Un navigateur pour le Web des données liées

Nicolas Chauvat, Fabien Amarger, Laurent Wouters

Un navigateur pour le Web des données liées

Nicolas Chauvat[1], Fabien Amarger[1], Laurent Wouters[2]

[1] LOGILAB, 104 boulevard Louis-Auguste Blanqui, 75013, Paris
prénom.nom@logilab.fr

[2] CÉNOTÉLIE, 91 rue du Faubourg Saint Honoré, 75008 Paris
lwouters@cenotelie.fr

Résumé : *Le web des documents et le web des données liées utilisent tous les deux les URL pour identifier les ressources et le protocole HTTP pour les échanger. Les navigateurs web actuels sont limités à l'affichage des documents HTML et, nativement, ils ne permettent pas l'affichage de données RDF. Nous proposons d'étendre les fonctionnalités des navigateurs pour parcourir le web dans son intégralité de la manière la plus intuitive possible, c'est-à-dire en suivant les liens entre ressources sans que leur format constitue un obstacle. Pour y parvenir, nous proposons une extension de navigateur permettant de visualiser les ressources RDF et de naviguer sur le Web en passant d'une ressource HTML à une ressource RDF de manière transparente.*

Mots-clés : Web des données liées, Navigateur web, Visualisation, RDF

1 Introduction

L'interconnexion des réseaux informatiques, nommée l'Internet, a permis l'interconnexion des systèmes documentaires et la réalisation de l'idée d'hypertexte sous la forme du Web, un système global d'identification et d'échange d'objets numériques. Le premier âge du Web s'est limité à un Web des documents liés, téléchargés et affichés par un navigateur hypertexte. Avec l'avènement du Web des données, ce sont des données liées qui sont maintenant échangées entre le client et le serveur. Ces données sont transmises de manière structurée, sans être encapsulées dans des documents et des descriptions textuelles. Puisque nous utilisons quotidiennement un navigateur hypertexte pour parcourir le Web des documents, il nous a paru nécessaire de nous interroger sur ce que pourrait être un navigateur pour le Web des données, qui nous permettrait de parcourir le gigantesque graphe global des données liées. Nous avons d'abord constaté que ce changement n'a rien de révolutionnaire : les données comme les documents sont désignées par des URL, téléchargées via HTTP et contiennent des liens vers d'autres données ou d'autres documents. La différence principale tient au fait que les documents sont décrits en HTML alors que les données sont exprimées dans de multiples vocabulaires métiers sur la base du formalisme RDF. En argumentant d'une part que rien ne devrait empêcher de créer des liens hypertextes entre les données et les documents et d'autre part que les documents ne sont qu'un type particulier de données encodées avec un vocabulaire dédié à la mise en page de texte, nous arrivons à la conclusion qu'il n'y a qu'un et un seul Web, et qu'il devrait nous suffire d'un et d'un seul navigateur. Le mécanisme de WebExtension permet justement d'étendre les fonctionnalités des principaux navigateurs (tels que Firefox et Chrome). Nous en avons tiré parti dans le but d'aboutir à une navigation aussi homogène et transparente que possible pour l'utilisateur, qui progresse en suivant les liens présents à l'écran et consulte tantôt des documents HTML, tantôt des données RDF, sans qu'une opération particulière ne soit nécessaire pour afficher les données RDF désignées par l'URL visible dans la barre d'adresse. Ayant constaté qu'un affichage reposant sur la structure du RDF, qu'il s'agisse d'un graphe ou d'une liste de triplets, était peu ergonomique faute de rendre compte de la nature des données, nous avons cherché à adapter la visualisation des données à leur type RDF. Ceci nous a conduit à un avantageux découplage entre le serveur qui fournit les données et le client qui choisit leur mise en page. La navigation en devient plus homogène, puisque les données de même nature peuvent être affichées de la même manière, indépendamment du serveur qui les publie.

Dans la suite de cet article, nous positionnons notre approche par rapport aux travaux qui se sont intéressés à la visualisation des données RDF, puis nous détaillons nos réalisations et nous les illustrons par une démonstration de deux cas concrets de navigation, avant d'ouvrir sur les perspectives que nous envisageons pour la suite de ce projet.

2 État de l'art

Le Web des données ouvertes et liées, aussi nommé Linked Open Data ou LOD, repose sur un ensemble de promesses comprenant :

— rendre les ressources (au sens large) adressables ;
— déréférencer l'adresse d'une ressource donnant accès aux connaissances relatives à cette ressource ;
— pouvoir faire référence à d'autres ressources pour décrire les connaissances associées à une ressource.

Dans une utilisation classique de RDF, ces promesses se déclinent par l'utilisation d'URI pour faire référence à une ressource, le déréférencement d'une telle URI renvoie un document RDF relatif à la ressource et les nœuds URI utilisés dans ces documents peuvent à leur tour être déréférencés, et ainsi de suite. Le LOD repose également sur la mise à disposition des données, en général par le biais d'entrepôts qu'il est possible de requêter. Afin de visualiser et naviguer dans les données du LOD, il est possible de distinguer deux paradigmes différents :

L'approche **centralisée** part du principe que l'intégralité des connaissances disponibles est accessible dans un unique entrepôt de triplets RDF qu'il s'agira de visualiser en tout ou partie. La difficulté est alors souvent de permettre à un utilisateur d'appréhender une nouvelle base de connaissances dont la taille peut être un obstacle à la compréhension.

L'approche **distribuée** part du principe que les connaissances sont réparties sur plusieurs serveurs à travers le web et qu'il va falloir naviguer de l'un à l'autre en suivant les liens pour aboutir au résultat de sa recherche. La difficulté principale est alors qu'on ne sait pas *a priori* où seront hébergées les données, ni quelles seront les ontologies utilisées pour définir les ressources rencontrées. Bien que centrée sur une ressource à la fois, cette approche permet de naviguer à travers l'ensemble des données via les liens présents. C'est en cela que l'analogie entre session de recherche sur le web de documents et session de recherche de connaissances sur le web des données liées, nous paraît pertinente.

Dans cette section nous étudierons les approches existantes pour ces deux paradigmes et nous aborderons une expérience interne à l'entreprise Logilab concernant l'utilisation de vues spécifiques à des vocabulaires RDF. Dans ce but, nous avons réincarné sous une forme plus moderne les idées développées durant les années 2000 dans le cadre de l'interface utilisateur du logiciel CubicWeb [1].

2.1 Consultation d'une base de connaissances

La question de la visualisation et de la navigation pour une base de connaissances complète se pose dans le cas d'entrepôts uniques, par exemple DBPedia. L'enjeu dans de tels cas est alors principalement exploratoire. Les approches relevant de ce paradigme permettent la visualisation et la navigation de ces bases en s'appuyant sur une logique de graphe. On peut ainsi naviguer de proche en proche entre les nœuds et obtenir une visualisation de ceux-ci, en général spécifique au fournisseur de l'entrepôt. Les travaux tels que IsaVizPietriga (2003), V-OWLLohmann *et al.* (2016), ou encore VizSKOSDestandau (2016) pour la visualisation de vocabulaire SKOS, proposent des visualisations génériques pour une exploration du graphe de connaissances dans son intégralité. Ces approches sont particulièrement pertinentes pour une visualisation d'une base de connaissances dans son intégralité et leur exploration sans connaître au préalable, ni le vocabulaire, ni ce que l'utilisateur cherche. Cependant, leur utilisation n'est possible que lorsque l'intégralité de la base de connaissance est accessible localement. Comme présenté dans l'introduction 1 nous souhaitons proposer un navigateur pour

1. `https://cubicweb.org`

le web des données liées. Il n'est donc pas envisageable de stocker l'intégralité du LOD et c'est pourquoi notre hypothèse initiale est l'affichage spécifique d'une ressource.

2.2 Affichage d'une ressource

Le second paradigme concerne la visualisation d'une ressource particulière. Dans le cadre du LOD, ce paradigme s'intéresse à la visualisation des connaissances à propos d'une ressource telle qu'accessible en déréférençant l'adresse de celle-ci. Dans le cas précis de RDF, cela correspond à s'intéresser à la visualisation spécifique de jeux de données RDF *relativement petits* puisque relatifs à une seule ressource à la fois, contrairement au paradigme précédent. Les approches comprises dans ce paradigme peuvent être catégorisées en deux groupes : les vues génériques et les vues spécifiques.

2.2.1 Vues génériques

Les connaissances sur le web des données liées étant représentées par des triplets RDF, il est possible de proposer une visualisation générique qui permet l'affichage d'une ressource sous la forme de triplets "sujet", "prédicat", "objet". Nous pouvons par exemple citer TabulatorBerners-Lee *et al.* (2008) qui propose un affichage sous forme d'arbre afin de parcourir le chemin de relation pouvant mener vers de nouvelles connaissances. Par exemple il est possible d'afficher les triplets dont une ressource est le sujet, et ensuite de parcourir (récursivement) les triplets dont un objet devient le sujet. La limite des approches proposant des vues génériques est aussi leur force. Bien que les vues n'ont besoin d'aucune configuration ou de développement pour être utilisables directement par un utilisateur, ces approches deviennent vite limitées dans le cadre d'un projet industriel. Effectivement, lorsqu'un client d'une entreprise décide d'avoir un affichage pour ses données exprimées en RDF, il est rare qu'une représentation aussi générique qu'un arbre ou un tableau de triplets soit suffisant, car l'étalon reste les interfaces utilisateur riches des applications faites sur-mesure. Nous pouvons prendre pour exemple une expérience observée chez Logilab. Notre entreprise développe le cadriciel de développement web CubicWeb [2]Simon *et al.* (2013), dont le modèle de données s'approche d'une base de connaissances de part l'utilisation d'entités reliées entre elles par des relations nommées et contenant un certain nombre d'attributs. Afin de visualiser ces données, CubicWeb possèdent un système de génération automatique de vues pour chaque type d'entité et de relation. De cette façon si un nouvel attribut, ou relation, est ajouté à l'entité, alors cette modification est directement visible sur l'interface, que ce soit en visualisation ou en modification. Le client observe donc un changement immédiat et peut émettre des retours. Cependant cette vue, auto générée, est rarement suffisante pour le client. Avec l'explosion des applications complexes côté clients et des technologies telles que React [3], Angular [4] ou encore Vue.js [5] (pour ne citer qu'eux), il devient difficile de proposer la même qualité de visualisation de données avec des vues auto générées et leur configuration devient problématique. Nous observons, dans la majorité de nos projets, la suppression de ces vues auto générées au profit de vues qui sont développées spécifiquement afin de répondre aux besoins des clients. Les vues auto générées ne sont donc pas une solution adaptée à nos besoins.

2.2.2 Vues spécifiques

Il faut pouvoir proposer des vues spécifiques pour des ressources RDF déréférencées. Pour cela, certains travaux proposent des approches intéressantes. Nous pouvons citer HaystackQuan *et al.* (2003) qui propose la définition de vues spécifiques pour chaque ressource. Bien que cette approche soit particulièrement intéressante, ces vues sont générées côté serveur. Il devient donc nécessaire d'avoir un serveur spécifique qui va servir les ressources et

2. https://www.cubicweb.org/
3. http://reactjs.org/
4. https://angular.io/
5. http://vuejs.org/

les vues associées pour obtenir le résultat souhaité. Notre objectif ici est de proposer une navigation sur le web de données liées, tout autant que sur le web des documents, Quel que soit le serveur fournissant la ressource et le type de ressource. Si chaque serveur proposant une ressource disponible sur le web des données liées devait être dans l'obligation de fournir une vue dédiée à cette ressource, alors la publication de données serait difficilement accessible. De plus cela contredirait la philosophie du web des données liées qui préconise de rendre disponible des données ouvertes et liées quelle que soit leur utilisation. D'autres approches, comme STTLCorby & Zucker (2015) proposent une visualisation spécifique, non pas d'une ressource, mais d'un résultat d'une requête SPARQL. Ce type d'approche ne permet pas une visualisation d'une ressource déréférencée : la ressource désirée étant un résultat d'une requête SPARQL, il faut obligatoirement un serveur SPARQL pour que STTL fonctionne. De plus cela empêche la navigation d'une ressource à une autre alors que le web de documents le permet. Enfin, Piggy BankHuynh *et al.* (2007) propose une extension aux navigateurs traditionnels (Firefox et Chrome) pour permettre l'extension de la visualisation de document grâce à du contenu RDF ou RDFa. Cette approche se concentre sur l'amélioration de l'affichage de document et non pas sur la visualisation de ressources RDF à proprement parler. Pour cela les documents HTML sont parcourus afin d'en extraire un certain nombre de triplets RDF (grâce au micro format ou au "scrapping") pour étendre la visualisation. Néanmoins cette approche est intéressante grâce à la décision d'utiliser une extension de navigateur pour étendre ses fonctionnalités.

2.3 Sélection d'une vue

Nous souhaitons une approche permettant une visualisation d'une ressource RDF déréférencée grâce à une vue spécifique pour un type de ressource donné. Nous réfléchissons donc au moyen de sélectionner une vue qui corresponde à ce que désire l'utilisateur. Les travaux effectués pour FresnelPietriga *et al.* (2006) proposent une réponse à cette problématique et ont bénéficié d'une standardisation du W3C [6] qui se décompose en deux parties. La première partie est FSL [7] un langage de sélection de triplets RDF, qui s'apparente à XPath. De cette manière, il est possible d'exprimer une contrainte de sélection pour l'application d'une vue spécifique, *e.g.* un sélecteur pour les ressources de type $foaf : Person$ ayant une valeur spécifique pour un attribut donné. L'objectif de FSL est de permettre la définition d'un sélecteur de vue et son partage entre les systèmes de visualisation de vue RDF. De cette manière, il devient plus aisé de définir un sélecteur de vue de façon générique et de le réutiliser. La deuxième partie de Fresnel est le formatage d'une ressource. Il ne s'agit pas de vue à proprement parler mais plus de contraintes de formatage telles qu'elles peuvent être définies dans un fichier CSS. Par exemple, nous pouvons spécifier que, pour un sélecteur FSL d'une ressource de type $foaf : Person$, le nom $foaf : name$ de cette ressource sera affiché en gras. Ces contraintes sont, elles aussi, partageables et génériques puisque c'est l'objectif premier de Fresnel. Bien que notre approche n'utilise pas encore le FSL comme un moyen d'exprimer la sélection d'une vue, ce langage nous semble être une approche pertinente. Les contraintes d'affichage peuvent elles aussi être utiles pour s'assurer d'un affichage particulier quelle que soit la vue sélectionnée dans notre approche. Nous reviendrons sur ce point dans cet article. La problématique de sélection de la vue à utiliser pour l'affichage d'une ressource spécifique n'est pas anodine. Nous pensons que, à l'instar du principe selon lequel les données personnelles appartiennent à l'utilisateur proposé dans SolidMansour *et al.* (2016), la décision de la vue à utiliser est de la responsabilité de l'utilisateur lui-même. Pour cela nous proposons un système de serveur de vue configurable que nous détaillerons dans cet article.

Nous avons vu dans ce chapitre que les approches proposant une visualisation d'une base de connaissances entière ne correspondent pas à notre besoin. De plus, les approches de visualisation d'une ressource spécifique ne permettent pas la définition d'une vue spécifique,

6. `https://www.w3.org/2005/04/fresnel-info/`

7. Fresnel Selector Language

générée côté client et quel que soit le serveur distribuant cette ressource. C'est pour cela que nous proposons ici un navigateur pour le web des données liées répondant à ces besoins.

3 Proposition scientifique

Notre approche, présentée dans cette section, a pour objectif de proposer un navigateur pour le web des données liées. Elle s'inscrit donc dans le second paradigme de visualisation présenté dans la section 2. Nous nous intéressons à la possibilité de naviguer entre les ressources du LOD tout en visualisant celles-ci à l'aide de vues spécialisées par catégories de données, mais homogènes quant à leur source, laissant ainsi un contrôle fin de leur visualisation à l'utilisateur final.

3.1 Une extension des navigateurs traditionnels

Notre objectif est de permettre une exploration du web des données liées et du web des documents de manière totalement transparente pour l'utilisateur. C'est-à-dire que nous souhaitons pouvoir passer d'une ressource HTML à une ressource RDF aussi simplement que par un clic sur un lien. Cette transparence va permettre aux utilisateurs d'utiliser les ressources disponibles sur le web des données lées aussi facilement que les ressources du web des documents. Il est alors nécessaire de générer une visualisation HTML à partir de ressources RDF. Compte tenu des règles de sécurité dans les navigateurs, il n'est pas possible de servir une application à une URL qui consulte des données issues d'un autre serveur sans la coopération de ce dernier (règles CORS). Contourner le problème via un serveur proxy poserait des problèmes de passage à l'échelle et de vie privée (espionner trafic). C'est pourquoi, le plus simple est d'étendre les fonctionnalités du navigateur via une Web Extension. Nous avons donc développé et rendu disponible une extension à ces adresses :

Firefox :
— `https://addons.mozilla.org/fr/firefox/addon/linked-data-browser/`
Chrome :
— `https://chrome.google.com/webstore/detail/cubicweb-linked-data-brow/nbhcjnnbhnmbahlofnbfekjaohgmlfcb`

Nous vous encourageons à télécharger et à tester ces extensions et surtout à ne pas hésiter à nous faire des retours pour des améliorations possibles.

3.1.1 Négociation de contenu

Une fois l'extension installée sur le navigateur, chaque page consultée est analysée pour observer si cette ressource (ici documentaire) est disponible sous forme d'un jeu de données RDF. À chaque chargement d'une page, deux méthodes sont utilisées pour détecter si la ressource est aussi disponible en RDF. La première méthode est l'analyse de l'entête de la réponse HTTP afin d'observer si un attribut "LINK" est présent avec une valeur pointant sur une ressource RDF. Cet attribut est défini dans la RFC5988 [8]. La deuxième méthode est de regarder le contenu de la balise "<head>" du contenu HTML, si la balise "<link rel='alternate' >" est disponible avec un type assimilé à une ressource RDF [9]. Nous ne prétendons pas couvrir tous les cas possibles de spécification de l'existence d'une ressource sous un format alternatif, mais ces deux méthodes nous semblent fiables et couramment utilisées. Nous l'avons, par exemple, utilisé sur data.bnf.fr ou encore sur dbpedia.org.

3.1.2 Activation de l'extension

Si au moins une de ces deux conditions est respectée alors l'extension s'active, et une icône apparaît à coté de la barre d'adresse. Si l'utilisateur clique sur cette icône, alors une nouvelle

8. `https://tools.ietf.org/html/rfc5988#section-3`
9. application/rdf+xml, text/turtle, ...

requête HTTP est envoyée en modifiant, dans l'entête HTTP, la valeur du champ "Accept" [10] pour demander le jeu de données RDF associé. De cette manière, l'extension récupère les triplets RDF associés à la ressource et peut ensuite utiliser le système de sélection de vue pour générer une visualisation adaptée. L'extension utilise un ensemble d'heuristiques pour déterminer quelle est l'entité RDF *principale* dans le jeu de données RDF récupéré. Il s'agit dans les cas simples de l'URI à l'origine de l'activation, laquelle peut être différente de l'URI déréférencée pour récupérer le jeu de données. Mais il est tout à fait possible que le sujet principal ne soit pas désigné par l'URI dans la barre d'adresse. C'est notamment le cas lors de l'utilisation de la relation $foaf : focus$ [11].

3.2 Sélection de vue

Une fois une ressource RDF identifiée, le jeu de données associé récupéré et l'URI de l'entité principale définie, l'extension doit sélectionner une vue pour générer la visualisation correspondante. Afin de permettre la diffusion et le contrôle des vues à utiliser, nous proposons la notion de serveur de vues. Ces serveurs rendent accessibles un ensemble de vues et leurs descriptions afin que l'extension puisse identifier quelle vue utiliser pour une ressource donnée.

3.2.1 Serveur de vues

Un serveur de vues met à disposition un ensemble de vues. Pour cela, le point d'entrée est un fichier JSON contenant le descriptif de toutes les vues disponibles sur ce serveur. L'URL de ce fichier JSON est la référence du serveur de vues. L'extension dispose d'une interface de configuration dans laquelle il est possible d'ajouter ou de supprimer un serveur de vue. De cette manière, l'utilisateur contrôle directement les vues qu'il souhaite utiliser ou non. Ce n'est plus le fournisseur des données qui a la responsabilité de la visualisation mais un développeur de vue qui les rendra disponible directement sur ce serveur de vues. De cette manière, la séparation entre la logique métier et l'affichage lors d'un développement d'une application web est d'autant plus respectée, grâce à cette gestion de serveurs de vues. La figure 1 présente un exemple d'un fichier JSON définissant un ensemble de vues.

Ce fichier contient deux vues. Une pour afficher une personne ($foaf : Person$) et une autre pour afficher un livre ($purl : Book$). Chaque vue a les caractéristiques suivantes :

identifier : l'identifiant de la vue

name : le nom de la vue qui sera affichée pour l'utilisateur

description : une description pour avoir un peu plus de détail sur la vue

entrypoint : le nom de l'objet Javascript servant de point d'entrée pour la vue dans la ressource principale

resourceCss : la liste des ressources CSS à importer

resourceJs : la liste des ressources Javascript à importer

resourceMain : la ressource Javascript principale qui contient le point d'entrée

Nous remarquons dans ce fichier de configuration que les deux vues sont disponibles sur le même ordinateur que le client ("http ://localhost :8080/..."). Il est tout à fait possible de distribuer des vues qui ne sont pas sur le même serveur.

3.2.2 Description des différentes vues disponibles

Grâce à ces serveurs de vues, il est possible de distribuer un certain nombre de vues depuis un serveur donné. Il est aussi possible que les vues dont l'utilisateur a besoin se trouvent sur plusieurs serveurs de vues. Par exemple, l'utilisateur souhaite une vue pour $foaf : Person$

10. https://developer.mozilla.org/fr/docs/Web/HTTP/Headers/Accept
11. http://xmlns.com/foaf/spec/#term_focus

```
[
  {
    "identifier": "::Logilab::Person",
    "name": "Logilab: Person View",
    "description": "Renders a person from a (possibly FoaF) dataset",
    "entrypoint": "VIEW_PERSON_ENTRYPOINT",
    "resourceCss": [
      {
        "location": "remote",
        "uri": "https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
      }
    ],
    "resourceJs": [],
    "resourceMain": {
      "location": "remote",
      "uri": "http://localhost:8080/view_person.js"
    }
  },
  {
    "identifier": "::Logilab::Book",
    "name": "Logilab: Book View",
    "description": "Renders a book from a dataset",
    "entrypoint": "VIEW_BOOK_ENTRYPOINT",
    "resourceCss": [
      {
        "location": "remote",
        "uri": "https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
      }
    ],
    "resourceJs": [],
    "resourceMain": {
      "location": "remote",
      "uri": "http://localhost:8080/view_book.js"
    }
  },
]
```

FIGURE *1 – Fichier de configuration d'un serveur de vue*

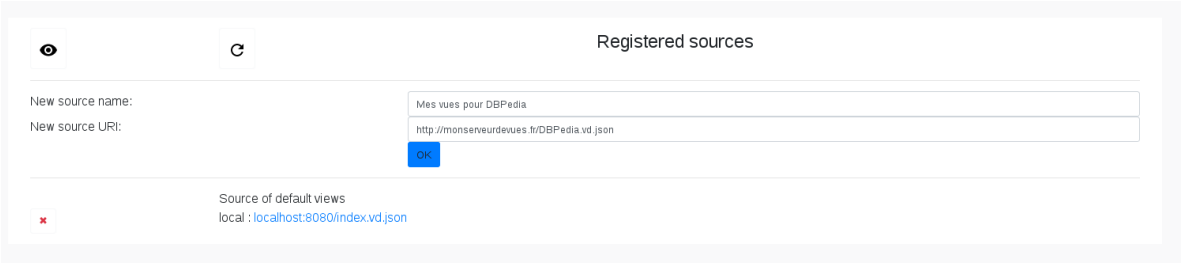

FIGURE *2 – Configuration d'un nouveau serveur de vues dans l'extension*

et une autre vue pour *dbpedia* : *Event* mais ces deux vues ne sont pas définies sur le même serveur. Dans ce cas-là, il est possible de définir plusieurs serveurs de vues comme configuration de l'extension comme illustré sur la figure 2. Sur cette figure un nouveau serveur de vues nommé "Mes vues pour DBPedia" et hébergé sur "http ://monserveurde-vues.fr/DBPedia.vd.json" est en train d'être ajouté. Lorsque le bouton "OK" sera cliqué, les vues de ce nouveau serveur apparaîtront dans la liste des vues disponibles. À chaque ajout d'un nouveau serveur, la liste des vues disponibles sur ce serveur est enregistrée. Cette liste est ensuite utilisée comme vue potentielle pour une ressource donnée.

3.2.3 Algorithme de sélection d'une vue en fonction de la source d'intérêt

Nous venons de voir qu'il est possible de définir plusieurs vues sur un serveur et plusieurs serveurs dans l'extension. Chaque vue dispose d'une fonction "priorityFor" [12] qui retourne un entier pour définir la priorité de cette vue pour visualiser une ressource en particulier. De cette manière dès qu'une ressource est à visualiser par l'extension, celle-ci parcourt toutes les vues disponibles dans les différents serveurs de vues et calcule cette priorité. La vue qui retourne la priorité la plus haute sera alors sélectionnée pour visualiser la ressource. Cette fonction de priorité peut implémenter différents types d'algorithme pour déterminer la pertinence d'une vue pour une ressource en particulier. De cette manière toutes les contraintes peuvent être exprimées ici. Le plus courant reste tout de même un filtre sur le type de la ressource. Par

12. La structure d'une vue sera décrite dans la section 3.3

exemple si l'utilisateur souhaite visualiser une ressource de type $foaf : Person$ la vue identifiée par " : :Logilab : :Person" va probablement retourner une valeur haute pour la priorité. Cette vue regarde le type de la ressource (un triplet de type : "<ressource> rdf :type foaf :Person"), si ce type est bien celui d'une personne du vocabulaire $foaf$, alors elle retourne une valeur forte (dans notre exemple la valeur est 10). Ce procédé a l'avantage d'être relativement simple à implémenter et permet d'avoir un système de sélection de vue à moindre coût. Néanmoins la définition d'une valeur comme priorité est forcément corrélée aux valeurs de priorité qu'ont pu définir les autres vues. Donc une valeur de priorité n'a de sens que dans le contexte d'un ensemble de vues. À partir du moment où les vues sont développées par plusieurs personnes, ces valeurs de priorités deviennent moins pertinentes puisque chaque personne peut définir une valeur arbitrairement. Une évolution intéressante ici serait de prendre en considération les FSL [13] pour définir les contraintes de sélection de vue de manière standardisée et permettre un calcul de priorité plus fin. Une fois une vue sélectionnée, l'utilisateur a toujours la possibilité de changer la vue choisie grâce à l'interface de l'extension. De cette manière si la sélection de la vue n'a pas sélectionné la vue la plus adaptée, l'utilisateur peut décider d'en utiliser une autre.

3.3 Définition d'une vue

Une fois une vue pertinente sélectionnée pour une ressource donnée, la vue est utilisée pour générer la visualisation. Pour cela, chaque vue définit une fonction "*render*" qui sera utilisée pour générer le contenu HTML de la visualisation en fonction de la ressource pertinente. Comme vu dans la section 3.2.1 des ressources externes sont chargées (comme des fichiers CSS ou Javascript) qui seront utilisées ensuite dans la visualisation elle-même. Une ressource peut être liée, par l'intermédiaire de relation, à d'autres ressources déréférencées. L'extension permet de passer d'une ressource à une autre par un simple clic.

3.3.1 Sélection des paramètres d'intérêts pour la vue

Seulement une partie des attributs présents dans le RDF sont pertinents pour une vue donnée. C'est pourquoi dans cette fonction *render*, les attributs d'intérêts sont pré-chargés. Cela signifie qu'un ensemble d'attributs et de relations pertinents pour cette visualisation sont pré-chargés en explorant les triplets du jeu de données retourné par la requête HTTP présentée en 3.1.2. Pour les labels qui sont pertinents, une langue par défaut est configurée dans la vue pour obtenir les chaînes de caractères dans la bonne langue. L'ensemble de ces valeurs sont ensuite fournies à la fonction *render*, qui est chargée de produire le HTML correspondant. L'avantage majeur de cette implémentation réside dans la liberté d'utilisation de ces valeurs et dans le développement de la visualisation. En effet si une personne désire développer une vue en utilisant une technologie spécifique, par exemple React, alors le template de la vue peut être un composant React. Il n'y a, ici, aucune contrainte concernant une technologie plutôt qu'une autre, si ce n'est le javascript initial pour charger la ressource et initialiser le template. Ce n'est pas anodin puisque dans l'univers du développement web, et particulièrement le web appelé "frontend", les technologies évoluent à une vitesse impressionnante. Il devient donc primordial de pouvoir s'adapter à cette évolution. Dans ce cadre, il est tout à fait possible d'utiliser, par exemple, Fresnel afin de réaliser le rendu d'une vue.

3.3.2 Pré-chargement des ressources liées

Dans cette approche, les attributs et relations d'une entité RDF sont très souvent utilisés dans la visualisation de l'entité RDF. Les attributs permettent d'observer directement l'information importante à afficher puisque l'objet du triplet est directement la valeur sous la forme d'une chaîne de caractères ou autre. Néanmoins, pour les relations, la problématique est que l'objet des triplets est une URI référençant une nouvelle entité. L'extension permet d'utiliser les triplets relatifs à cette nouvelle entité, s'ils sont déjà présents dans le jeu de

13. Fresnel Selector Language `https://www.w3.org/2005/04/fresnel-info/fsl/`

données RDF courant, ou bien s'ils peuvent être déréférencés via leur URI en se basant sur les promesses du LOD et pour récupérer des triplets distants. Ces nouveaux triplets peuvent alors être utilisés dans la visualisation de l'entité RDF originale. Un exemple simple est l'affichage d'une vue pour les ressources de type "$foaf : Person$", qui détiennent une relation "$foaf : knows$" vers une autre "$foaf : Person$". Cette autre "$foaf : Person$" est une URI alors que pour la visualisation nous aimerions avoir le nom de cette personne pour afficher la liste des connaissances d'une personne directement sous la forme d'une liste de noms au lieu d'une liste d'URIs. Le choix qui a été fait ici est d'afficher dans un premier temps les URI de ces personnes dans la visualisation, et de récupérer en tâche de fond les noms de ces personnes de manière asynchrone comme décrit ci-dessus. Ainsi, dès que les triplets concernant une personne connue sont récupérés, le nom est extrait et vient remplacer l'URI qui est affichée de manière temporaire. De cette manière, l'interface n'est jamais bloquée ou incomplète. L'information qui y est affichée se précise au fur et à mesure que les connaissances sont récupérées.

3.3.3 Changement de vue dynamique

L'affichage d'une relation revient donc à afficher une URI, puis un label préféré (par exemple le nom d'une personne) dès qu'il est récupéré. Seulement l'URI qui fait référence à cette ressource liée peut, elle aussi, être utilisée comme ressource principale. Si l'utilisateur clique sur une URI déréférencée avec négociation de contenu disponible (comme vue dans la section 3.1.1), alors tout le processus est relancé depuis le début. C'est-à-dire que l'url change dans la barre d'adresse du navigateur, que cette ressource devient la ressource principale, les vues disponibles dans l'ensemble des serveurs de vues sont parcourues pour déterminer la vue la plus pertinente, les attributs et relations pertinents sont récupérés et la visualisation est générée. De cette manière il devient aisé de naviguer d'une ressource à une autre. Un avantage majeur ici est que ce système s'applique quel que soit le serveur qui fournit la ressource. De cette manière, la navigation peut commencer sur notre propre base de connaissances en observant une visualisation d'une personne qui connait "Claude Debussy" qui est référencé par son URI sur DBPedia [14]. Un simple clic sur un lien vers cette URI permet d'afficher une visualisation pour la ressource "$dbpedia : Claude\_Debussy$". Pour aller encore plus loin dans l'interopérabilité et dans la généricité de la navigation, si un lien présent sur une visualisation pointe vers un document, et non plus sur une ressource déréférencée, alors l'extension se désactive automatiquement et laisse place à la visualisation "classique" d'un document HTML par le navigateur. L'hypothèse initiale qui stipulait que nous souhaitons un navigateur qui soit transparent pour l'utilisateur quant au type de la ressource à afficher est ici respectée. Une critique majeure peut être faite sur la nécessité d'installer une extension pour permettre l'affichage de ressources RDF. Il serait donc intéressant d'introduire directement ce fonctionnement dans les navigateurs traditionnels. Si ce fonctionnement convainc une communauté suffisamment grande il pourrait être intéressant de proposer une standardisation de cette forme de communication. Pour l'instant nous nous positionnons comme une preuve de concept. Néanmoins, nous envisageons d'introduire ce type de fonctionnement dans le framework web développé par Logilab [15] pour faciliter le développement de vues dans nos différents projets.

L'ensemble des propositions présentées dans cette section est disponible sous licence LGPL3 à cette adresse : `https://www.cubicweb.org/project/cubicweb-linked-data-browser/`. Une documentation est aussi mise à disposition pour donner plus de détails sur l'implémentation et aider les personnes désirant mettre en œuvre cette approche. Nous restons bien entendu à l'écoute des retours qui peuvent être faits pour améliorer ce système.

14. `http://dbpedia.org/page/Claude_Debussy`
15. CubicWeb`http://www.cubicweb.org`

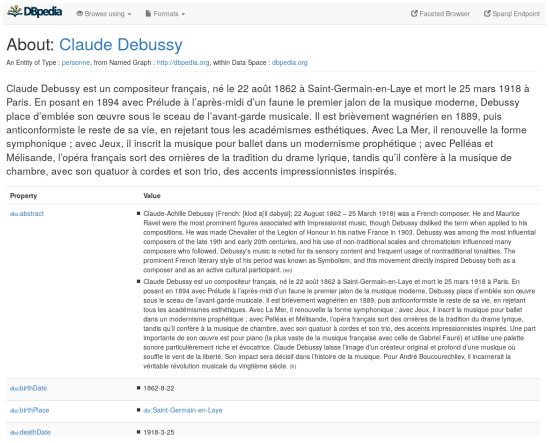

FIGURE *3 – Capture d'écran de la page HTML concernant Claude Debussy sur DBPedia*

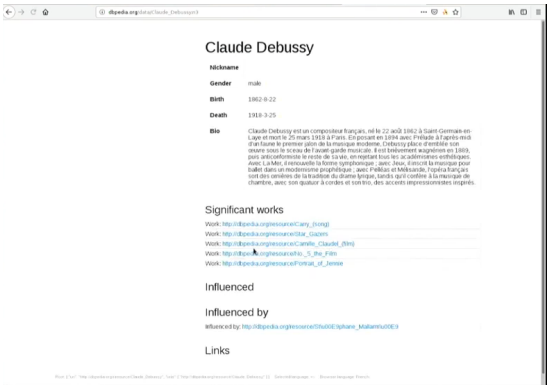

FIGURE *4 – Visualisation générée par la vue " : :Logilab : :Person" avant le chargement des ressources liées*

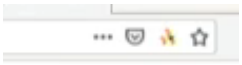

FIGURE *5 – Icône d'activation de l'extension*

4 Démonstration

Afin d'illustrer notre proposition nous allons, dans cette section, présenter deux cas concrets d'utilisation de notre extension. Le premier est une recherche sur le compositeur Claude Debussy pendant laquelle nous passerons d'une visualisation d'un document HTML à la visualisation de données RDF. Le second est l'utilisation de notre extension afin de visualiser les données du site de la conférence SemWeb.pro.

4.1 Claude Debussy sur DBPedia

Notre premier exemple se place suite à une recherche sur votre moteur de recherche préféré des mots clefs "Claude Debussy". Cette session de recherche vous amène directement sur la page HTML concernant la personne "Claude Debussy" sur DBPedia : `http://dbpedia.org/page/Claude_Debussy`. La figure 3 présente cette page.

Sur cette image nous pouvons observer que la visualisation proposée par DBPedia est une liste de triplets dont "$dbpedia : Claude\_Debussy$" est le sujet. Cette visualisation n'est pas modifiable puisque générée par le serveur sous forme HTML. Néanmoins, DBPedia propose la possibilité d'effectuer de la négociation de contenu pour obtenir les données RDF qui ont été utilisées pour générer cette visualisation. Il est possible de le vérifier en regardant dans la barre d'adresse, l'icône de l'extension apparaît comme sur la figure 5. Cette icône n'apparaît que lorsque la négociation de contenu est disponible.

Si nous cliquons sur cette icône, alors l'extension s'active. La première tâche de l'extension est de récupérer les données RDF associées à la page qui était en cours de consultation. Une fois la liste des triplets RDF récupérée, celle-ci est analysée pour déterminer la vue la plus appropriée. Dans notre cas, le triplet indiquant que la ressource "$dbpedia : Claude\_Debussy$" est de type "$foaf : Person$" permet de sélectionner la vue " : :Logilab : :Person" fournie par le serveur dont la configuration est présentée sur la figure 1 à la page 7. Lorsque l'extension a sélectionné la vue la plus appropriée (ici " : :Logilab : :Person"), celle-ci est utilisée afin de générer la visualisation pour les données RDF récupérées. La figure 4 permet de voir la visualisation générée. Cette visualisation est relativement simple, néanmoins, il est possible de voir que seulement certaines données concernant Claude De-

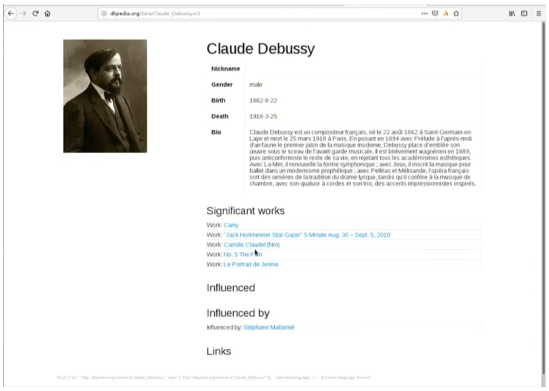

FIGURE 6 – *Visualisation générée par la vue " : :Logilab : :Person" avec les données des ressources liées*

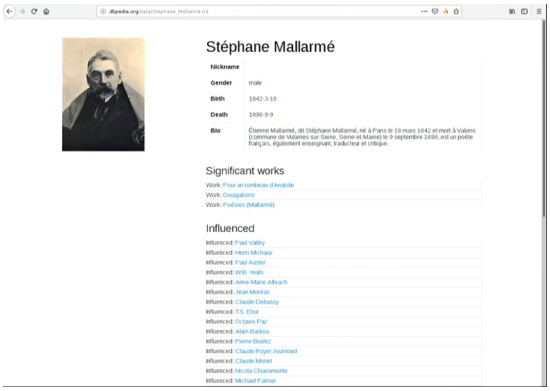

FIGURE 7 – *Visualisation pour Stéphane Mallarmé générée par la vue " : :Logilab : :Person"*

bussy sont affichées et non plus l'ensemble des triplets, comme c'est le cas pour la visualisation proposée par DBPedia. De plus, ici, un certain nombre de ressources liées sont présentées dans cette visualisation mais les ressources liées ne sont pas récupérées directement lors de la négociation de contenu. Elles sont interrogées, par une requête AJAX, dès l'affichage de la visualisation de la vue. C'est pour cela que nous pouvons ici observer des URI affichées à côté des relations vers des ressources liées. Par exemple dans la catégorie "Influenced by" l'URI correspondante à la ressource de Stéphane Mallarmé est affichée. Dès que les données correspondantes à cette ressource sont récupérées, le nom de la personne est présentée à la place de l'URI. Le même fonctionnement est appliqué pour chaque URI présentée, mais aussi pour la photo de Claude Debussy. La figure 6 présente la même visualisation mais avec toutes les ressources liées récupérées.

Dans cette visualisation, nous pouvons voir que Stéphane Mallarmé a influencé Claude Debussy. La chaîne de caractères "Stéphane Mallarmé" a été récupérée en interrogeant la ressource référencée par l'URI "`http://dbpedia.org/resource/Stéphane_Mallarmé`". Cette chaîne de caractères est affichée comme un lien pointant vers la ressource associée. Il est donc possible de cliquer directement sur ce lien pour accéder à la visualisation de la ressource pour Stéphane Mallarmé. Stéphane Mallarmé étant aussi une personne, la visualisation utilise la même vue. Cette visualisation est présentée sur la figure 7. Nous avons vu qu'il est possible de passer d'une visualisation d'un document HTML vers une ressource RDF. Il est aussi possible de passer d'une ressource RDF vers une autre ressource RDF. Tout le traitement de sélection de vue et de génération de la visualisation étant effectué côté client, tout est dynamique. Par conséquent, il est possible d'effectuer le chemin inverse, c'est à dire d'une ressource RDF vers un document HTML.

4.2 Visualisation des données de la conférence SemWeb.pro

Une version spécifique du site de la conférence SemWeb.pro 2018 [16] a été déployée afin de permettre la négociation de contenu sur toutes les pages. Des vues spécifiques ont également été développées et rendues disponibles sur le serveur de vues "`https://views.semweb.pro/`". Le même procédé que pour la page de Claude Debussy est appliqué, mais cette fois ci sur le document HTML répertoriant les communications lors de SemWeb.pro 2018. La figure 8 présente la visualisation HTML générique du site de la conférence générée par le serveur. En activant l'extension la vue " : :Logilab : :Conference" est utilisée pour générer la visualisation présentée sur la figure 9.

Si nous cliquons par exemple sur la présentation se nommant "Un navigateur pour le web des données" nous arrivons sur une visualisation générée par la vue " : :Logilab : :Conference-

16. `http://demo.semweb.pro`

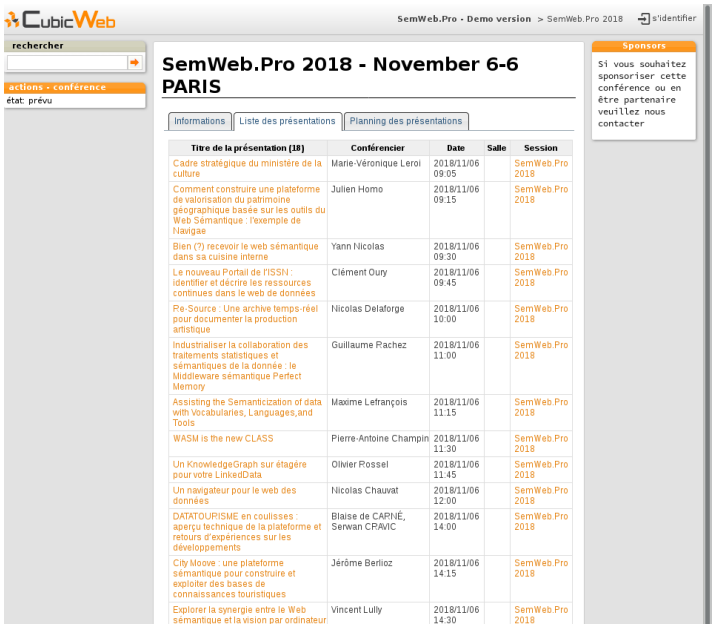

FIGURE 8 – *Visualisation générée par le serveur pour la conférence SemWeb.pro 2018*

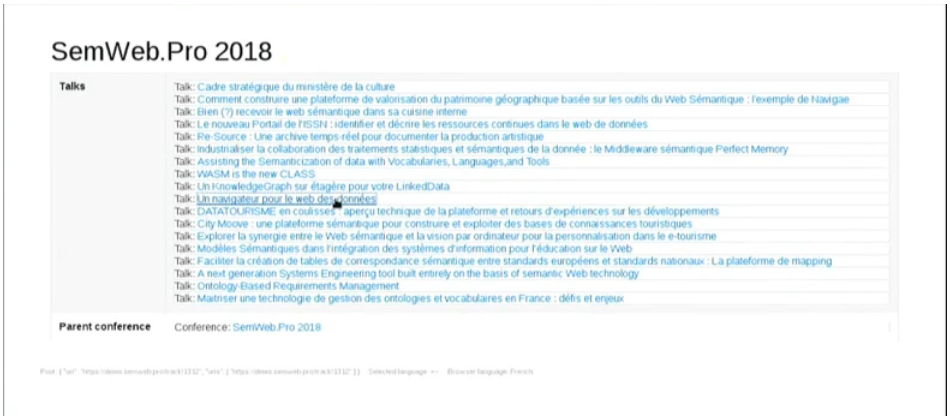

FIGURE 9 – *Visualisation générée par la vue " : :Logilab : :Conference" pour la conférence SemWeb.pro 2018*

FIGURE *10 – Visualisation d'une communication de la conférence SemWeb.pro 2018, générée par la vue " : :Logilab : :ConferenceTalk"*

FIGURE *11 – Visualisation d'un auteur d'une communication de la conférence SemWeb.pro 2018 en utilisant la vue " : :Logilab : :Person"*

Talk". Cette visualisation est présentée sur la figure 10. Nous observons que le changement de ressource dans l'extension peut aussi amener à changer de vue pour générer la visualisation. Si nous cliquons sur l'auteur de la communication ("Nicolas Chauvat") alors nous pouvons observer une visualisation générée pour des ressources de type "$foaf : Person$". La vue utilisée sera donc " : :Logilab : :Person", comme pour la ressource "Claude_Debussy" provenant de DBPedia. Une vue créée pour cette extension peut donc être utilisée pour un type de ressource en particulier, quel que soit le serveur qui distribue cette ressource. La visualisation pour "Nicolas Chauvat" est présentée sur la figure 11.

Sur cette figure il est possible de voir que la conférence SemWeb.pro ne détient que très peu d'informations sur Nicolas Chauvat. Mais aussi que Nicolas Chauvat semble connaître Claude Debussy. Cette ressource est celle provenant de DBPedia. Si nous cliquons sur ce lien nous tomberons sur la figure 6. Cet exemple illustre le passage transparent d'un serveur à un autre en suivant les liens affichés et en réutilisant la même vue pour mettre en page des données de même type indépendamment du serveur dont elles proviennent. Il illustre aussi le fait que les mêmes données, ici la description d'une conférence, peuvent être mises en page soit par le serveur (cf figure 8), soit par le navigateur (cf figure 9).

5 Conclusion et perspectives

5.1 Un navigateur pour le web des données liées

Le web est un système décentralisé d'identification (URL) et d'échange (HTTP) d'objets numériques qui peuvent être des documents (HTML) ou des données (RDF). Nous avons cherché à disposer d'un navigateur avec lequel parcourir de façon homogène et transparente l'ensemble du Web, en suivant à chaque fois les liens qui pointent d'une ressource à la suivante, qu'elle soit constituée par un document HTML ou des données RDF. Nous avons développé une WebExtension qui permet, lorsque l'URL de la barre d'adresse désigne des données RDF, de mettre en page automatiquement les données reçues pour les visualiser. Pour cela, nous avons défini la notion de "vue" comme étant une fonction javascript qui prend en entrée un graphe RDF et produit en sortie un document HTML. Une fois installée, l'extension du navigateur est configurée en y ajoutant ces vues publiées statiquement sur des serveurs HTTP. Lors du chargement de données RDF, un mécanisme de sélection choisit la vue la plus adaptée et l'applique pour que les données reçues soient mises en page et que le HTML résultant soit affiché. Ceci a lieu côté navigateur, sans interaction avec le serveur. Nous avons ainsi obtenu un navigateur aux fonctionnalités étendues, avec lequel la séparation entre la publication des ressources par le serveur et leur visualisation par le client est plus

nette qu'elle ne l'était jusqu'à présent, ce qui présente plusieurs avantages.

Le premier avantage est la possibilité de naviguer de façon plus homogène parmi des sites qui utilisent les mêmes ontologies. Comme les vues ne dépendent pas du serveur depuis lequel les données sont téléchargées, le sentiment de parcourir un ensemble cohérent d'informations est nettement renforcé, même si l'on passe d'un serveur à un autre durant la navigation, car les données de même nature donnent lieu au même affichage. Ceci nous rapproche de la consultation d'un graphe de données global plutôt que d'un ensemble de silos interconnectés.

Le deuxième avantage est la possibilité de personnaliser les interfaces homme-machine, puisque la sélection de la visualisation est faite dans le navigateur à partir des vues configurées. Deux utilisateurs peuvent donc faire des choix de configuration différents et ne pas avoir le même affichage quand ils consultent la même URL qui désigne des données RDF. Chacun pourra donc disposer d'une vue adaptée à ses préférences ou à sa tâche en cours.

Le troisième avantage est d'abaisser globalement le coût de l'échange d'information entre serveur et client, puisqu'il devient possible de mutualiser le coût de la mise au point de la visualisation. Là où chaque organisme souhaitant partager des données avait l'obligation de développer et de maintenir une interface utilisateur, nous proposons des vues qui ne dépendent que du type des données et peuvent être utilisées avec un nombre quelconque de serveurs. Un organisme qui souhaite partager ses données peut donc le faire à moindre coût en choisissant des ontologies pour lesquelles plusieurs des vues nécessaires sont déjà disponibles et utilisées.

5.2 Perspectives

Bien que l'extension décrite ci-dessus soit d'ores et déjà utilisable, nous avons identifié plusieurs pistes d'amélioration.

Diagnostic en cas d'erreur

Notre extension est fonctionnelle, mais dans de nombreux cas, ne donne pas les résultats escomptés. La forme RDF des données n'est pas trouvée, la vue espérée n'est pas sélectionnable, les liens ne désignent pas les ressources attendues, etc. Les raisons peuvent être multiples : absence des en-têtes adéquates, erreur dans la négociation de contenu, mécanisme de redirection mal mis en oeuvre, triplets RDF manquants, etc. Pour établir un diagnostic facilement, il est nécessaire que l'extension fournisse un maximum d'informations sur les échanges qui ont eu lieu avec le serveur HTTP. Nous avons commencé à développer une barre latérale qui peut afficher ces informations et poursuivrons ce travail rapidement.

Simplification du développement et du déploiement des vues

Le développement et le déploiement de nouvelles vues doit être aussi simple que possible. Le choix qui a été fait d'utiliser des fonctions javascript pour les vues laisse une grande latitude au développeur et n'impose pas de changement radical par rapport aux méthodes et outils ayant cours actuellement. Fournir des exemples d'utilisation des librairies les plus répandues pourrait faciliter l'adoption. Une documentation existe déjà, mais elle est améliorable et pourrait être assortie de plus d'exemples et de modèles à partir desquels démarrer un nouvel ensemble de vues.

Compatibilité avec Fresnel

Comme décrit dans la section 3.2.3, la sélection de vue se fait actuellement avec un système de priorité. Lorsque l'extension est configurée avec des vues issues de plusieurs serveurs, il n'existe aucune garantie quant à la cohérence des priorités fournies par ces différentes vues. Donner un score de priorité à une vue sans connaître ni les critères ni les autres valeurs n'est pas chose aisée et peut amener à des conflits et des sélections erronées. Bien que l'utilisateur puisse changer de vue s'il le souhaite, il serait intéressant d'améliorer ce système

de sélection de vue pour toujours avoir la vue la plus pertinente sans intervention de l'utilisateur. Une solution pourrait être d'associer une expression Fresnel Selector Language (FSL) à chaque vue et d'utiliser ces expressions au moment de sélectionner la meilleure vue possible pour les données reçues.

Algorithme de selection de vue

Il est possible aussi de trouver une meilleure méthode pour la selection de la vue la plus pertinente. Une idée serait de pouvoir proposer à l'utilisateur du navigateur de pouvoir ordonner les serveurs de vues pour définir une priorité. L'extension essaierait d'abord de trouver une vue pertinente dans le premier avant d'étudier le deuxième, et ainsi de suite. Les scores de priorité des vues seraient uniquement en fonction des autres vues du même serveur et non de toutes les vues ajoutées par l'utilisateur.

Participation à la standardisation de la navigation RDF

Nous avons vu dans la section 3.1.1 que nous utilisons soit un paramètre de la réponse HTTP soit une balise dans le code HTML pour vérifier la disponibilité de la ressource dans un autre format. Il serait intéressant de couvrir toutes les possibilités de référence à des formats alternatifs pour la négociation de contenu et surtout de pouvoir respecter une méthode standard, si elle existe ou si une convention finit par apparaître.

API Hypermedia

Nous nous sommes concentrés sur la lecture des ressources publiées sur le Web, mais sommes bien conscients de l'importance de l'écriture et de la modification des données. Nous comptons explorer les relations entre nos travaux et les standards existants pour les API Hypermedia et la manipulation de RDF, parmi lesquels HAL, Hydra et Linked Data Platform.

Vues pour la consultation d'entrepôts RDF

Comme expliqué dans la section 2, nous avons distingué la navigation au sein du Web de la consultation d'un graphe RDF publié par exemple via un point d'accès SPARQL. Nous pensons que les vues telles que nous les avons définies et employées pourraient être réutilisées par un outil de consultation générique de graphe RDF.

Industrialisation

Nous souhaitons aussi obtenir un système suffisamment efficace, fiable et robuste pour qu'il puisse être utilisé au quotidien par nos clients. Nous développons aujourd'hui principalement en utilisant CubicWeb, qui propose un système de vues générées par le serveur. Notre objectif serait de les supprimer au profit de vues génériques ou spécifiques ajoutées au navigateur comme décrit ici. En découplant la publication des données du déploiement des vues, nous espérons faciliter leur modification et donc accroître notre capacité à livrer des changements en continu, comme encouragent à le faire les méthodes agiles que nous pratiquons.

Collaborations et logiciel libre

Le code source que nous avons développé est publié sous la licence LGPLv3 à l'url `https://www.cubicweb.org/project/cubicweb-linked-data-browser`. Nous sommes ouverts à toutes formes de collaboration.

Remerciements

Nous souhaitons remercier Noé Gaumont et Frank Bessou, qui sont nos collègues chez Logilab et qui ont participé à la rédaction, à la structuration et à la relecture de cet article. Nous remercions aussi Olivier Cayrol qui a su trouver le budget nécessaire à la rédaction de cet article. Enfin nous remercions tous les contributeurs de CubicWeb pour leurs idées et en particulier Sylvain Thénault, qui en a été le développeur principal pendant de longues années.

Références

BERNERS-LEE T., HOLLENBACH J., LU K., PRESBREY J. & SCHRAEFEL M. (2008). Tabulator redux : Browsing and writing linked data. *loutre*.

CORBY O. & ZUCKER C. F. (2015). Sttl : A sparql-based transformation language for rdf. In *11th International Conference on Web Information Systems and Technologies*.

DESTANDAU M. (2016). Vizskos, a visualizer for skos based thesaurus. *In Cartographie meetup, Paris*.

HUYNH D., MAZZOCCHI S. & KARGER D. (2007). Piggy bank : Experience the semantic web inside your web browser. *Web Semantics : Science, Services and Agents on the World Wide Web*, **5**(1), 16–27.

LOHMANN S., NEGRU S., HAAG F. & ERTL T. (2016). Visualizing ontologies with vowl. *Semantic Web*, **7**(4), 399–419.

MANSOUR E., SAMBRA A. V., HAWKE S., ZEREBA M., CAPADISLI S., GHANEM A., ABOULNAGA A. & BERNERS-LEE T. (2016). A demonstration of the solid platform for social web applications. In *Proceedings of the 25th International Conference Companion on World Wide Web*, p. 223–226 : International World Wide Web Conferences Steering Committee.

PIETRIGA E. (2003). Isaviz : A visual authoring tool for rdf. *World Wide Web Consortium.[Online]. Available : http ://www. w3. org/2001/11/IsaViz*.

PIETRIGA E., BIZER C., KARGER D. & LEE R. (2006). Fresnel : A Browser-Independent Presentation Vocabulary for RDF. In D. HUTCHISON, T. KANADE, J. KITTLER, J. M. KLEINBERG, F. MATTERN, J. C. MITCHELL, M. NAOR, O. NIERSTRASZ, C. PANDU RANGAN, B. STEFFEN, M. SUDAN, D. TERZOPOULOS, D. TYGAR, M. Y. VARDI, G. WEIKUM, I. CRUZ, S. DECKER, D. ALLEMANG, C. PREIST, D. SCHWABE, P. MIKA, M. USCHOLD & L. M. AROYO, Eds., *The Semantic Web - ISWC 2006*, volume 4273, p. 158–171. Berlin, Heidelberg : Springer Berlin Heidelberg.

QUAN D., HUYNH D. & KARGER D. R. (2003). Haystack : A platform for authoring end user semantic web applications. In *International Semantic Web Conference*, p. 738–753 : Springer.

SIMON A., WENZ R., MICHEL V. & DI MASCIO A. (2013). Publishing bibliographic records on the web of data : opportunities for the bnf (french national library). In *Extended Semantic Web Conference*, p. 563–577 : Springer.