158850
```

Ensuite j'ai écris une fonction en PHP pour créer un tableau contenant de manière unique toutes les caractéristiques disponibles. Merci à la fonction *array_search()*.

```
$i = 1; $features_cle = array(0 => 'cle'); $features_texte = array(0 => 'texte');
$fp = fopen("feature_liste.txt","r");
while (!feof($fp)) {
        $maLigne = fgets($fp, 255);
        $part =explode(';', $maLigne);
        $texte = $part[0];  $cle= $part[1];

        if(isexist($cle, $text, $features_cle)){
                $features_cle[$i] = $cle;
                $features_texte[$i] = $texte ;
                $i++; }  }

function isexist($cle, $text, $features){
        if(!array_search($cle, $features)){ return true;
        } else {return false} }
```

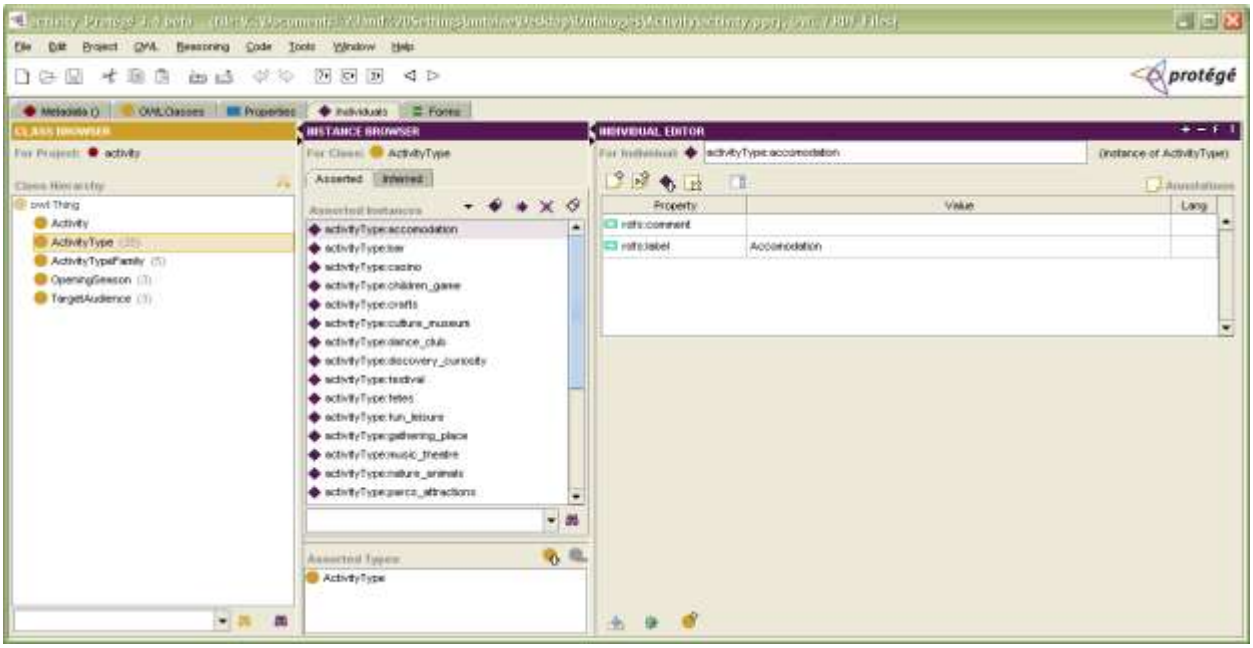
Finalement il ne me restait plus qu'à copier les valeurs et les Id dans Protégé.

Activtiy.owl

Cette ontologie définit les concepts d'activité, de famille et de type d'activité, de saison d'ouverture et de publique cible. Elle dispose également de quelques propriétés dont le prix, l'accès, une particularité, sa difficulté ainsi que ces périodes d'ouverture.

Je voulais utiliser la propriété difficulté dans mon démonstrateur, pour restreindre les suggestions disponibles suivant le caractère d'une personne. Finalement j'ai abandonné cette idée vu qu'une grande partie des activités disponibles venaient d'un export du site www.loisirs.ch et que malheureusement je n'avais pas cette appréciation.

On peut noter également que nous avons choisi une fois encore de crée un certain nombre de données manuellement dans l'onglet *Individual*
Toutes ces données se retrouvent sur le site :
http://www.tourismesuisse.com/valais/activite/f_a_1_0_0-tourisme.html et
http://www.loisirs.ch/



A noter aussi l'utilisation de deux propriétés symétriques :

- hasSimilarActivity

Je pensais utiliser cette propriété pour calculer avec mon moteur d'inférence toutes les autres activités qui seraient du même type. Le principe fonctionne très bien et les règles d'inférences à ce sujet aussi mais j'ai trouvé plus élégant de ne pas utilisé cette approche.
En effet, si je reprends les requêtes de mon démonstrateur, il m'est facile de lister toutes les activités d'un certain type et ensuite de les présenter à l'utilisateur. De plus, cette approche est nettement plus performante car je calcule ces données qu'au moment ou j'en ai besoin et non au démarrage d'OntoMea.

- hasNearActivity

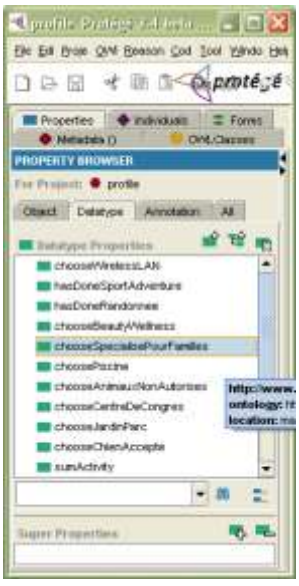
A nouveau, je pensais utiliser cette propriété pour saisir des activités proches. Cette fois la solution est venu de GeoNames. Comme l'ai précisé précédemment, lors de mon export XSLT, j'ai rajouté une donnée RDF spécifiant la nature de mon information
http ://www.geonames.org/ontology/#Feature

Et grâce à OntoMea ainsi que les Web services mis à disposition par GeoNames, l'application à été chercher automatiquement un certain nombre d'informations sur Internet concernant les lieux. De plus, le raisonneur a automatiquement rajouté un certain nombre de propriété sur toutes mes instances de lieux GeoNamess en rapport à la propriété *isLocationOfLind* de memont.owl. Je dispose ainsi d'un moyen de lister automatiquement et uniquement lors que c'est nécessaire tous les hôtels et activités d'un lieu.

Exemple pour Saas-Fee :
Une interrogation de toutes les informations disponibles pour l'uri suivante, qui est d'ailleurs aussi une url valide, http://sws.geonames.org/2658898/

```
http://www.memoria-mea.ch/memonto.owl#isLocationOfLink
http://www.websemantique.ch/onto/hotel/CH_3906_hotel_imseng
http://www.memoria-mea.ch/memonto.owl#isLocationOfLink
http://www.websemantique.ch/onto/hotel/CH_3906_hotel_elite_saas_fee
http://www.memoria-mea.ch/memonto.owl#isLocationOfLink
http://www.websemantique.ch/onto/hotel/CH_3906_hotel_europa_annex
…
```

Profil.owl



Cette classe toute simple qui ne fait que de définir le concept de profile me sert de modèle pour mes utilisateurs enregistrés. Elle définit quelques propriétés vers mes deux objets *Hotel* et *Activity*. hasBeenX pour ce qu'elle a fait ou elle a déjà été et likeToX pour ce qu'elle aimerai faire ou aller. Les objets ne sont pas disponible pour l'instant mais elles le seront grâce à quelques règles de profiles que j'ai écris.

Pour ce qui est des données littérales, vous pourrez constater que j'ai en crée beaucoup. Dont, la somme des hôtels où une personne est allée, ainsi que des chooseX pour les caractéristiques d'un hôtel et hasDoneX pour une activité. Ces propriétés n'acceptent

que des *int* qui me serve à compter le nombre de fois où une personne à fait quelque chose. Je réutiliserai ces nombres essayé de déduire les goûts d'une personne.

Je reviendrais après sur cette limitation que j'ai eue avec les règles d'inférences. La raison est que ni SPARQL ni le raisonneur ne sont capables de faire un COUNT.

5.3 ONTOMEA

Dans ce chapitre je vais reprendre en grande partie le travail de Frédéric Favre concernant OntoMea et les raisonneurs. C'est lui qui c'est occupé d'un certain nombre d'améliorations. Je me permets de reprendre ce chapitre car j'ai passé un certain temps à étudier et utiliser ce produit. De plus, il me semble essentiel de présenter cette partie pour comprendre et manipuler mon démonstrateur.

Dans le cadre du projet Memoria-Mea, le moteur sémantique OntoMea est utilisé afin de traiter les données provenant de différentes informations multimédias concernant un utilisateur. Son rôle est de traiter les données sémantiques et de fournir un système de déductions basé sur les ontologies fournies.

Les différentes informations sont formatées et intégrées dans la base de connaissances du moteur OntoMea. Elles sont ensuite disponibles grâce aux différents services fournis par un serveur Web embarqué.

Au niveau technique, le moteur OntoMea est développé en Java en se basant sur le framework Jena. Les données sont stockées dans une base HSQLDB. Ce moteur est destiné à un usage local et gère ainsi les données propres à un utilisateur.

La base de connaissance se compose de trois **graphes** distincts:

- Un graphe de données (asserted model) contenant les différentes instances.
- Un graphe de schémas (schema model) contenant les ontologies.
- Un graphe d'inférences (inferred model) déduit à partir du graphe de données et des schémas d'ontologies.

Structure

La solution OntoMea comporte un répertoire KBOntoMea contenant toutes les informations nécessaires à la gestion des données sémantiques. Ce répertoire est structuré de la façon suivante:

- Un répertoire Data contenant l'ensemble des données chargées lors du lancement du moteur.
- Un répertoire Schémas contenant les différents schémas d'ontologies.

Configuration

La configuration du moteur OntoMea est basée sur le fichier *OntoMeaConfig.xml* disponible à la racine de la solution. Il contient les informations concernant l'adresse et le port sur lesquels les services sont atteignables.

Il est également possible de spécifier les schémas d'ontologies que l'on désire charger dans la base de connaissances lors du démarrage. Les informations nécessaires sur les schémas sont les suivantes:

- Le nom de l'ontologie.
- L'Url du fichier décrivant l'ontologie.
- Le lien en local où sauver le schéma après son téléchargement.

Par défaut, le moteur contrôle si le fichier existe déjà dans le répertoire des schémas. Dans le cas contraire, il le télécharge depuis l'Url définit dans le fichier de configuration.
Il permet finalement de configure les différent modes du raisonneur.

Services

Le moteur OntoMea propose trois services. Ces services sont basés sur l'architecture de Joseki et sont interrogeables directement à travers le réseau via l'adresse et le port

spécifié dans le fichier de configuration. Les différents services disponibles sont les suivants:

- KBService qui propose plusieurs fonctionnalités permettant d'interroger et de gérer la base de connaissances. Par exemple, il est possible de récupérer les données propres à un utilisateur ou de les mettre à jour.
- LocService permet d'interroger les Web Service de GeoNames afin de récupérer des données géographiques sous forme XML. C'est ce service qui m'a récupéré les données sur mes localisations que je vais utiliser ultérieurement.
- SparqlService permet d'effectuer directement une requête SPARQL sur le graphe inféré. C'est ce service que j'ai utilisé dans mon démonstrateur PHP.

Serveur Web

Le serveur Web embarqué permet d'accéder aux différents services. Il peut être appelé directement depuis n'importe quelle application (.NET, Java, PHP, …). Le résultat est renvoyé en format XML.

Par exemple, pour interroger le service LocService afin de connaître via GeoNames l'ensemble des pays d'Europe, il faut utilise l'Url suivantes:

```
http://localhost:2020/loc?query=countries_list&continentCode=eu&lang=fr
```

Cette requête est composée des éléments suivants:

- localhost correspond à l'adresse du serveur. Le moteur OntoMea étant destiné à une utilisation locale, cette valeur vaut par défaut localhost.
- 2020 correspond au numéro de port par défaut. A modifier si l'on souhaite traverser un firewall en spécifiant le port 80 (HTTP) ou 443 (HTTPS).
- loc correspond au service des locations. Cette argument peut-être remplacé par les valeurs « kb » ou « sparql » selon le service à interroger.
- countries_list détermine la méthode du service que l'on désire interroger.
- &continentCode=eu&lang=fr correspond aux arguments nécessaires à la méthode.

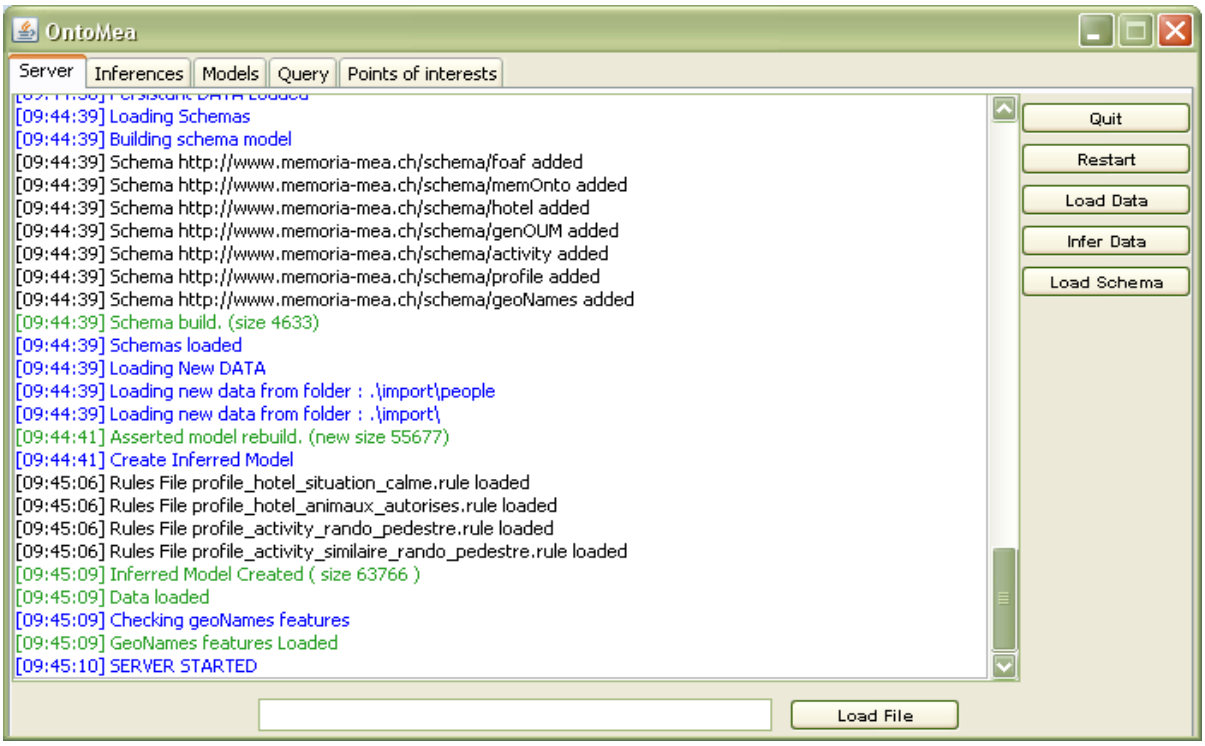
5.4 L'INTERFACE D'ONTOMEA

L'interface a été séparée en cinq onglets distincts:

- « Main » est l'interface principale du moteur. On y retrouve les principales fonctionnalités de gestion ainsi que les logs de fonctionnement du serveur.
- « Inferences » permet de configurer en cours d'exécution le moteur d'inférences d'OntoMea.
- « Models » permet de visionner et d'exporter les Named Graphs contenus dans la base de connaissances.
- « Query » affiche l'ensemble des requêtes effectuées sur la base de connaissances et, le cas échéant, les erreurs survenues.
- « Points of interests » permet d'importer les données des points d'intérêt d'un utilisateur. *(Voir le travail de diplôme de Fréderic Favre)*

Les différentes fonctionnalités de ces onglets sont détaillées dans les chapitres suivants.

Le menu « Main »



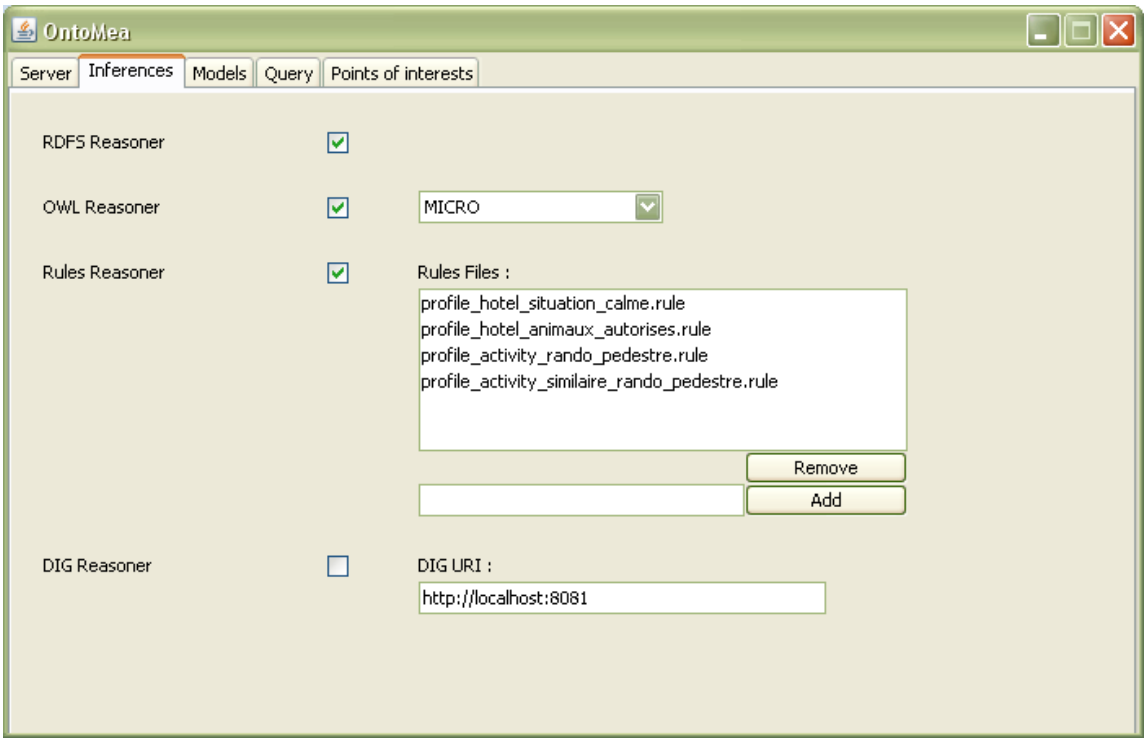
Cette interface apparaît par défaut lors du lancement de l'application. Elle comporte l'ensemble des logs de fonctionnement du moteur. La couleur indique le type de logs affichés. Les couleurs utilisées par l'interface sont les suivantes:

- Noir pour les informations de base (modification de la configuration, chargement des Named Graphs, …).
- Bleu pour définir les étapes lors du fonctionnement du moteur (chargement des données, création du graphe d'inférences).
- Vert pour les opérations modifiant la base de connaissances (ajout de données, mise à jour des graphes de base ou de schémas, exportation d'un graphe,…)
- Rouge pour les erreurs de fonctionnement du serveur.

En plus de la journalisation des évènements, il est possible d'effectuer diverses opérations sur le serveur en cours d'exécution:

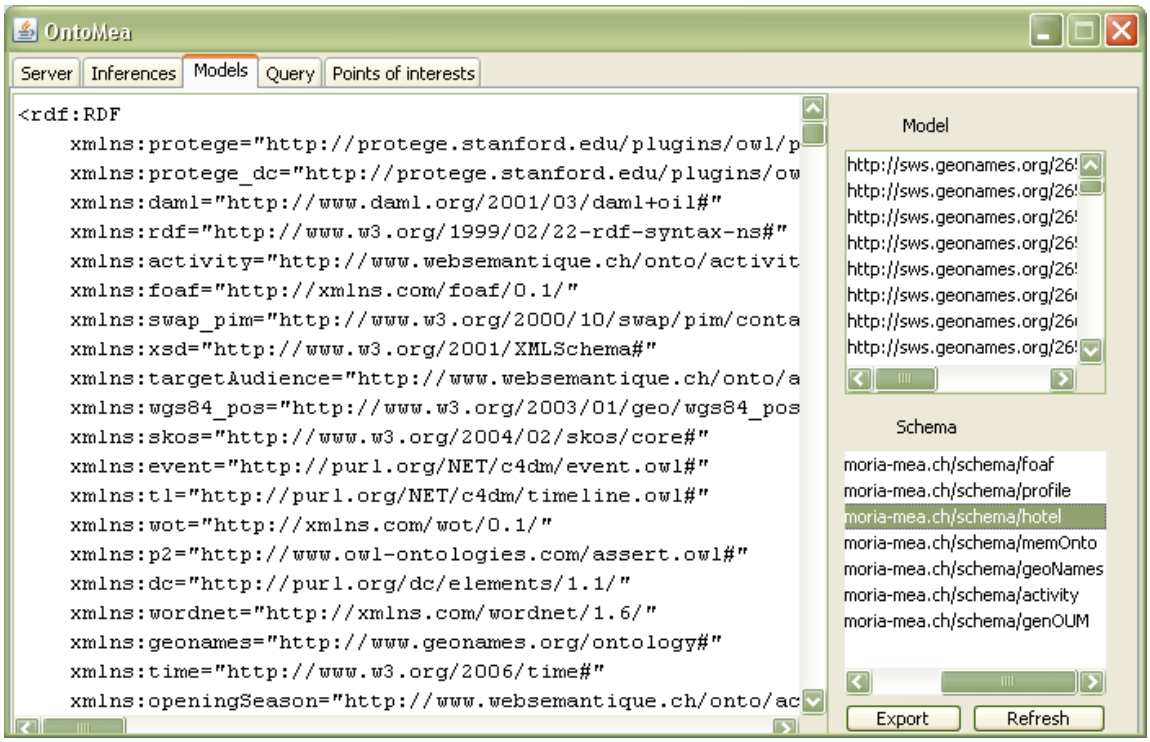
- « Quit » permet de quitter l'application.
- « Restart » permet de redémarrer le serveur. Cette fonctionnalité est utile si l'on désire prendre en compte des modifications effectuées dans le fichier de configuration.
- « Load Data » vérifie s'il existe des données dans le répertoire import et les intègre dans la base de connaissances.
- « Load Schemas » permet de recharger les schémas définis dans le fichier de configuration.
- « Infer Data » force le moteur d'inférences à régénérer le graphe inféré. Cette fonctionnalité est utile si l'on désire tester le nombre de déductions obtenues par les divers raisonneurs.
- « Load File » permet de charger un fichier RDF ou OWL dans la base de connaissances. Il est possible d'y insérer un chemin local ou une URL. Si le fichier est valide, un nouveau Named Graphs persistant est généré.

Le menu « Inferences »



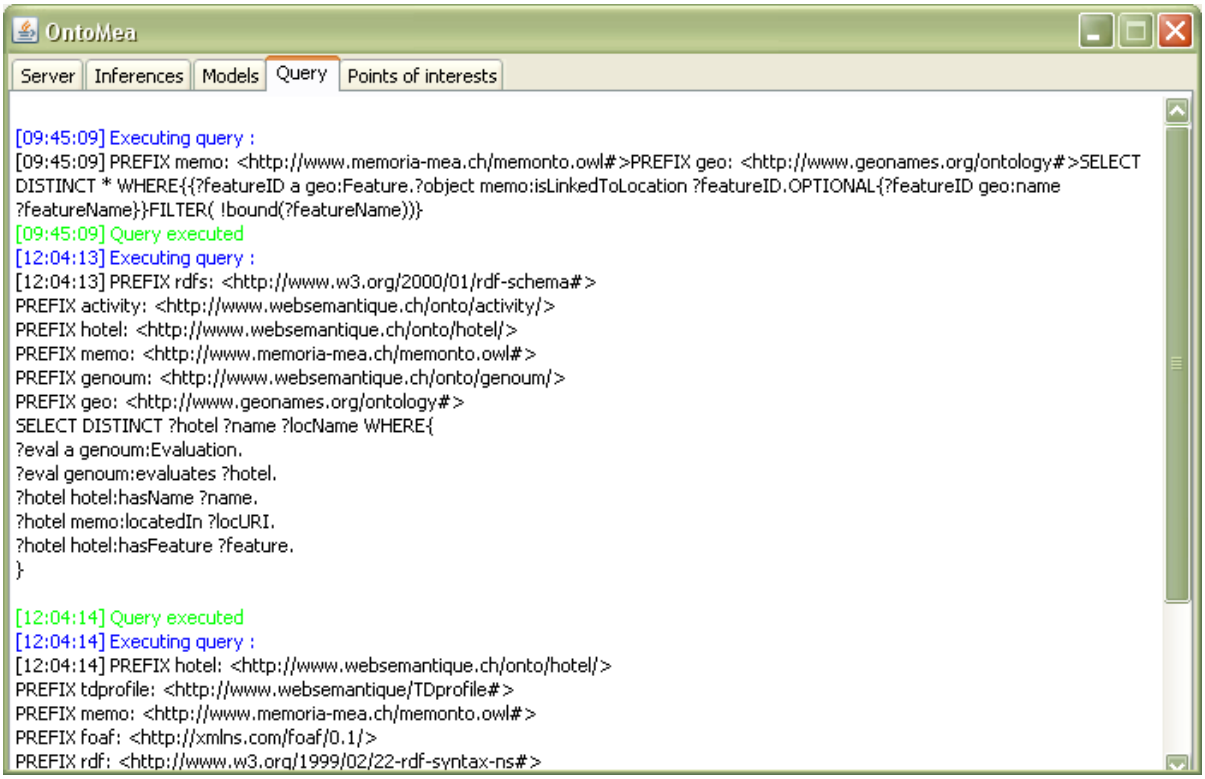
Au lancement de l'application, l'interface de gestion des inférences reprend les paramètres du fichier de configuration. Il est possible de les modifier en cours d'exécution.
Chaque modification depuis cette interface entraînera une régénération du graphe d'inférences lors d'une requête sur la base de connaissances. Cependant, les modifications effectuées ne modifient pas le fichier de configuration.
Les paramètres de configuration des différents raisonneurs sont expliqués en détail dans le chapitre Les différents raisonneurs.

Le menu « Models »



Cette interface permet de visualiser l'ensemble des Named Graphs contenus dans la base de connaissances. Elle est composée de deux listes, l'une pour les graphes de données et l'autre pour les schémas. En sélectionnant un graphe, il est possible d'afficher l'ensemble des triplets qu'il contient sous format RDF.
Il est également possible d'exporter un graphe dans un fichier RDF. Les fichiers exportés sont placés automatiquement dans le répertoire « Export » à la racine de la solution.

Le menu « Query »



En cliquant sur l'onglet « Query », il est possible de visualiser l'ensemble des requêtes effectuées sur la base de connaissances. Les couleurs utilisées sont les suivantes:

- Noir pour les requêtes.
- Bleu lors de la réception d'une requête.
- Vert lorsque la requête a été exécutée avec succès.
- Rouge lorsqu'une erreur s'est produite.

5.5 LES DIFFÉRENTS RAISONNEURS

Un raisonneur est un logiciel basé sur un algorithme qui permet de:

- Créer des **inférences** à partir d'une ontologie et permet donc de créer automatiquement une nouvelle information.
  Par exemple, soit deux personnes et une relation symétrique 'ami de', si la personne A est ami à la personne B alors le raisonneur peut déduire de lui même que la personne B est ami à la personne A.

- Valider le schéma d'une ontologie et ainsi vérifier qu'il n'a ai pas d'inconsistances. Par exemple, l'héritage de deux classes disjointes.

Il existe pour chaque langage, un raisonneur particulier. La complexité et la puissance des déductions dépendent du type de raisonneur choisi.
Dans le moteur OntoMea, les quatre raisonneurs suivants sont disponibles:

- RDFS
- OWL
- DIG
- Générique (utilise des règles prédéfinies)

Raisonneur RDFS

Le raisonneur RDFS est la solution la plus légère mais également la plus performante. Elle est idéale pour les ontologies simples n'utilisant que les règles de déductions suivantes :

- subClassOf
- subPropertyOf
- type
- domain
- range

Exemple de déduction possible effectuée par le raisonneur RDFS. Grâce à la propriété 'subClassOf', nous allons gérer l'héritage.

```
<rdf:Description rdf:about=" #Mere">
   <rdfs:subClassOf rdf:resource=" #Personne"/>
</rdf:Description>
<rdf:Description rdf:about=" # Pere ">
   <rdfs:subClassOf rdf:resource=" #Personne"/>
</rdf:Description>

<rdf:Description rdf:about=" #Antoine">
   <rdf:type rdf:resource=" #Pere"/>
</rdf:Description>
```

Cette ontologie précise que les classes 'Mere et 'Pere' sont des sous-classes de 'Personne' et que 'Antoine' est une instance de la classe 'Pere.

Ensuite, si l'on désire récupérer toutes les ressources du type 'Personne' il ne nous reste plus qu'à activer le raisonneur RDFS. Ainsi, le raisonneur logiquement va ajoute au modèle de base les lignes suivantes. Car si Antoine est un père alors c'est une personne aussi. Fonction transitive : (A -> B) & (B -> C) -> (A->C)

```
<rdf:Description rdf:about=" #Antoine">
   <rdf:type rdf:resource=" #Personne"/>
</rdf:Description>
```

Ce raisonnement est commun à tous les raisonneurs. Le nombre et la complexité des nouvelles données déduites dépendent du type de raisonneur.

Raisonneur OWL

Le raisonneur OWL est une implémentation du standard OWL/Lite. S'il est nécessaire d'utiliser un raisonneur OWL/DL, il est préférable d'utiliser l'interface DIG d'OntoMea.
Jena offre trois niveaux de complexité pour son raisonneur OWL :

- Micro
- Mini
- Full

Le niveau de complexité augmente selon le type de raisonneur sélectionné et, en contre partie, les performances diminuent. Malgré tout, OWL/Micro permet d'effectuer la plupart des déductions OWL et reste le meilleur compromis pour la plupart des ontologies.

Le raisonneur OWL permet d'effectuer des déductions plus complexes que RDFS. Il prend en charge la notion de restriction. Ainsi, il est possible de créer des classes définies avec certaines conditions de restriction.

Par exemple, si la classe 'Car' a une restriction 'owl :hasValue 4' sur la propriété 'hasWheelNumber', le raisonneur déduira automatiquement que l'instance suivante appartient à la classe 'Car' :

```
<Vehicule rdf:ID=" #MyVehicule">
   <has:WheelNumber>4</ has:WheelNumber >
</Vehicule >
```

Interface DIG

Le dernier raisonneur disponible dans l'application OntoMea est un raisonneur interfacé avec un moteur externe offrant une interface DIG. Ce genre de moteur utilise le langage OWL/DL et peut être une alternative au raisonneur OWL de Jena qui est une implémentation de OWL/Lite.

Afin de configurer OntoMea pour une utilisation avec un moteur externe, il est nécessaire de préciser dans le fichier de configuration l'URL de l'interface DIG.
Le graphe d'inférences généré par ce processus ne contient pas les nouvelles déductions par défaut. Lors d'une requête, le moteur d'inférences externe sera interrogé.

A noter qu'OWL/DL, bien que plus complète, nécessite une plus grande précision dans les schémas d'ontologies. De nombreux schémas fonctionnant parfaitement avec OWL/Lite peuvent engendrer des erreurs lors de leur utilisation via l'interface DIG.

Raisonneur Générique

Le raisonneur générique de Jena permet d'exécuter des inférences selon un jeu de règles définies. Ces règles peuvent provenir de jeux déjà définis comme RDFS ou OWL. Mais il est également possible de créer ses propres règles en se basant sur la syntaxe fournie par Jena.

Dans le moteur OntoMea, le raisonneur générique fonctionne en parallèle avec les autres raisonneurs. Il n'est donc pas nécessaire d'ajouter des jeux de règles RDFS ou OWL lors de la création de règles personnalisées.
Il est possible dans le fichier de configuration d'OntoMea de préciser quels sont les fichiers contenant les règles personnalisées. Les formats admis pour la balise <uri>x</uri> sont les suivants :

- l'URL du fichier de règles.
- Le chemin absolu du fichier sur la machine locale.
- Le nom du fichier à condition qu'il se trouve dans le répertoire KBOntoMea/Data/Rules.

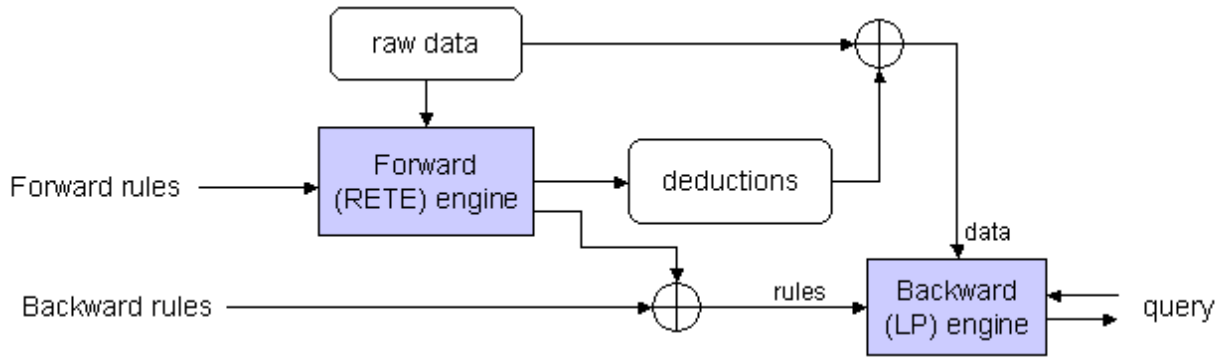
Les modes de fonctionnements du raisonneur générique

Le raisonneur générique est en fait constitué de deux moteurs de règles distincts :

- *Forward engine* permet de déduire des inférences pendant la création du graphe inféré. Ce graphe contient alors les données de base ainsi que les nouvelles déductions générées par le raisonneur générique.

- *Backward engine* est un moteur d'inférence qui s'exécute lors d'une requête sur le graphe d'inférences. Aucunes nouvelles données ne sont ajoutées aux données de bases. Lorsque des inférences sont déduites lors de l'exécution d'une requête, les résultats obtenus sont stockés en mémoire.

Le moteur OntoMea utilise par défaut la configuration hybride. Cette structure permet d'effectuer des inférences en deux temps. Les règles attribuées au *Backward engine* peuvent se baser sur les résultats obtenus par le *Forward engine*.



Format de règles du raisonneur générique

Le format utilisé par les règles personnalisées de Jena est composé de trois parties distinctes :

- Une liste de préfix qui définissent les ontologies utilisées dans les règles.
  `@prefix foaf: <http://xmlns.com/foaf/0.1/>.`
- Une liste de conditions (prémisses).
- Une liste de conclusions.

```
Rule      :=   bare-rule .
          or   [ bare-rule ]
             or   [ ruleName : bare-rule ]

bare-rule :=   term, ... term -> hterm, ... hterm    // forward rule
          or   term, ... term <- term, ... term    // backward rule

hterm     :=   term
          or   [ bare-rule ]

term      :=   (node, node, node)           // triple pattern
          or   (node, node, functor)        // extended triple pattern
          or   builtin(node, ... node)      // invoke procedural primitive

functor   :=   functorName(node, ... node)  // structured literal

node      :=   uri-ref                        // e.g. http://foo.com/eg
          or   prefix:localname                       // e.g. rdf:type
```

```
          or   <uri-ref>                    // e.g. <myscheme:myuri>
          or   ?varname                   // variable
          or   'a literal'                // a plain string literal
          or   'lex'^^typeURI             // a typed literal, xsd:* type
names supported
          or   number                     // e.g. 42 or 25.5
```

Le tableau de description de la syntaxe propre au 'général purpose rule engine'

Le moteur d'inférences analyse les conditions et, s'il trouve des ressources qui les vérifient, ajoute alors les déclarations définies dans les conclusions de la règle. Les conditions peuvent prendre deux formes différentes :

- Un triplet. Par exemple: ?s rdfs :subClassOf ?p
  signifie que la ressource ?s doit être une sous-classe de ?p.
- Une fonction. Par exemple: greaterThan(?x, ?y).

Les conclusions ne sont possibles que sous la forme de triplets.

Il existe deux modes d'exécution des règles. Comme défini précédemment, les règles peuvent être exécutées avant ou après la création du graphe d'inférences. Pour spécifier le mode d'exécution d'une règle personnalisée, il est nécessaire d'utiliser les formats suivants :

- conditions -> conclusions pour le mode *forward*.
- conclusions <- conditions pour le mode *backward*

Il est évidement nécessaire que le raisonneur soit configuré dans le mode approprié ou dans le mode hybride. Dans le cas contraire, la règle sera exécutée dans le mode défini par le raisonneur.
(Voir : http://jena.sourceforge.net/inference/)

Exemple et limitation

Ci-dessous, un exemple d'une des mes règles pour définir les préférences en matière d'hôtel.
La règle sélectionne le nombre de fois où une personne a été dans un hôtel avec les caractéristiques suivantes : animaux autorisés ou chien autorisés ainsi que les animaux non-autorisés.

A partir de ces données, je calcule si le quotient du nombre de fois où il a choisi *animaux autorisés* est supérieur à 80% et les *non-autorisés* inférieur à 20%. De la j'en conclue que cette personne voyage avec un animal de compagnie c'est pourquoi je vais lui suggérer un hôtel qui a cette caractéristique.

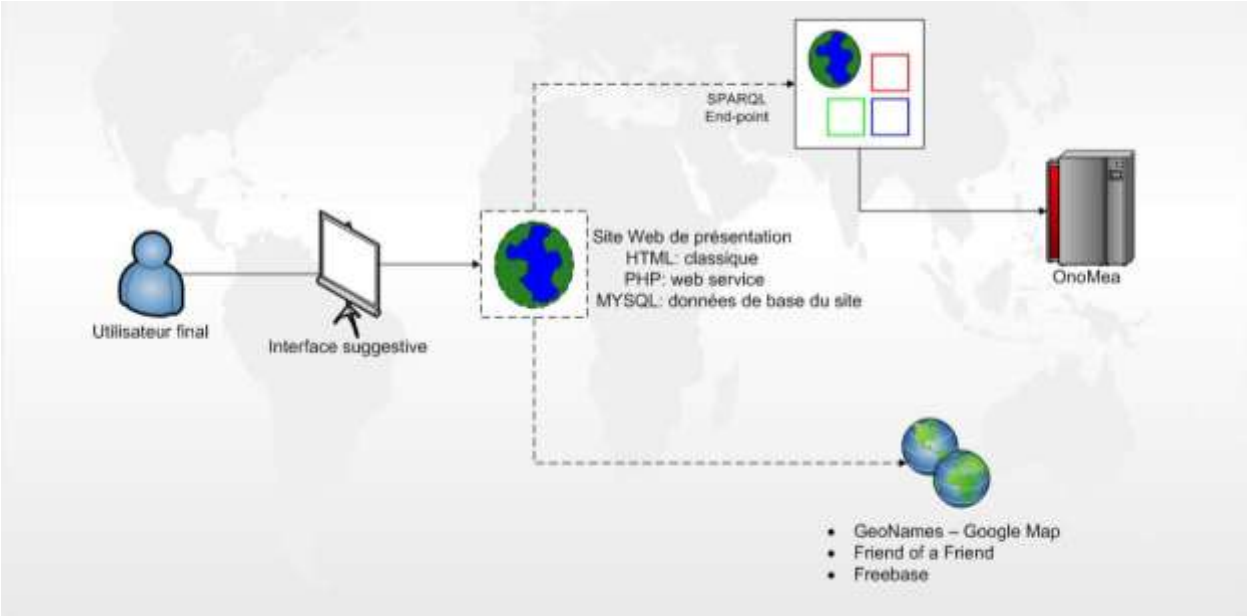
```
@prefix tdprofile: <http://www.websemantique/TDprofile#>.
@prefix memo: <http://www.memoria-mea.ch/memonto.owl#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
[animaux_autorises:
(?personne tdprofile:chooseAnimauxAutorises ?a),
(?personne tdprofile:chooseChienAccepte ?b), max(?a, ?b, ?c),
(?personne tdprofile:sumHotel ?sH), quotient(?c, ?sH, ?qu), greaterThan(?qu, 0.8),
(?personne tdprofile:chooseAnimauxNonAutorises ?d),
quotient(?d, ?sH, ?qu2), lessThan(?qu2, 0.2)
-> (?personne tdprofile:likeToHaveHotel
<http://www.websemantique.ch/onto/hotel/Chien_accepte>)]
```

Comme vous pouvez le remarquer en analysant le code ci-dessus, j'ai créé des propriétés qui nous donnent le nombre de fois où une personne a choisi une caractéristique. Evidemment, j'ai tout d'abord cherché à calculer cette somme à l'aide d'une règle et malheureusement je suis tombé sur une des limitations des raisonneurs.
Actuellement, il n'a pas de support pour les fonctions d'agrégations dans Jena donc il m'est impossible de compter dans ma règles le nombre d'occurrences d'une caractéristique par rapport à un utilisateur. On pourrait imaginer contourner cette limitation avec le langage SPARQL d'ARC *(Voir : http://arc.semsol.org/)*

6. PHASE DE DÉVELOPPEMENT DU DÉMONSTRATEUR

Dans ce chapitre, je vais présenter les idées et le fonctionnement de mon démonstrateur. L'idée général étant de mettre en avant l'intérêt que pourrait avoir le monde du tourisme à sémantiser ses informations.

Ci-dessous, un schéma qui explique l'architecture générale de mon application Web.



6.1 IDÉES GÉNÉRALES POUR LE DEMONSTRATEUR

Je vais simuler 3 utilisateurs, à partir de mon ontologie *Profil* et *Foaf,* qui ont déjà réalisé quelques activités chacun ainsi que deux ou trois réservations dans un hôtel particulier (DA). Le but de cette simulation est de fournir un échantillon de données pour pouvoir inférer des tendances et des goûts chez chacun d'eux.

Note : on peut imaginer obtenir ces informations à partir du projet CTMS. Il s'agit d'un projet qui permettrait d'attribuer un identifiant unique pour un touriste et lui permet ainsi d'accéder à divers activités au sein d'une région.
(Voir : http://www.websemantique.ch/index.php?option=com_content&task=view&id=52&Itemid=138)

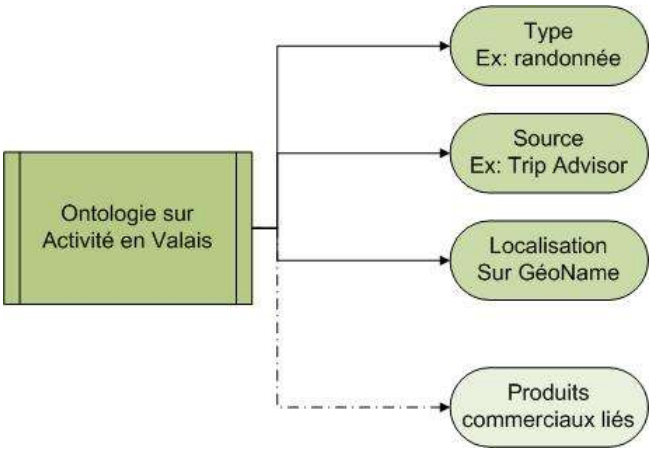
Depuis ces enregistrements au format RDF, on pourrait réaliser plusieurs scénarios :

1. A partir des nos données (DA), on va écrire une série de règles (fichier .rule) et déduire ces préférences en matière de location de chambre d'hôtel. Ainsi on va générer automatiquement ses données personnelles.

   Exemple : si quelqu'un loue systématique une chambre dans un hôtel qui contient un service spéciale pour les chiens, on peut en déduire que cette personne voyage toujours avec son animal de compagnie, donc il faudra lui suggérer un hôtel avec un tel service et bien entendu ne pas lui proposer des hôtels où les animaux sont proscrits.

   Note : il est évident que pour réaliser une telle démonstration, nos activités de base devront être très ciblées sur des choix relevant, et ainsi nous permettre de déduire une vérité marquante.

2. J'ai créé une ontologie en rapport aux activités en Valais. Il s'agirait simplement de stocker quelques exemples d'activités sur un modèle très simple. Au maximum une dizaine d'activité avec quelques informations les caractérisant :



Depuis nos données (DA), on va écrire une nouvelle série de règles et déduire les préférences en matière d'activité.

Suite à notre premier scénario, on pourrait imaginer insérer d'abord un choix par rapport aux activités, donc déterminé une région et ensuite suggérer un hôtel dans cette région. Ou même calculer la distance entre différent hôtels et activités pour proposer une solution de proximité.

3. On pourrait encore simuler quelques données commerciales qu'on stockerait soit dans notre ontologie d'activité, soit dans une ontologie propre aux objets commerciaux pour suggérer à un client des articles en relations à sa préférence.

   Exemple : pour reprendre le cas de tourisme qui voyage avec son chien, on pourrait lui suggérer un magasin spécialisé dans ce domaine ou, une promotion chez un partenaire affilié pour une marque X de croquette pour chien. Je n'ai pas vraiment exploré cette idée car ne présentait que peu d'intérêt.

Liste des caractéristiques et activités pour le démonstrateur

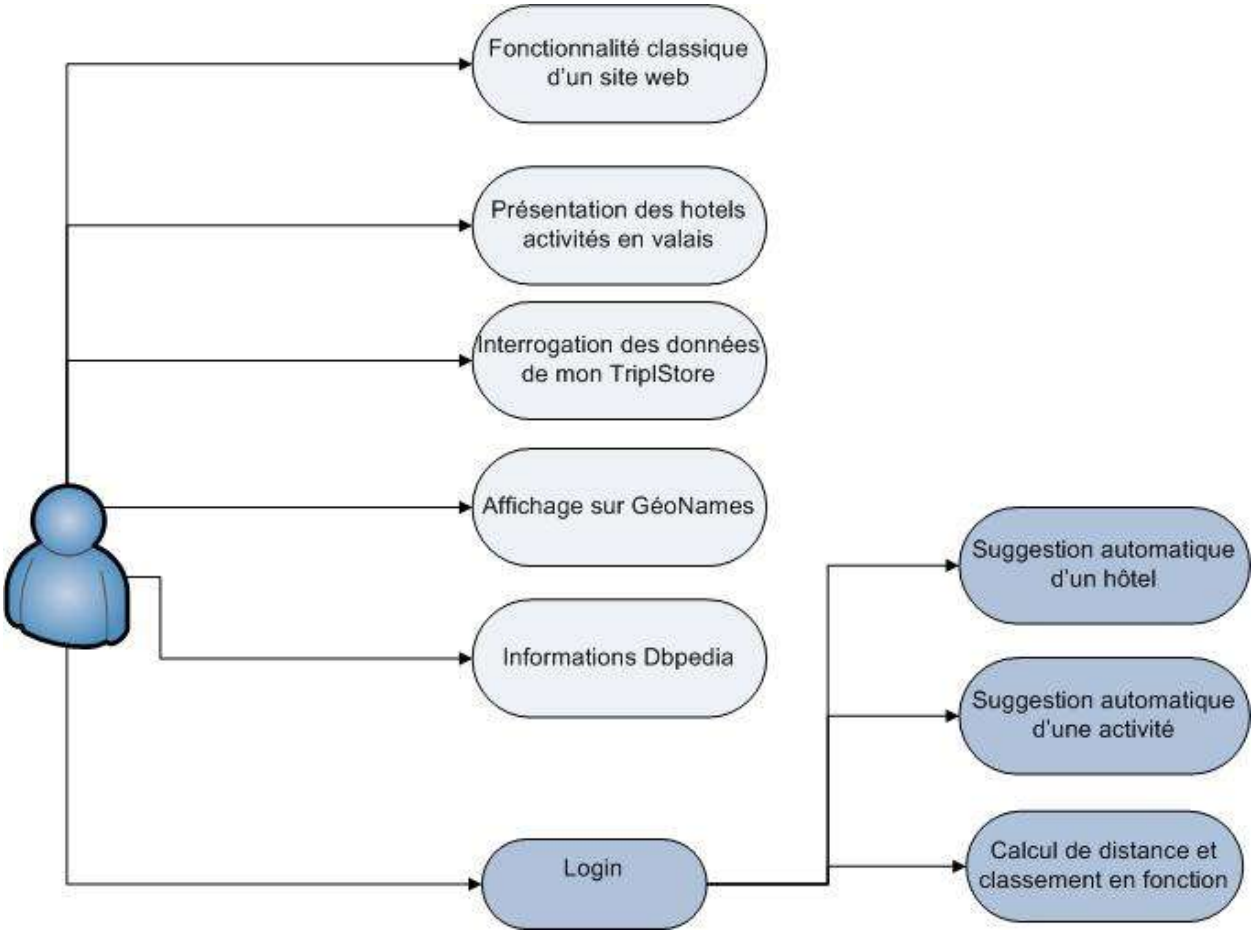
L'objectif étant de ciblée le résultat sur certaines caractéristiques déterminées. J'ai arbitrairement choisi ces caractéristiques et activités pour leur caractère démonstratif.

- Sur le choix des hôtels suivant leurs caractéristiques pour:
  - L'ami des animaux: animaux acceptés (chien) et non-acceptés
  - La sportive en vacances: beauty-wellness, piscine, salle de fitness, sauna
  - L'entrepreneur: centre de congres, connextion_modem_fax, wireless_LAN

- Sur certaines activités pour déterminer une région :
  - L'ami des animaux: faire ressortir un lac, une excursion
  - La sportive en vacances: n'importe quel sport dans la région de Zermatt,
  - L'entrepreneur: Verbier Festival, sites historiques, musées

(Voir lse site http://www.tourismesuisse.com/valais/activite/f_a_1_0_0-tourisme.html et http://www.loisirs.ch/)

Schéma des cas d'utilisation possible pour l'utilisateur final



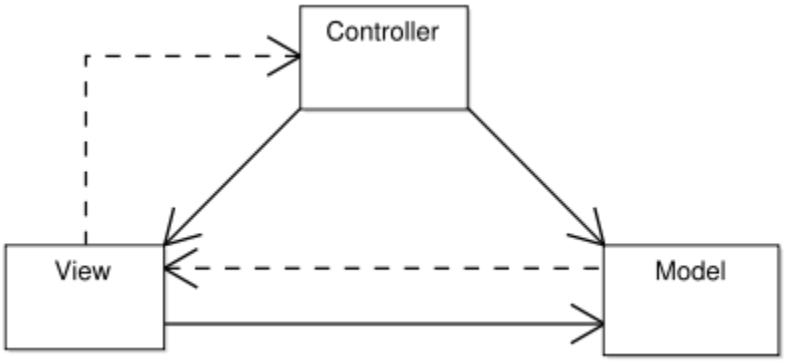
6.2 JOOMLA

Dans ce chapitre, je ne vais pas vous présenter l'outil CMS, mais certains points que je trouve intéressants comme son architecture MVC et l'utilisation des Web services REST d'OntoMea. J'y rajouter également un petit manuelle pour installer facilement mon site de démonstration à partir du dossier qui ce trouve sur le cd.

Joomla est un système de gestion de contenu (en anglais, CMS, pour *Content Management system*) créé par une équipe internationale de développeurs récompensée à maintes reprises. Un CMS est un logiciel Web qui vous permettra à un utilisateur landa de créer un site internet dynamique en toute simplicité.
Joomla est un CMS Open Source distribué sous licence GNU/GPL (gratuit) avec lequel vous pourrez mettre en ligne du contenu et mettre à disposition de vos visiteurs des services (forum, boutique en ligne, galerie photos,...), le tout sans connaissance technique particulière… enfin presque ;-)
(Source : http://www.joomla.fr/)

Pour ma part joomla est surtout un framework que j'utilise pour développer des sites dans mon travail. Cependant, c'était la première fois que j'utilisai la nouvelle version 1.5 Cette version apporte quelques changements par rapport à l'ancienne 1.13. Notamment elle utilise le *design patern* MVC pour la conception des composants.

Ce modèle d'architecture impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur.
(Source : http://fr.wikipedia.org/wiki/Mod%C3%A8le-Vue-Contr%C3%B4leur)



Le modèle

Le modèle représente le comportement de l'application : traitements des données, interactions avec la base de données, etc. Il décrit ou contient les données manipulées par l'application. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient. Le modèle offre des

méthodes pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle sont dénués de toute présentation.

Dans mon application, comme mes données ne se trouve par dans une base de données relationnelles, mes modèles vont préparer la *query* en fonction de différents paramètres et ensuite appeler le service Web d'OntoMea qui me renvoi de l'XML.

```
function getTd( $options=array(), $mypref ){
        $query = $this->_getTdQuery($options, $mypref);
        $serveur = "http://localhost:2020/sparql?query=";
        $request= $serveur . $query ;

        //Web service REST call
        $result = file_get_contents($request);
        $xml = simplexml_load_string($result);

        //var_dump ($xml);
        return @$xml;}
```

Exemple du modèle suggestion.php

Le contrôleur

Le contrôleur prend en charge la gestion des évènements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les évènements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour qu'elle se mette à jour. Certains évènements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier. Le contrôleur n'effectue aucun traitement, ne modifie aucune donnée. Il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondante à la demande.

Dans mon application, le contrôleur ne fait que de modifier les *views* car j'utilise généralement toujours le même modèle. Mon 2ème modèle listeinfo.php est une version simplifié du premier suggestion.php.

La vue

La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, …). Ces différents évènements sont envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle. Plusieurs vues, partielles ou non, peuvent afficher des informations d'un même modèle. Elle peut être conçue en html, ou tout autre « langage » de présentation.

Dans joomla, l'idée va encore un peu plus loin car la vue ne s'occupe pas des détails de l'affichage, elle s'occupe d'assigner certaines informations à la page et aussi de référencer les données du modèle. Elle passe ensuite le relais au *template* qui lui s'occupe de l'affichage des données assignées ainsi que de l'interface avec l'utilisateur. Attention, le *template* ci-dessus n'a rien à voir avec le *template* général du site.
On peut noter l'utilisation du $this auquel les données ont été assignées.

```
class TdViewSuggestion extends JView {
	function display($tpl = null) {
		$user =& JFactory::getUser();
		$myuser = $user->username ;
		if($myuser != NULL){
			$this->assignRef('myuser', $myuser);
			$model = &$this->getModel();
			$mypreferences = NULL;

			$file = "generique_hasdone_activity.sparql";
			$hasdone_activity = $model->getTd($file, $mypreferences);
			$this->assignRef('hasdone_activity' , $hasdone_activity);

			parent::display($tpl);
		} else {
			echo "<h3>Vous devez vous connecter!</h3>";
		} } }
```

Une partie de la vue suggestion\view.html.php

Les services REST

Concrètement, l'infrastructure REST (pour REpresentational State Transfer) repose sur le même principe de fonctionnement que les Web Services. Elle s'appuie sur les protocoles Internet, dont HTTP, pour véhiculer des messages décrits au format XML. L'idée étant équivalente : disposer d'une interface d'intégration inter applicative non-intrusive pour les systèmes en présence. Principale différence : REST se limite au champ du transfert de données d'une application à l'autre.

Pour preuve : le module en marque blanche proposé Amazon.com est exploitée à 85% en mode d'interrogation REST, contre 5% en mode Web Services.

Malgré tout, l'interface REST affiche de nombreuses lacunes. Elle ne supporte par exemple aucun mécanisme standardisé de sécurité ni de gestion de l'intégrité des flux. Autre manque flagrant à noter : l'absence de dispositif d'orchestration de processus métier.
(Source : http://www.journaldunet.com/solutions/0410/041026_webservices.shtml)

Vous pouvez voir sur la séquence de code suivante qu'il est très facile d'appeler ce service en PHP.

```
//Web service REST call
$result = file_get_contents($request);
```

Installation

Pour installer mon démonstrateur, nous aurons besoin de :

1. Un moteur PHP type EasyPhp ou XAMPP. Ce sont des kits d'installations d'Apache qui contiennent MySQL, PHP et PhpMyAdmin. Ils sont réellement très faciles à installer et à utiliser.

Personnellement, je recommande XAMPP que j'utilise tous les jours. Il suffit de lancer l'application et vérifier l'adresse http://localhost/xampp

2. OntoMea avec naturellement les ontologies hotel, activity et profil ainsi que les données RDF de l'export Tomas, les exports de Fabian sur les activités et les hôtels de Tripadvisor.ch et loisirs.ch ainsi que d'un utilisateur test que j'ai nommé touriste.rdf.

Il faudra aussi lancer les quelques règles d'inférences que j'ai écrit. Vous trouver sur le cd annexe tous ces documents.

Une fois ces deux applications installées, il faut copier le site à proprement parlé. Soit le répertoire travail_diplome_site sous *C:\Xampp\htdocs\x*.
Ensuite on va importer la base de données depuis l'outil phpMyAdmin. On choisit importer, on utilise la section *fichier à importer* pour sélectionner notre sauvegarde au format sql (*db.sql* à la racine du site), et on click sur exécuter en bas à droite.

Note pour l'import: il est possible qu'il faille créer manuellement la base. Il suffit de cliquer sur la petite maison (accueil) tout en haut à gauche. Sur cette page on choisit créer un base de donnée : *websemantique*, type: *interlcassement* Créer... puis on réexécute l'import des tables et des données.

Finalement il ne nous reste plus qu'à vérifier la configuration de note site joomla sur le fichier configuration.php. Ce fichier regroupe toutes les informations de configuration générale utilisé par le framework joomla. Il faudra notamment changer :

```
var $log_path = 'C:\\Developpement\\My Webs\\Travail_Diplome_site\\logs';
var $tmp_path = 'C:\\Developpement\\My Webs\\Travail_Diplome_site\\tmp';
```

6.3 COMPOSANT TOURISME SÉMANTIQUE

Mon application se divise en 8 pages Web et cherche à démontrer, dans une certaine mesure, les liaisons qui ont été établies entre toutes mes données et des données externes ainsi que la possibilité de déduire certaines préférences pour un utilisateur enregistré.

Query Builder

Cette page correspond à mon premier développement sur ce site. L'objectif étant de fournir un moyen de tester les services REST avec des requêtes existantes.
Le *query builder* permet de charger un fichier SPARQL et de l'afficher dans l'éditeur ou simplement d'éditer manuellement une requête.

Pour l'affichage des données, je donne la possibilité d'afficher les solutions sous trois formes :

- XML : affiche les données telles que renvoyées par le service, au format XML brute.
- SimpleXML : librairie introduit avec l'arrivée de PHP 5. Elle permet de lire ou modifier facilement des fichiers XML. Cette solution me permet d'afficher facilement mes données grâce à sa méthode asXml().
- DOM : il s'agit d'un API qui permet de construire en mémoire un arbre représentant le document XML. Cette API standardisée par le W3C permettant, entre autres, de manipuler des documents, de créer des nœuds, les déplacer, ajouter des attributs ou des fils et finalement de détruire des sous-arbres.

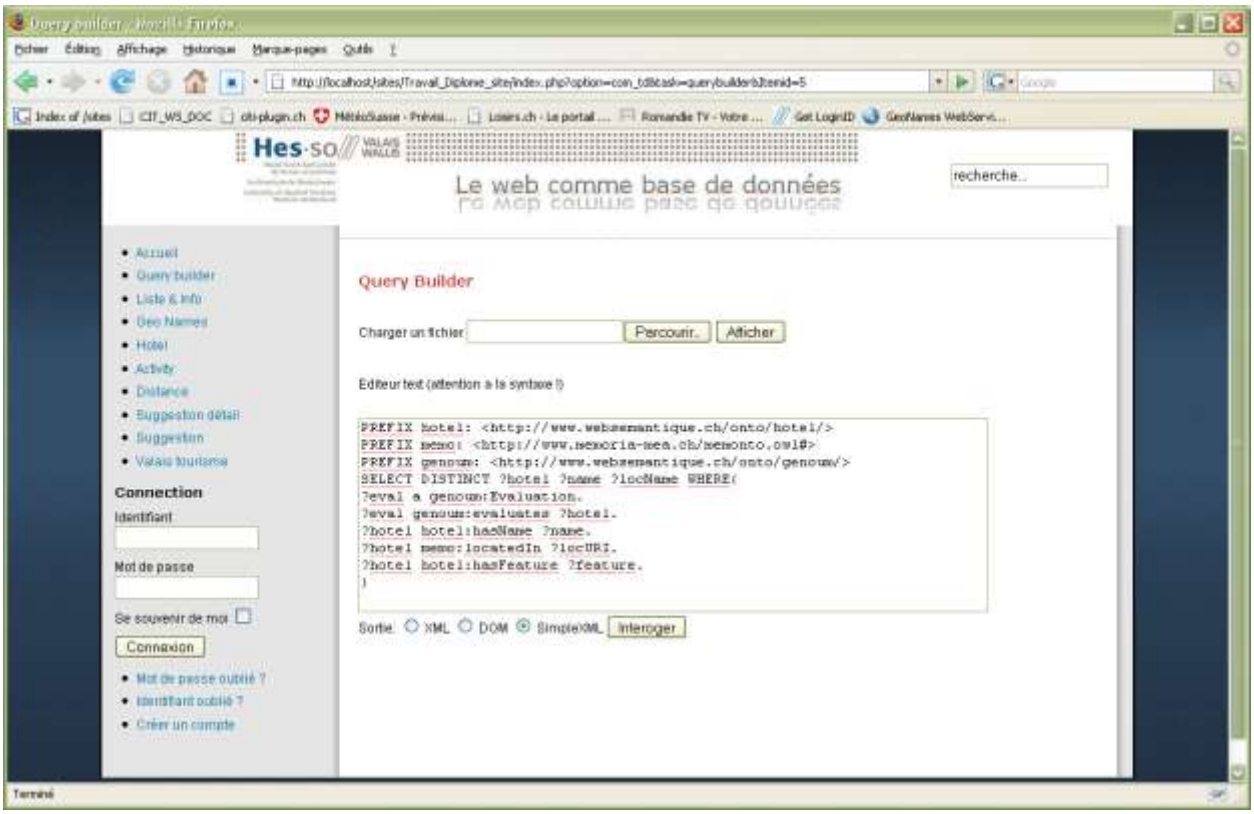
```
$document = new DomDocument();
$document->loadXML($result);
$racine = $document->documentElement;
$results = $document->getElementsByTagName('result');
foreach($results as $node){
        if($node->hasChildNodes()){
                foreach($node->childNodes as $childnode){
                        if($childnode->nodeType == XML_ELEMENT_NODE){
                                $myvalue = $childnode->nodeValue;
```

code tirée de function.php

L'interface DOM est assez exhaustive cependant, on peut noter que, bien que le code DOM est clair et facile à relire, il peut se révéler long et trop détaillé par rapport à *SimpleXML.* Deuxièmement, tout le document est interprété en une fois et chargé en mémoire, donc attention aux gros documents !

- Il faut naturellement aussi mentionner l'API SAX qui est un moteur événementiel dont le rôle est d'analyser un flux de données XML. Le moteur parcourt le flux à la recherche d'événements (balises ouvrantes, nœuds, …) et appelle en conséquence les fonctions utilisateurs qu'on y associe. Je ne suis intéressé à cette API car elle complexe et DOM me suffisant.

Sur le printscreen ci-dessous, j'ai arbitrairement choisi une requête assez intéressante. En effet, cette requête me permet de savoir le nombre d'hôtel qui se retrouvent dans mon export Tomas ET dans l'export de Fabian.
Si l'on observe le code, on voit que je précise les hôtels qui ont une évaluation (Tripadvisor.ch) ainsi que des caractéristiques (Tomas). Ainsi, les hôtels répondant à ces critères me permettent de prouver que j'ai bien **différents sources d'informations pour le même sujet**.



Liste & Info

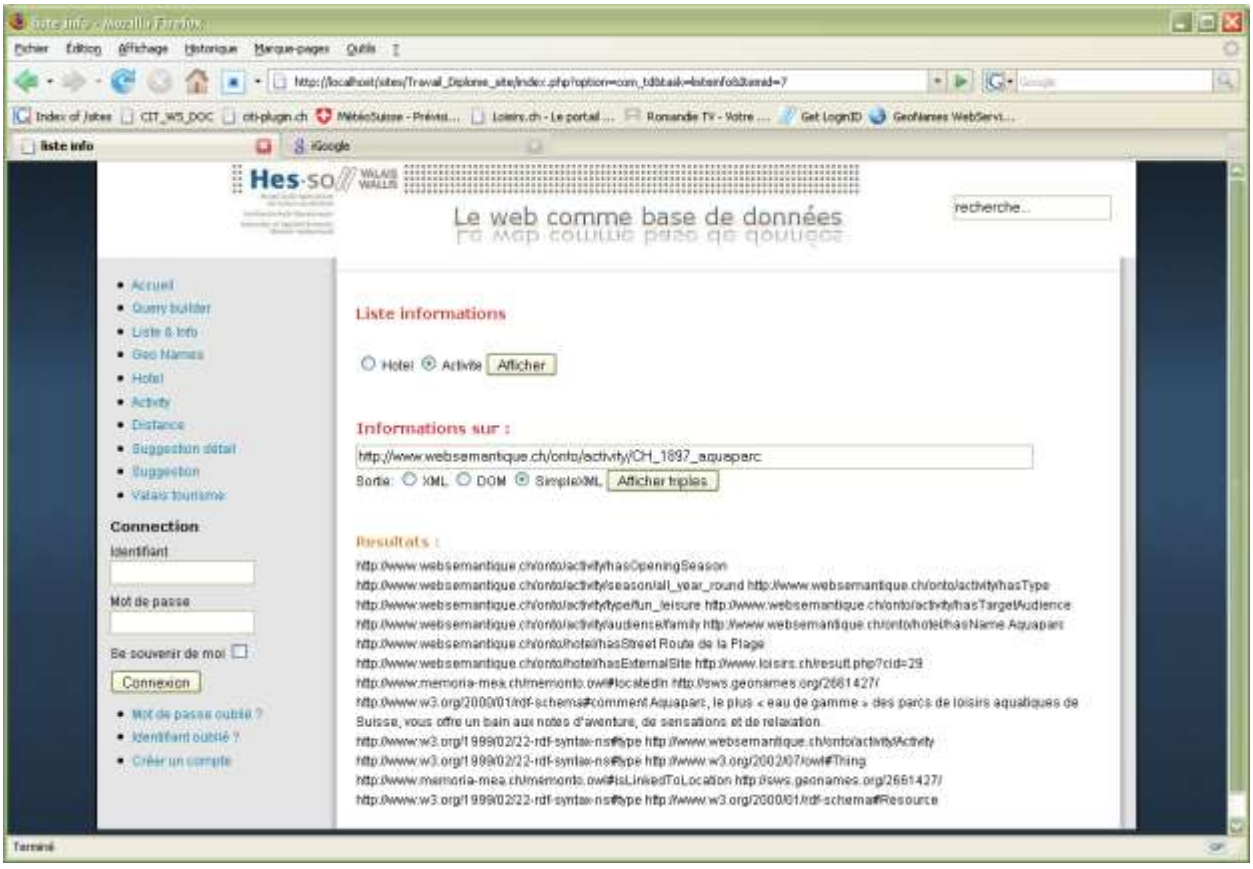
Cette page me permet de faire un listing complet de toutes les activités et de tous les hôtels stockés dans mon **triplestore**.

De plus elle dispose d'un formulaire qui permet d'obtenir toutes les informations sur un sujet générique. Ainsi, je peux savoir toutes les informations, écrites ou inférées en rapport à un hôtel ou une activité, mais aussi un lieu GeoNamess ou n'importe quoi d'autres.

Pour réaliser cette opération, j'utilise des requêtes SPARQL générique que je charge dans mon application, puis je remplace le sujet par le champ de mon formulaire.

```
SELECT * WHERE{
sujet ?p ?c }
```

generique_sujet.sparql



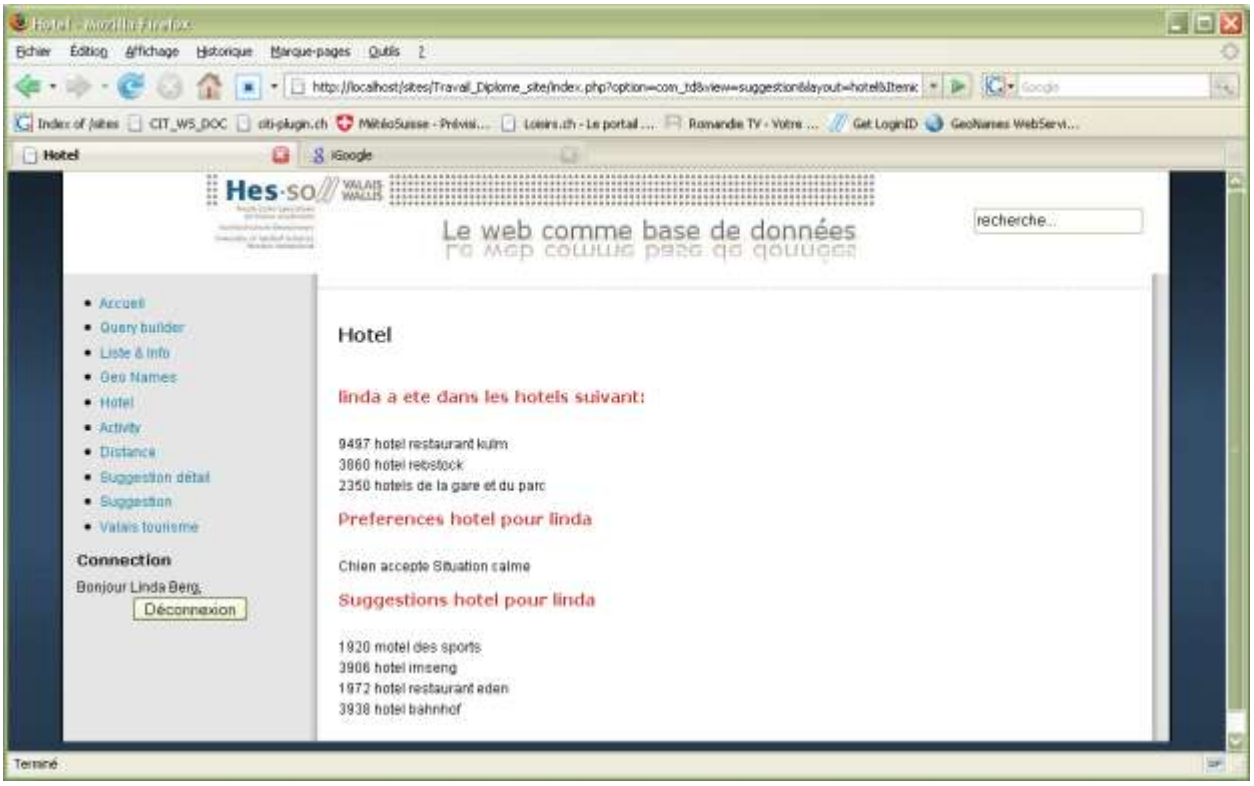
GeoNames

Cette page me permet simplement d'afficher sur le site de GeoNames la localisation d'une activité ou d'un hôtel. Il me suffit de récupérer les GeoNameID de mon sujet et de l'afficher dans un iframe.

Hotel et Activity

Ces deux pages me permettent de mettre en avant mes inférences. Donc il est nécessaire de se loguer dans l'application. Vous pouvez utiliser le compte *linda* password *linda*.

Ces pages se divisent en 3 parties, ce que je sais d'une personne, ce que je peux en déduire et ce que je peux lui suggérer.



Pour les informations sur une personne, je vais simplement lire les données qui me sont fournis dans le fichier touriste.rdf.

Pour les déductions, je m'appuie sur quelques règles d'inférences que j'ai écrites.
Exemple pour une situation calme :

```
@prefix tdprofile: <http://www.websemantique/TDprofile#>.
[situation_calme:
(?personne tdprofile:chooseSituationCentrale ?a),
(?personne tdprofile:sumHotel ?b) quotient(?a, ?b, ?c) greaterThan(?c, 0.8)
-> (?personne tdprofile:likeToHaveHotel
<http://www.websemantique.ch/onto/hotel/Situation_calme>)]
```

profile_hotel_situation_calme.rule

Explication : soit la rège situation calme ; si une personne à choisi plus de 80% des fois un hôtel avec une situation calme, c'est qu'elle l'a fait délibérément.

Cette règle me permet aussi de mentionner la simplicité de mes déductions. Il est évident que ce résultat pourrait très bien être le fruit de hasard.
De plus j'ai choisi des exemples qui allaient bien pour mon démonstrateur, mais dans une solution en production, il faudra encore pondérer l'importance de certains critères par rapport aux autres. Pour ma part, je ne fais que les accumuler. Il sera intéressant de lier ces données à un programme de User Modeling.

A noter aussi que je peux calculer de deux manières les hôtels et les activités semblable :
Je peux soit utiliser la propriété *hasSimilarActivity* avec mon moteur d'inférence.
Ou, je peux utiliser un FILTER(!BOUND(?personne)) dans mes requêtes SPARQL.
(Voir : les différents langages du Web sémantique -> SPARQL)

Distance

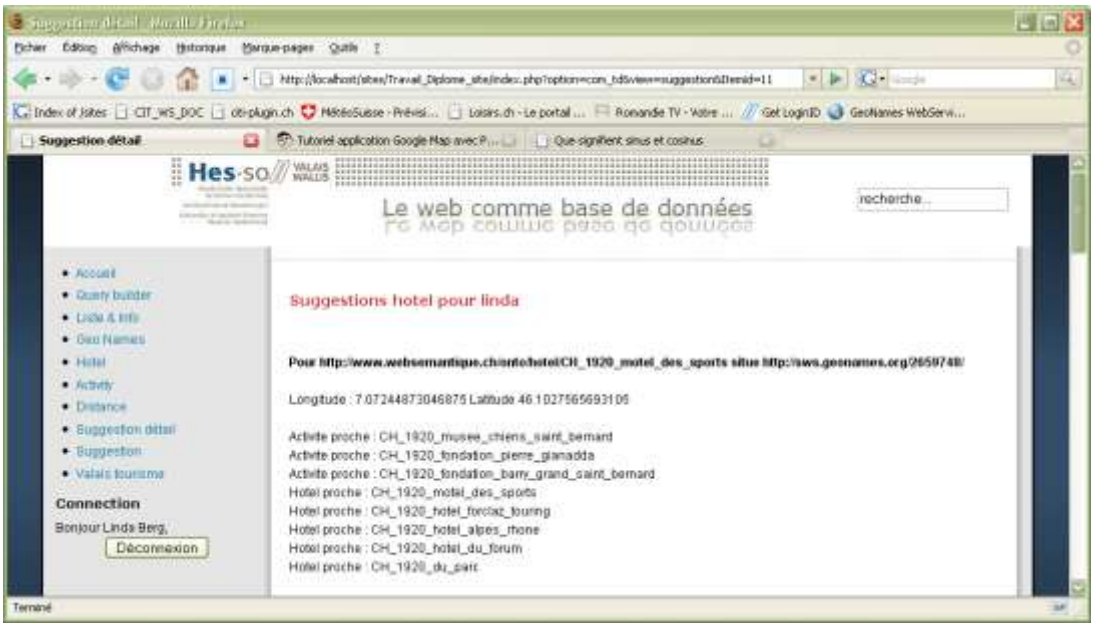
Cette page s'appuie sur une fonction trigonométrique que j'ai trouvé sur Internet et me permet de calcule la distance entre 2 points. Il suffit de saisir, sous forme de degré les coordonnées (latitudes, longitudes) pour obtenir la distance **en ligne droite** qui séparent ces deux points.

Encore une fois, il faudra implémenter un vrai service qui ne calcule non pas la distance en ligne droite, mais la distance en voiture surtout pour le Valais. Malheureusement, je n'ai pas trouvé un tel service sur Internet.
GoogleMap Itinéraire propose ce service, mais il m'est impossible d'intégrer cette fonctionnalité dans mon application, car le traitement se fait chez Google et il n'existe pas d'API à ma connaissance pour appeler cette fonction. Je parle de *GoogleMapAPI.class.php*.
(Voir : http://www.phpinsider.com/php/code/GoogleMapAPI/)

Suggestion détail

Cette page reprend les informations que j'ai suggérer à mon utilisateur et présente certaines informations par rapport à sa localisation.
Les informations proviennent de GeoNames et du raisonneur. Je peux ainsi afficher les coordonnées (pour le calcul de distance), les hôtels et activités qui sont dans le même endroit et afficher, s'il existe, le lien Wikipedia et Dbpedia.

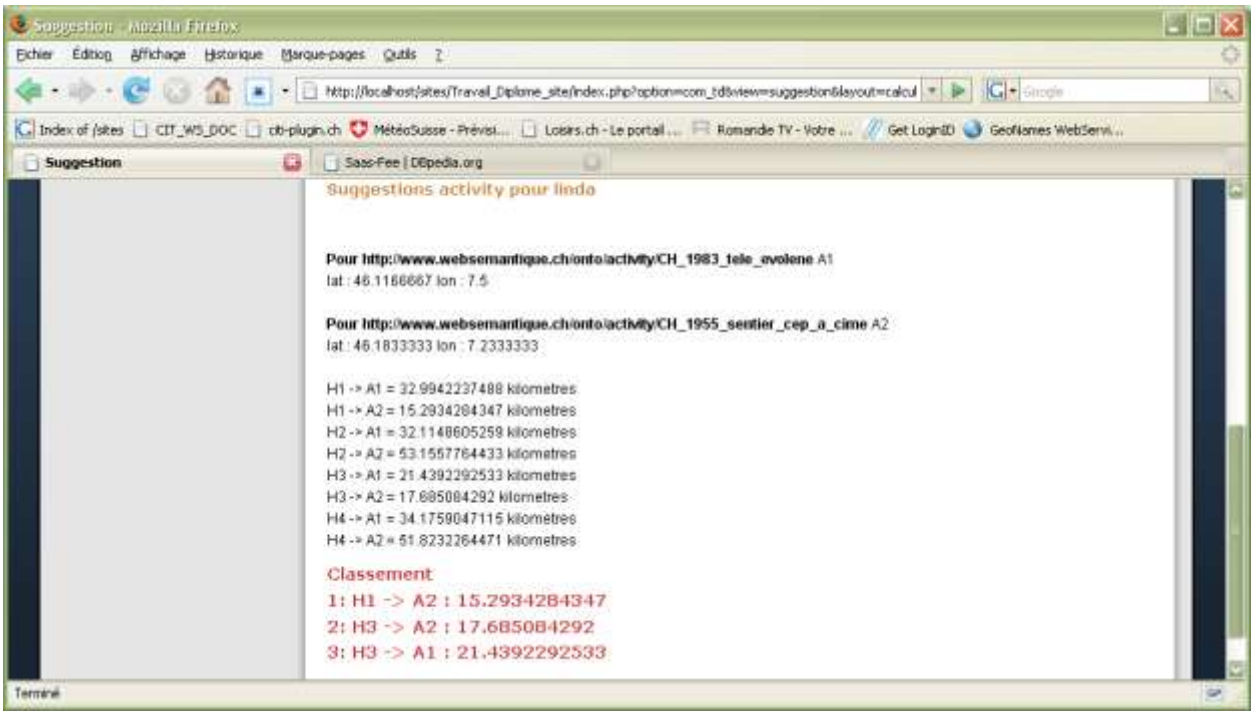
On pourrait imaginer exploiter les données sémantiques de Dbpedia pour présenter certaines données, par exemple les photos référencées…



Suggestion

Cette page regroupe toutes les suggestions faites pour une personne et va calculer, pour chaque pair *hotel->activity,* la distance les séparant.
Elle propose ensuite un classement en fonction de la distance minimal.

Vous pouvez tester cette fonctionnalité en désactivé dans OntoMea une règle de profil (exemple : *profile_hotel_animaux_autorises.rule*), et ainsi ouvrir l'éventail de possibilités suggérées.
Un nouveau classement sera alors établi.



7. CONCLUSION

Demain, le Web Sémantique serait peut-être un regroupement de données très diverses, en provenance de sources toutes aussi éparses. A la différence que toutes ces informations seraient utilisables de manière complète. Nous pourrons alors rêver de moteur de recherche extrêmement pointu capable de répondre à des demandes complexes et ainsi, la vision de Tim-Berners Lee aura pris jour.

Mais ce scénario soulève de nombreuses questions, notamment la confiance que l'on place dans une source d'information. Car si l'on se base sur les métadonnées d'une page et non sur son contenu lui-même, rien ne prouve que l'on trouvera vraiment une explication satisfaisante sur la page elle-même. C'est pourtant le parie que fait Yahoo qui projette d'utiliser les marqueurs sémantiques pour examiner la signification réelle des données sur une page Web.

De plus, l'une des idées fondamentales du Web Sémantique est de pouvoir suggérer des modifications par le biais du navigateur, afin de permettre à tous de corriger les erreurs trouvées. Cette idée pourra donc garantir la validité des informations…

Alors, le Web Sémantique révolutionnera-t'il la recherche d'information ?

Les avantages que cette technologie peut apporter dans de nombreux domaines sont indéniables, et certains obstacles humains peuvent être contournés. Les outils permettant de faciliter le travail de 'sémantisation' et d'exploitation de tels documents se multiplient. Et de nombreux projets nous permettent déjà d'ajouter du sens aux documents d'une machine (Nepomuk KDE) ou d'une entreprise (Webfountain).

Malheureusement, comme c'est souvent le cas avec les nouvelles technologies, le problème majeur demeure le processus d'adoption. C'était déjà le cas avec XML ou AJAX et ça le sera certainement aussi pour le Web Sémantique.

Je tiens à remercier Anne Le Calvé pour son soutient tout au long du projet ainsi que Fabian Cretton pour sa disponibilité et ses compétences en la matière.

BIBLIOGRAPHIE

RDF OWL

Introduction à OWL
http://www.lespetitescases.net/definir-une-ontologie-avec-owl
Documentation OWL
http://www.w3.org/TR/owl-features/
http://www.yoyodesign.org/doc/w3c/owl-features-20040210/
Protoégé
http://protege.stanford.edu/
Documentation RDF
http://xmlfr.org/documentations/tutoriels/041015-0001

XML XSD

Documentation,
http://xmlfr.org/w3c/TR/xslt/#section-Creating-Processing-Instructions
Tutoriaux
http://www.w3schools.com/xsl/
http://www.gchagnon.fr/cours/xml/recapxsl.html
Fonctions sur les strings
http://articles.techrepublic.com.com/5100-10878_11-1054414.html

SPARQL

Documentation et tutoriaux
http://www.w3.org/TR/rdf-sparql-query/
http://www.yoyodesign.org/doc/w3c/rdf-sparql-protocol/
Tutorial ARC
http://jena.sourceforge.net/ARQ/Tutorial/

PHP

http://www.php.net/

Livre : PHP 5 avancée 3ème édition, aux édition Eyrolles

RULE

Documentation Jena
http://jena.sourceforge.net/inference/
SWRL documentation
http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab
Le forum jena-dev, merci à Dave Reynolds de chez Hewlett-Packard Limited
http://tech.groups.yahoo.com/group/jena-dev/

Concept Linked Data

http://wiki.dbpedia.org/Interlinking
http://en.wikipedia.org/wiki/Linked_Data

Idées intéressantes

Du blog de Frédérick Giasson
http://fgiasson.com/blog/index.php/2007/06/04/my-personal-library-and-the-semantic-web/
http://fgiasson.com/blog/index.php/2008/01/20/data-referencing-data-mobility-and-the-semantic-web/
Vue globale du sujet
http://www.clever-age.com/veille/clever-link/le-web-semantique-en-entreprise-comment-et-a-quels-niveaux.html

Source des données

Hôtels
http://www.hotel-valais.ch/hotel/valais/suisse/liste-hotels.html
http://www.tripadvisor.fr/
Activités
http://www.loisirs.ch/
http://www.tourismesuisse.com/valais/activite/f_a_1_0_0-tourisme.html
Données sémantiques externes
http://www.geonames.org/
http://fr.wikipedia.org/wiki/Accueil et http://dbpedia.org/About

GLOSSAIRE

Axe (XSLT):

Défini le sens de la relation entre le nœud courant et le jeu de nœuds à localiser.

Bot :

Agent logiciel automatique qui interagit avec des serveurs informatiques. On les utilise principalement pour effectuer des tâches répétitives que l'automatisation permet d'effectuer rapidement.

Classe ou **concept** :

Permet de définit un objet dans une ontologie.

Graphe :

Ensemble de points, dont certaines paires sont reliées par des lignes. Les points sont appelés sommets et les lignes sont nommées arêtes. On peut représenter un triplet sous la forme d'un graphe : sujet-prédicat-objet

Inférence :

« L'inférence est une opération mentale qui consiste à tirer une conclusion (d'une série de propositions reconnues pour vraies). Ces conclusions sont tirées à partir de règles de base. » *(Source Wikipedia)*

Ontologie ou **schéma**:

Une ontologie permet de structurer un ensemble d'objets afin de leur donner du sens. Elle définit l'ensemble des rapports possibles entre ces objets et permettent

de créer certaines restrictions sur ces liens ainsi elle précise des termes et des relations d'un sujet précis.

OntoMea :

Logicel développé par la Hevs dans le cadre du projet Memoria. OntoMea dispose d'un triple store, d'un raisonneur et fournit des Webs services pour son interrogation externe.

OWL :

« Web Ontology Language est un dialecte XML basé sur une syntaxe RDF. Il fournit les moyens pour définir des ontologies Web structurées. » *(Source Wikipedia)*

Prédicats (XSLT):

Test qui qui permet d'affiner la recherche sur le jeu de nœuds à récupérer.

Protégé :

Logicel open-source permettant de créer graphiquement des ontologies OWL.

RDF :

Resource Description Framework est un modèle de graphe destiné à décrire les ressources Web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions. Développé par le W3C, RDF est le langage de base du Web sémantique. Il permet donc de définir des métas-données afin de préciser les caractéristiques d'une information.

RDFS :

« RDF Schema ou RDFS est un langage extensible de représentation des connaissances. Il appartient à la famille des langages du Web sémantique publiés

par le W3C. RDFS fournit des éléments de base pour la définition d'ontologies ou vocabulaires destinés à structurer des ressources RDF. » *(Source Wikipedia)*

Triplets :

Propositions de type sujet-verbe-complément.

Triple store :

Système de stockage des données ayant pour structure un graphe de triplets.

XML :

Langage de balisage qui permet de créer des structurations de données spécialisés.

Xpath :

Language pour identifier, sélectionner et manipuler les noeuds dans une structure XML.

XSD :

Fichier au format XML qui définit de façon structurée le type de contenu et la syntaxe d'un document XML.

XSLT :

Langage de transformation procédurale appliqué à un fichier XML.

Web sémantique:

« Il désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wide Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles, utilisant notamment la famille de langages développés par le W3C. » (Source: Wikipedia)

DÉCLARATION SUR L'HONNEUR

Je déclare, par ce document, que j'ai effectué le travail de diplôme ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de diplôme, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après : Anne Le Calvé

LISTE DES ANNEXES

Annexe 1 : cahier des charges
Annexe 2 : planification

CAHIER DES CHARGES

SUJET DU TRAVAIL

Le but de ce travail de diplôme est de démontrer l'intérêt des nouvelles technologies du Web 3.0 dit Web sémantique dans l'application du tourisme en Valais. L'objectif étant de montrer aux acteurs du monde du tourisme, que cette sémantisation du Web, et plus particulièrement des informations descriptives de leur régions devraient être mis-à-jour pour permettre la création de nouveau outils de recherche et de planification de vacances avec une réelle plus value par rapport aux offres traditionnelles.

CONDITIONS CADRES

Web 3.0 ou Web sémantique

« Le Web sémantique désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wide Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles, utilisant notamment la famille de langages développés par le W3C. »
(Source : http://fr.wikipedia.org/wiki/Web_s%C3%A9mantique*)*

Le Web comme base de données

Pour ce travail de diplôme, nous allons principalement nous servir de données provenant du site officiel du tourisme valaisan:
http://www.valaistourisme.ch/fr/welcome.cfm

Ce site est très complet, et les informations qui proposent se prêtent bien à un travail de sémantisation. De plus valaistourisme.ch a accepté de travailler avec nous et nous a notamment fournit un 'dump' d'une petite partie de leur donnée.

Ontologie

Une ontologie permet de structurer un ensemble d'objets afin de leur donner du sens. Elle définit l'ensemble des rapports possibles entre ces objets et permettent de créer certaines restrictions sur ces liens.
Pour démontrer l'intérêt du Web 3.0, je pense créer ou utiliser une/des ontologies relative au monde du tourisme. Il s'agit de mettre en avant la possibilité de produire des déductions intelligentes à partir informations disponibles sur le site du tourisme valaisan ainsi que d'autres sources de données qui seront à définir ou à simuler.

PROBLÉMATIQUE

Récupérer les données du Web

Pour ce travail de diplôme nous allons utiliser internet comme source de données. Or internet est actuellement une immense bibliothèque sans classement ni bibliothécaire. Alors le premier problème consiste à récupérer cette masse d'informations depuis internet ?
Une fois que tout cet amas de page aura été collecté, il s'agit encore d'en extraire une information utile. Donc mon deuxième problème sera comment parcourir toutes ces pages HTML et se focaliser sur l'information utile?

Gestion des données

Une fois les données rapatriées du Web, il faudra encore les stocker. Il s'agit ici de manipuler toutes ces données brutes au format texte pour les stocker au format RDF dans une base de données performante.

Mettre à disposition les données

Comment présenter ces données à l'utilisateur et faire ressortir de manière transparente la plus-value du Web sémantique ? Cette partie est la plus créative de mon travail ; il s'agit d'imaginer ce que l'on pourrait déduire de nouveau par rapport à ce que l'on sait déjà.

TRAVAIL À RÉALISER

Récupérer les données du Web

Pour récupérer toutes ces données, qui pourront se présenter soit forme de page HTML, soit sous forme de base de données ou de fichier XML, il s'agit de tester et d'implémenter différentes techniques.
Une partie de mon travail consiste à explorer différentes méthode, outils et langages pour réaliser cette collecte d'information.

Créer et gérer les ontologies

Il va falloir créer ou utiliser une/des ontologies du tourisme propre à notre site (ontologie de domaine). Il existe déjà beaucoup d'ontologie relative aux voyages et tourisme, mais la nature même des informations est très ciblée. Et il me semble vraiment intéressant de créer une telle ontologie, susceptible d'être améliorer en cours du projet.

Stocker les données

Pour stocker ces données, différente solutions existent déjà. Mais comme tout ce travail à déjà été réalisé dans d'autres travaux et que nous avons déjà à disposition une base de données qui gère les triples stores avec un moteur d'inférences, nous allons utiliser OntoMea.

OntoMea est un programme écrit en partenariat avec la HEVS de Sierre qui va nous permettre :

- de stocker les données au format RDF dans un triple store
- d'implémenter des règles d'inférence sur ces données pour ajouter une plus-value à mon travail
- finalement, il fournit aussi un SPARQL end-point susceptible d'être manipuler depuis une API externe pour interroger nos données à distance.

Présentation des données

Je pensais créer une interface en Java ou un site en PHP qui mettrait en évidence :

- Déduire les intérêts des utilisateurs en fonctions de ces choix. Par exemple, en connaissant les types d'activité qu'une personne a déjà réalisés, on peut en déduire ses goûts et lui suggérer automatiquement une liste d'activités et même certaines auxquelles il n'aurait pas pensé lui-même.
- Toujours sur le même principe, en connaissant le type d'hôtel qu'un client a déjà fréquenté, on peut en déduire ses besoins et lui suggérer un hôtel qui va implicitement répondre à ces attentes alors même que l'utilisateur n'aura rien spécifier.
- Donnée un lien pour accéder aux commentaires d'autres utilisateurs via des informations de voyages comme TriAdvisor.

Liaison de données externes

Il serait aussi possible dans une deuxième phase du projet de lier :

- Les informations géographiques avec les données de http://www.geonames.org
- Les informations sur les personnes avec http://www.foaf-project.org/
- Les informations sur les stations et les monuments avec http://www.freebase.com/ ou http://dbpedia.com

PLANIFICATION

Phase de recherche et de compréhension des technologies. Cette partie a pour but de se familiariser avec le Web sémantique (RDF, OWL, Ontologie, SPARQL).
Dans le même temps, je compte le temps passé à tester différents produits *(Voir état de l'art)* ainsi que le temps passé explorer différentes solutions pour rapatrier des données d'Internet *(Voir phase de collecte d'informations)*
Durée : 10 jours.

Une phase de réflexion et de modélisation. Il s'agit de définir les objectifs à atteindre en tenant compte, des technologies disponibles ainsi que des données disponibles pour imaginer un modèle de réflexion qui mettent en avant les Web sémantique. Je compte aussi le temps passé à me familiariser avec OntoMea et les exemples (RDF, OWL, RULE) qui me sont fournis dans la solution.
Durée 10 jours.

Une phase de développement durant laquelle :

- Les ontologies nécessaires à ce projet vont être modélisées et installées.
- La transformation des données XML via un script XSLT va être réalisée. Je compte le temps passé sur l'étude du langage XML ainsi que XSLT.
- Les règles d'inférences vont être écrites afin de déduire une information utile.
- Une application de démonstration en PHP va être réalisée.

Durée 30 jours.

Une phase de tests et de débogage de mon application PHP.
Durée 2 jours.

Une phase de documentation sur les technologies et les langages appris ainsi que tous les autres aspects de rédaction du rapport final.
Durée 8 jours.

(Pour le détail de la planification, voir le fichier Planning.xls)