Un navigateur pour les données liées du Web

Olivier Corby, Catherine Faron Zucker

Un navigateur pour les données liées du Web

Olivier Corby[1], Catherine Faron Zucker[2]

[1] INRIA
olivier.corby@inria.fr

[2] UNIVERSITÉ NICE SOPHIA ANTIPOLIS
faron@unice.fr

Résumé : Nous présentons une plate-forme pour la conception de navigateurs permettant une navigation hypertexte dans des graphes RDF locaux ou sur le Web et présentant ces données RDF en HTML. Cette plate-forme, nommée ALIGATOR (A LInked data naviGATOR), est basée sur le langage de transformation de graphe RDF STTL (SPARQL Template Transformation Language), conçu comme une extension du langage SPARQL. ALIGATOR se présente sous la forme d'un serveur HTTP embarquant, outre un service SPARQL, un moteur de transformation et un service Web permettant d'exécuter des transformations. Nous montrons les capacités du système en présentant trois navigateurs : un premier pour exécuter des requêtes SPARQL sur un graphe RDF local ou sur le Web et présenter les résultat en HTML, un second pour naviguer sur le graphe RDF de DBpedia avec des formats de présentation dédiés à certains types de ressources et un troisième pour présenter une vue unifiée d'un graphe local lié au graphe de DBpedia.

1 Introduction

Le Web de documents structurés qui repose sur le standard XML a rapidement été muni du standard XSLT pour engendrer des formats de présentation tels que HTML ou bien pour écrire des transformations de XML vers XML. De la même manière, le Web de données sémantiques qui repose sur le standard RDF a maintenant besoin d'un langage de transformation pour présenter les données RDF dans des formats lisibles tels que HTML et pour écrire des transformations de RDF vers RDF.

La transformation et la présentation de données RDF est encore un problème ouvert. Nous avons proposé dans (Corby & Faron-Zucker, 2014) et (Corby & Faron-Zucker, 2015) un langage de règles de transformation pour RDF, *SPARQL Template Transformation Language* (STTL), dont le format de sortie est textuel — sans restriction a priori sur le format de texte. L'état de l'art sur les autres solutions existantes pour transformer des données RDF présenté dans les articles ci-dessus mentionnés montre qu'elles sont toutes liées à la syntaxe XML ou à un format spécifique de sortie, ou aux deux, sauf Fresnel (Bizer *et al.*, 2005). Mais Fresnel s'intéresse à la présentation des données RDF et ne traite pas du problème plus général de leur transformation. STTL permet une approche générique pour écrire des transformations RDF vers différents formats de sortie.

Nous nous intéressons dans cet article plus particulièrement à la transformation de données RDF en HTML et nous présentons une plate-forme basée sur le langage STTL, nommée ALIGATOR (acronyme de *A LInked data naviGATOR*), et permettant de concevoir des navigateurs pour le web des données liées et de présenter ces données en HTML. ALIGATOR est constitué d'un serveur HTTP, un service REST, un moteur de transformations STTL et une bibliothèque de transformations STTL. Le code HTML engendré par les transformations contient des liens hypertextes vers le serveur ce qui permet d'offrir une navigation hypertexte sur des graphes RDF.

Cet article est organisé comme suit : la section 2 présente brièvement le langage STTL, la section 3 présente le service REST appelant le moteur de transformations STTL. La section 4 présente différentes applications Web permettant de présenter et naviguer sur des données RDF.

2 Le langage de transformation STTL

Le langage STTL, que nous avons conçu, est une extension du langage *SPARQL 1.1 Query* avec une clause TEMPLATE permettant d'engendrer un résultat sous forme de texte à partir des solutions d'une clause WHERE. La clause TEMPLATE peut contenir du texte, des variables et des expressions. Les variables sont remplacées par les valeurs trouvées dans les solutions, les expressions sont évaluées et le tout est concaténé sous forme d'une chaîne de caractères qui est le résultat retourné par le template. L'exemple suivant montre la traduction d'un énoncé OWL exprimé dans la syntaxe RDF de OWL — une restriction de type "allValuesFrom" — dans la syntaxe fonctionnelle de OWL.

```
TEMPLATE {
  "allValuesFrom(" ?p " " ?c ")"
}
WHERE {
  ?in a owl:Restriction ;
    owl:onProperty ?p ;
    owl:allValuesFrom ?c
}
```

Une transformation STTL est un ensemble de templates dédiés à la transformation d'énoncés RDF dans un certain modèle (e.g. les données de DBpedia sur les rois de France) ou pour un certain langage (e.g. OWL/RDF) dans un format textuel (e.g. une présentation des rois de France en HTML ou une représentation d'énoncés OWL dans la syntaxe fonctionnelle du langage). L'exécution d'un template peut récursivement déclencher l'exécution d'autres templates de transformation ; l'appel à un template se fait au moyen d'une fonction d'extension `st:apply-templates` [1] appelée dans la clause TEMPLATE. Une variante de l'exemple précédent est le template suivant :

```
TEMPLATE {
   "allValuesFrom("
     st:apply-templates(?p) " "
     st:apply-templates(?c) ")"
}
WHERE {
  ?in a owl:Restriction ;
    owl:onProperty ?p ;
    owl:allValuesFrom ?c
}
```

1. Le préfixe `st:` correspond au namespace `http://ns.inria.fr/sparql-template/`

Dans ce template, l'appel de la fonction `st:apply-templates` avec la variable `?p` en paramètre déclenchera la sélection d'un template de transformation qui sera appliqué sur le résultat (binding) associé à la variable `?p` lors de l'appariement du graphe requête de la clause WHERE avec le graphe RDF à transformer. La même chose se produira pour la variable `?c`. Des templates nommés sont également prédéfinis (et d'autres peuvent être définis et nommés) qui peuvent être exécutés directement par un appel à la fonction d'extension `st:call-template`. Voici un template de transformation équivalent au précédent où les templates appelés dans la clause TEMPLATE sont indiqués explicitement : le template `st:property` dédié à la présentation d'une propriété dans la syntaxe fonctionnelle de OWL est appelé avec la variable `?p` en paramètre et le template `st:value` dédié à la présentation des valeurs de propriété est appelé avec la variable `?c` en paramètre.

```
TEMPLATE {
   "allValuesFrom("
     st:call-template(st:property, ?p) " "
     st:call-template(st:value, ?c) ")"
}
WHERE {
  ?in a owl:Restriction ;
    owl:onProperty ?p ;
    owl:allValuesFrom ?c
}
```

3 Le serveur ALIGATOR

Nous présentons dans cette partie la plate-forme ALIGATOR qui repose sur le langage STTL et permet de réaliser des navigateurs hypertextes pour le Web de données. Le code source est disponible dans la distribution de Corese-KGRAM [2] (Corby & Faron-Zucker, 2010) (Corby *et al.*, 2012). Un serveur de démonstration conçu avec ALIGATOR est disponible en ligne : `http://corese.inria.fr`.

Plus précisément, Corese intègre un serveur HTTP Jetty avec des services Web REST qui implémentent le protocole SPARQL 1.1 [3]. Un SPARQL endpoint Corese exécute des requêtes SPARQL convoyées par HTTP et retourne les résultats également par HTTP. Ces résultats sont exprimés dans les formats standards SPARQL Query Results XML Format, Turtle, RDF/XML ou JSON-LD.

Outre cette implémentation standard, ALIGATOR fournit un service Web qui permet d'exécuter des transformations STTL pour engendrer du code HTML à partir de données RDF. Nous appelons *RDF2HTML* ces transformations particulières. Ce service permet de réaliser des navigateurs hypertextes sur un graphe RDF local ou distant.

3.1 Le service STTL

Le service `STTL` répond à deux scénarios décrits sur la Figure 1. Dans le Scénario 1, une transformation RDF2HTML est exécutée sur un graphe RDF. Etant donné un URI, la transfor-

2. http://wimmics.inria.fr/corese
3. http://www.w3.org/TR/sparql11-protocol/

mation engendre une page HTML décrivant la ressource RDF correspondante. Le code HTML contient des liens hypertextes, vers le service STTL, pour décrire les ressources liées à cet URI. Ces liens hypertextes déclenchent un appel au service STTL qui applique la transformation et engendre en retour une nouvelle page HTML. On implémente ainsi une navigation hypertexte sur un graphe RDF. La transformation peut, par exemple, engendrer une table HTML avec les triplets dont l'URI est le sujet ou l'objet, comme le fait DBpedia [4].

Dans le Scénario 2, le service STTL exécute une requête SPARQL sur le graphe puis applique une transformation RDF2HTML sur le résultat de la requête. Pour les requêtes de type CONSTRUCT et DESCRIBE, la transformation est exécutée directement sur le graphe RDF résultat de la requête. Pour les requêtes SELECT et ASK, le résultat de la requête est traduit en graphe RDF en utilisant le vocabulaire publié par le W3C, RDF Data Access Working Group [5]. La transformation est ensuite appliquée sur le graphe produit.

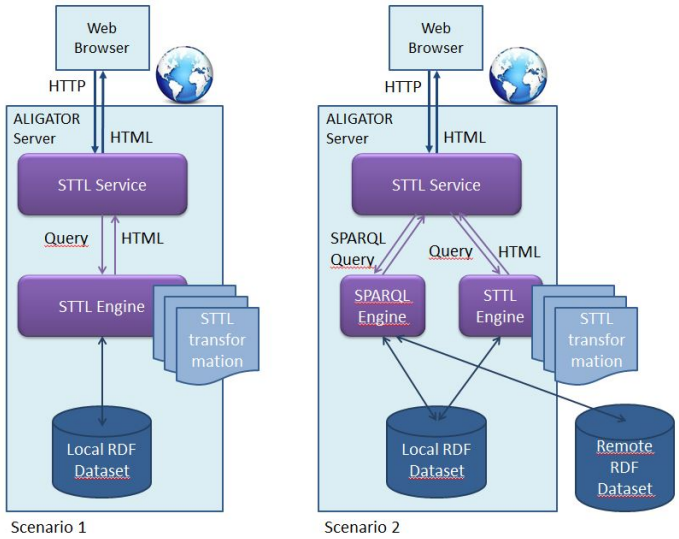

FIGURE 1 – Deux scénarios de transformation RDF2HTML. Scénario 1 : Transformation d'un graphe RDF. Scénario 2 : Transformation du résultat d'une requête SPARQL

Une requête pour une transformation STTL est envoyée à un service ALIGATOR dans un URL dont la partie hiérarchique se termine par `/template` et dont la partie requête contient des paires clé-valeur spécifiant la requête au serveur. La clé `query`, reprise du protocole SPARQL, permet d'indiquer une requête SPARQL à exécuter. La clé `transform` permet de spécifier l'URL de la transformation à appliquer sur le résultat de la requête ou sur le graphe du serveur. Par exemple, l'URL suivant demande l'exécution d'une transformation STTL spécifiée par `st:sparql` sur le résultat d'une requête SPARQL qui retourne tous les triplets d'un graphe RDF donné.

```
http://corese.inria.fr/template?
  query=SELECT * WHERE { ?x ?p ?y }&
  transform=st:sparql
```

4. e.g. http://dbpedia.org/resource/Berlin
5. http://www.w3.org/2001/sw/DataAccess/tests/result-set.n3

3.2 Profil de transformation

Un URL peut donc comporter une requête, une transformation ou les deux. Pour simplifier l'interaction avec le service STTL, nous introduisons la notion de *profil* de transformation RDF2HTML. Un *profil* permet de définir une chaîne de traitements simple, composée d'une requête SPARQL optionnelle et d'une transformation STTL, et de la nommer avec un URI. Un profil est décrit en RDF : la propriété `st:query` définit un chemin (un URL) vers une requête SPARQL et la propriété `st:transform` spécifie l'URL d'une transformation STTL. Voici un exemple de description de profil :

```
@prefix st: <http://ns.inria.fr/sparql-template/>
st:dbpedia a st:Profile ; st:query <q1.rq> ; st:transform st:navlab .
```

Dans cette description, `q1.rq` contient par exemple la requête SPARQL suivante qui interroge le serveur DBpedia :

```
CONSTRUCT { ?x ?p ?y }
WHERE { service <http://fr.dbpedia.org/sparql> { ?x ?p ?y } }
```

Dans l'URL transportant une requête pour une transformation STTL, l'URI d'un profil est indiqué par un argument `profile` dont la valeur est l'URI du profil :

```
http://corese.inria.fr/template?profile=st:dbpedia
```

Une requête peut préciser l'URI d'une ressource sur lequel focaliser l'exécution du service :

```
http://corese.inria.fr/template?profile=st:dbpedia
&uri=http://fr.dbpedia.org/resource/Antibes
```

La description de profil spécifie alors le nom d'une variable dont la valeur sera fixée dynamiquement dans la requête SPARQL comme étant la valeur de l'argument `uri` :

```
st:dbpedia a st:Profile ;
  st:query <q1.rq> ; st:variable "?x" ; st:transform st:navlab .
```

La requête SPARQL exécutée par le serveur est complétée par une clause `values` avec l'URI donné comme valeur de l'argument `uri`. Dans notre exemple, la requête que sera exécutée est donc la suivante :

```
CONSTRUCT { ?x ?p ?y }
WHERE { service <http://fr.dbpedia.org/sparql> {?x ?p ?y} }
VALUES ?x { <http://fr.dbpedia.org/resource/Antibes> }
```

La définition d'un profil de transformation permet d'interroger un serveur distant, DBpedia, sur une ressource particulière, la ville d'Antibes, puis d'appliquer au graphe résultat une transformation RDF2HTML qui engendre une page HTML décrivant la ressource. Le code HTML engendré peut contenir des liens hypertextes sur les autres ressources reliées à la ressource courante. Nous obtenons ainsi un navigateur hypertexte sur un graphe RDF distant (celui de DBpedia). Cette technologie permet donc d'engendrer un navigateur hypertexte sur un SPARQL endpoint. On peut aussi coupler l'interrogation d'un graphe local avec un SPARQL endpoint distant et réaliser ainsi un *mashup* de données liées.

3.3 Contexte de transformation

Nous avons défini la notion de contexte de transformation qui permet à un serveur STTL de transmettre au moteur de transformation STTL des informations relatives au contexte d'exécution de la transformation. Par exemple, pour engendrer des liens hypertextes, le moteur de transformation peut avoir besoin du nom du service. La spécification d'un contexte de transformation évite de "coder en dur" de telles informations dans la transformation, de manière à la rendre le plus générique possible.

Le contexte peut être consulté par le moteur de transformation au moyen d'une fonction d'extension SPARQL `st:get`. Les paramètres possibles du contexte sont les suivants : le nom du service, le nom du profil, le nom de la transformation, l'URI de la ressource courante. Voici un exemple de contexte :

```
st:get(st:service)   = template
st:get(st:profile)   = st:dbpedia
st:get(st:transform) = st:navlab
st:get(st:uri) = http://fr.dbpedia.org/resource/Antibes
```

3.4 Liens hypertextes dynamiques

Une des clés de notre approche et du système développé réside en la capacité à engendrer dynamiquement des liens hypertextes dans le code HTML produit. Lorsque l'un de ces liens est suivi, un nouvel appel au serveur est produit, pour engendrer de nouvelles pages HTML relatives à une nouvelle ressource (avec de nouveaux liens hypertextes), grâce à une nouvelle transformation RDF2HTML.

Voici un exemple d'un tel lien hypertexte ; l'attribut `href` de l'élément `a` a pour valeur un URL qui contient une requête au service STTL du serveur :

```
<a href='/template?profile=st:dbpedia
&uri=http://fr.dbpedia.org/resource/Antibes'>Antibes</a>
```

Voici un exemple de template qui engendre un lien hypertexte vers une ressource `?x` avec un titre `?t` :

```
TEMPLATE st:link(?x, ?t) {
  "<a href='/template?profile=st:dbpedia&uri=" str(?x) "'>"
  str(?t) "</a>"
}
WHERE { }
```

Pour éviter de coder en dur les information relatives au serveur, celles-ci peuvent être représentées dans le contexte :

```
TEMPLATE st:link(?x, ?t) {
   "<a href='/" st:get(st:service)
   "?profile="  st:get(st:profile)
   "&uri=" str(?x) "'>"
   str(?t) "</a>" }
WHERE { }
```

4 Trois navigateurs ALIGATOR

Plusieurs exemples de navigateurs construits avec ALIGATOR en réponse à différents besoins sont disponibles en ligne sur le serveur de démonstration `http://corese.inria.fr`. Nous présentons ici trois d'entre eux.

4.1 Navigateur pour un service SPARQL

Nous avons conçu un navigateur ALIGATOR permettant d'exécuter une requête SPARQL et d'en présenter le résultat en HTML. Le résultat d'une requête de la forme SELECT ou ASK est traduit en RDF en utilisant le vocabulaire W3C RDF Data Access Working Group [6]. Une transformation RDF2HTML est ensuite appliquée sur ce graphe. Nous avons défini le profil `st:sparql` pour identifier cette transformation. Pour les requêtes de la forme CONSTRUCT ou DESCRIBE, la transformation `st:sparql` est directement appliquée sur le graphe RDF résultat. Les règles de cette transformation sont disponibles en ligne [7].

La figure 2 est une capture d'écran de la transformation du profil `st:sparql` appliquée au résultat d'une requête SPARQL (cela est visible dans l'URI entrée dans le navigateur Chrome utilisé). Les liens hypertextes visibles sur la page HTML générée sont des liens vers le serveur ALIGATOR comme expliqué précédemment.

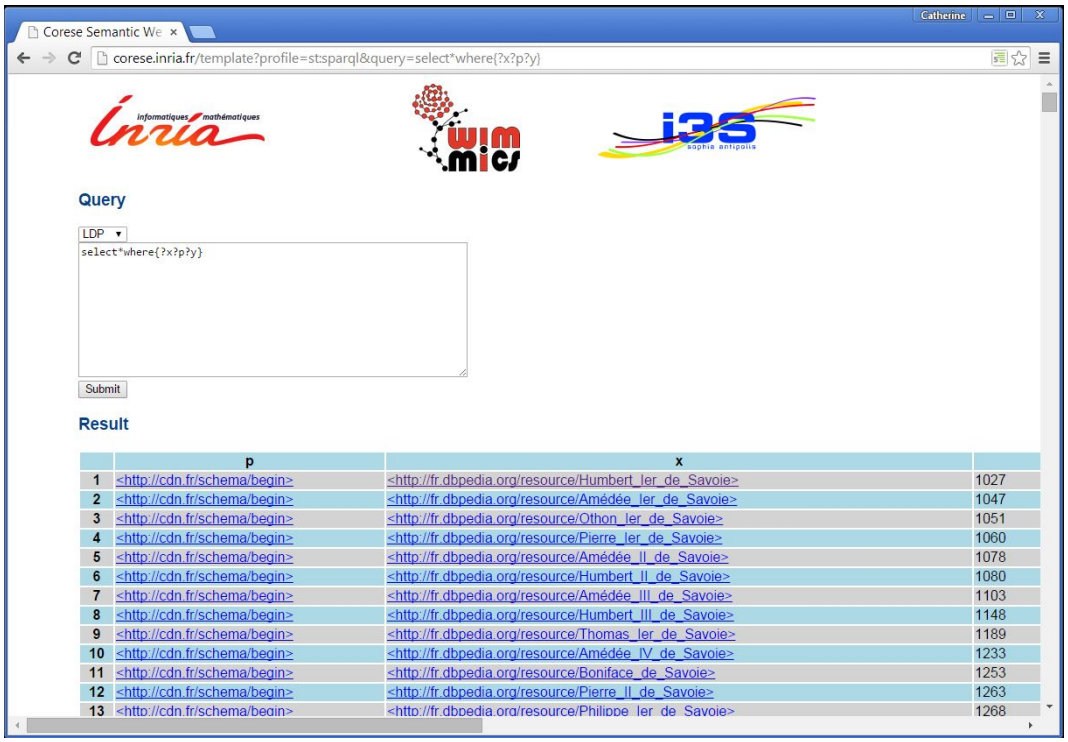

FIGURE 2 – Navigation dans les résultats d'une requête SPARQL

6. http://www.w3.org/2001/sw/DataAccess/tests/result-set.n3
7. http://ns.inria.fr/sparql-template/

Considérons par exemple la requête SPARQL : `SELECT ?x ?n WHERE ?x rdfs:label ?n` et l'exemple suivant de solution à cette requête, exprimée en RDF :

```
@prefix rs: <http://www.w3.org/2001/sw/DataAccess/tests/result-set#>
@prefix ex: <http://fr.dbpedia.org/resource/>

[] rs:resultVariable "x", "n" ;
   rs:solution
     [ rs:binding [ rs:variable "x" ; rs:value ex:Auguste ],
                  [ rs:variable "n" ; rs:value "Auguste"  ] ],
     [ rs:binding [ rs:variable "x" ; rs:value ex:Tibère  ],
                  [ rs:variable "n" ; rs:value "Tibère"   ] ].
```

Voici le template principal de la transformation `st:sparql` qui traite les résultats des requêtes de la forme SELECT :

```
prefix rs: <http://www.w3.org/2001/sw/DataAccess/tests/result-set#>
TEMPLATE {
  "<td>"
  coalesce(st:call-template(st:display, ?val), " ")
  "</td>" ; separator = " " }
WHERE {
  ?x rs:solution ?in
  ?x rs:resultVariable ?var
  OPTIONAL { ?in rs:binding [ rs:variable ?var ; rs:value ?val ] } }
ORDER BY ?var
```

La clause WHERE de ce template se focalise sur une solution `?in` qui est un ensemble de liaisons de variables. La clause OPTIONAL énumère ces liaisons de variables ; cette énumération est dans un sous-graphe optionnel car il se peut que certaines variables (`?val`) n'aient pas de valeur. La clause TEMPLATE engendre une cellule de table HTML pour chaque variable (`?val`) avec le résultat de la présentation de la valeur `st:call-template(st:display, ?val)`, ou bien un espace s'il n'y a pas de valeur disponible.

4.2 Navigateur DBpedia

Avec la même technologie, il est également possible de concevoir des navigateurs dédiés à des domaines ou des applications spécifiques. Nous avons ainsi développé un navigateur hypertexte dédié à certaines ressources de DBpedia : les personnes et les lieux. Il repose sur un serveur ALIGATOR offrant un service de transformation STTL avec une nouvelle transformation dédiée, dans le profil `st:navlab`. Son principe de fonctionnement est le suivant. Une requête SPARQL de la forme CONSTRUCT interroge le graphe distant DBpedia, avec une clause SERVICE, sur une personne ou un lieu et retourne un graphe RDF résultat. La transformation `st:navlab` est ensuite appliquée sur ce graphe résultat. Elle engendre une page HTML dédiée à la présentation des ressources retournées par la requête SPARQL. Les lieux sont géolocalisés sur une carte interactive. La figure 3 est une capture d'écran d'une page HTML générée par le navigateur ALIGATOR pour DBpedia. L'URI entrée dans le navigateur Chrome utilisé est

un exemple de requête HTTP envoyée au serveur ALIGATOR. Celui-ci produit en réponse une page HTML engendrée dynamiquement avec le profil `st:dbpedia` qui utilise la transformation `st:navlab`.

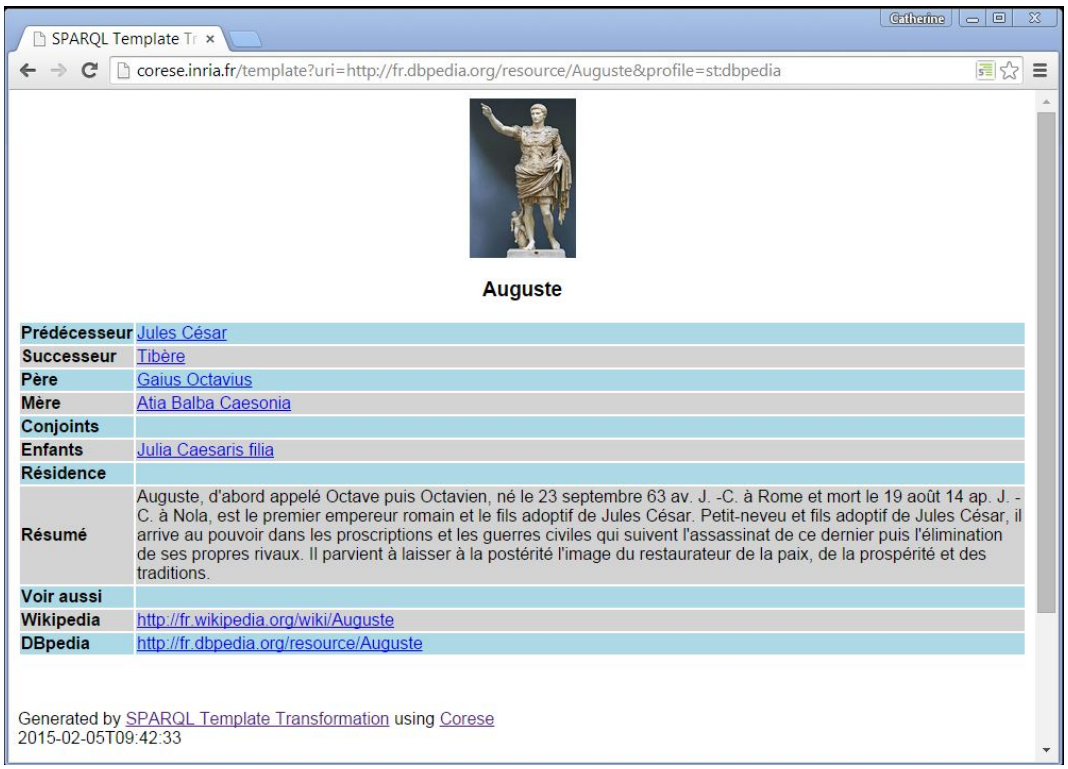

FIGURE 3 – Navigateur DBpedia

4.3 Navigateur Historique

Nous avons développé un troisième type de navigateur qui permet de présenter les données issues d'un graphe local lié à un graphe distant, selon le principe du Web de données liées. La source distante est encore une fois DBpedia ; le graphe RDF local contient un ensemble d'événements et de personnages historiques dont les URI sont ceux de DBpedia [8]. Les énoncés RDF locaux sont stockés dans des graphes nommés annotés avec des thèmes tels que la France, l'Empire, etc.

La transformation RDF2HTML que nous avons développée engendre une page HTML par siècle. Dans chaque siècle, les ressources sont classées par ordre chronologique et rangées dans les colonnes d'une table en fonction du thème de leur graphe nommé. Par exemple, une colonne "France" pour les descriptions dans le graphe annoté par le thème "France". Des liens hypertextes vers les ressources correspondantes de DBpedia sont engendrés selon le même principe que dans la section précédente (avec la transformation `st:navlab`). La figure 4 est une copie d'écran d'une page HTML engendrée par ce navigateur historique.

8. `http://fr.dbpedia.org/resource/`

FIGURE 4 – Navigateur historique

Le template suivant joue un rôle essentiel dans la transformation `st:navlab`. Sa clause WHERE retourne les dates (`?d`) comprises dans un intervalle (par exemple un siècle), triées par ordre chronologique. Sa clause TEMPLATE permet d'engendrer une ligne de table HTML pour chaque date et une cellule de table pour chaque thème dans laquelle sont affichées les éventuelles ressources correspondant à la date et au thème.

```
prefix cn: <http://cdn.fr/schema/>
TEMPLATE  cn:table(?min, ?max) {
  "<tr>"
  "<th class='date'>" st:call-template(cn:wikidate, ?d) "</th>"
  "<td>" st:call-template(cn:date, ?d, cn:fr)      "</td>"
  "<td>" st:call-template(cn:date, ?d, cn:emp)     "</td>"
  "<td>" st:call-template(cn:date, ?d, cn:mds)     "</td>"
  "<td>" st:call-template(cn:date, ?d, cn:context) "</td>"
  "<td>" st:call-template(cn:date, ?d, cn:cdn)     "</td>"
  "</tr>\n" }
WHERE {
  { SELECT DISTINCT ?d  WHERE { ?uri cn:date ?d }}
  FILTER(?min <= ?d && ?d <= ?max)
}
ORDER BY asc(?d)
```

5 Conclusion

Nous avons présenté la plate-forme ALIGATOR permettant de concevoir des navigateurs pour les données liées du Web sémantique. Elle repose sur le langage STTL qui est une extension de SPARQL permettant d'écrire des transformations déclaratives de RDF vers des formats textuels, en particulier ici vers HTML. Le moteur de transformation STTL et le serveur STTL qui constituent le navigateur sont disponibles (en open source) dans la plate-forme Corese. Plusieurs applications de cette technologie sont disponibles en ligne sur un serveur de démonstration à l'adresse `http://corese.inria.fr`.

Un avantage de cette approche, d'un point de vue technique, est qu'elle ne nécessite pas d'apprendre un nouveau framework Web. Il suffit de connaître SPARQL et HTML (et éventuellement JavaScript). STTL permet ainsi de passer "directement" de RDF à HTML avec SPARQL, sans langage de programmation.

Du point de vue de l'ingénierie des connaissances, les avantages de cette approche sont multiples. Tout d'abord, le fait de reposer sur le langage SPARQL est un atout de ce point de vue-là : avec une nouvelle forme de requête dont la clause WHERE est commune aux autres formes de requêtes SPARQL, STTL permet d'envisager de réutiliser des patrons de toutes formes de requêtes lorsque celles-ci sont capitalisées. Egalement, comme pour l'écriture de requêtes SPARQL en général, les patrons de conception mis en œuvre dans la construction des bases RDF sur lesquelles opèrera une transformation STTL peuvent être réutilisés pour écrire les templates qui composent celle-ci. Enfin, la déclarativité du langage STTL permet de capturer les connaissances expertes nécessaires pour opérer des transformations sur des données RDF. Les transformations STTL peuvent être vues comme des connaissances de *présentation* capitalisables, partageables, réutilisables dans des scénarios ou selon des points de vue sur les données similaires.

Dans la continuité de ces conclusions, nous envisageons d'une part d'exploiter les fonctions de HTML 5 couplées avec JavaScript ainsi que d'engendrer des vues graphiques avec des librairie dédiées (e.g. 3D.js). Nous projetons d'autre part d'explorer plus avant la capitalisation de patrons de requêtes SPARQL et patrons HTML et le couplage de ces deux types de patrons.

Remerciements

Nous remercions Eric Toguem (U. de Yaoundé, Cameroun) et Alban Gaignard (CNRS) pour la première version du serveur HTTP qui embarque Corese ainsi que Fuqi Song pour le déploiement du serveur de démonstration `corese.inria.fr` .

Références

BIZER C., LEE R. & PIETRIGA E. (2005). Fresnel - A Browser-Independent Presentation Vocabulary for RDF. In *Second International Workshop on Interaction Design and the Semantic Web @ ISWC'05*, Galway, Ireland.

CORBY O. & FARON-ZUCKER C. (2010). The KGRAM Abstract Machine for Knowledge Graph Querying. In *IEEE/WIC/ACM International Conference on Web Intelligence*, Toronto, Canada.

CORBY O. & FARON-ZUCKER C. (2014). SPARQL Template : un langage de Pretty-Printing pour RDF. In *Proc. 25e Journées francophones d'Ingénierie des Connaissances*, Clermont-Ferrand.

CORBY O. & FARON-ZUCKER C. (2015). STTL: A SPARQL-based Transformation Language for RDF. In *Proc. 11th International Conference on Web Information Systems and Technologies, WEBIST 2015*, Lisbon, Portugal.

CORBY O., GAIGNARD A., FARON-ZUCKER C. & MONTAGNAT J. (2012). KGRAM Versatile Data Graphs Querying and Inference Engine. In *Proc. IEEE/WIC/ACM International Conference on Web Intelligence*, Macau, China.

HARRIS S. & SEABORNE A. (2013). *SPARQL 1.1 Query Language*. Recommendation, W3C. http://www.w3.org/TR/sparql11-query/.