|  |
| --- |
| Master 1 DSC - Groupe 2 - Interopérabilité |
| Document final de conception |
| 26/03/2020 |

|  |
| --- |
| ROMDAN Elias  TROTTA Nicolas |

1. Introduction

Dans une époque où la quantité d’informations est devenue exponentielle et dans la majorité des cas accessible par tout le monde, nous avons besoin d’un système pour gérer ce flux d’informations qui ne cesse d’augmenter, c’est-à-dire rendre l’accès aux différentes informations plus rapides et plus faciles à faire pour les utilisateurs.

Étant donné que le traitement de toutes les données qui circulent dans le monde est une grande mission à faire pour un tel projet, nous allons se concentrer uniquement sur le traitement des informations en relation avec la métropole de Saint-Étienne. Même au sein d’une métropole il y a une grande quantité d’informations à prendre en compte d’où l’utilité de composer les différents secteurs en ontologie.

Notre groupe a décidé de travailler sur l’ontologie des entreprises, ce travail consiste à collecter les sources exposant des informations (SIRET, raison sociale, chiffre d’affaires, adresse…) sur les entreprises localisées à Saint-Étienne, extraire les données et les insérer dans une Wikibase personnelle qui jouera le rôle d’un Hub.

L’objectif principal de notre application est de répondre aux questions qui peuvent être posées par les utilisateurs sur la métropole de Saint-Étienne. Pour atteindre cet objectif, quelques piliers doivent être mise en place auparavant d’où la composition de l’application en 3 grands modules :

* Extraction des données des sources entrées selon une structure définie et insertion de ces données dans une base propre à l’application.
* Insertion des données récupérées de la base de l’application dans la Wikibase personnelle.
* Interrogation de la Wikibase personnelle à travers l’API QAnswer.

2. Technologies utilisées

Spring : Framework qui permet de définir l'infrastructure d'une application Java et de rendre le développement et les tests en back-end plus facile à réaliser.

Thymeleaf : Moteur de Template utilisé pour la génération de vue (HTML/XML…) pour les applications web basées sur le modèle MVC.

H2 : Une base de données en mémoire légère utile pour stocker les données sans l’utilisation d’un logiciel tiers.

QAnswer : Un système de question/réponse qui permet de fournir une réponse à une question posée en fouillant dans l’une des bases du système.

Wikibase : Base de connaissance qui regroupe les différentes informations sur la métropole de Saint-Étienne.

3. Méthode de travail

Git : Gestionnaire de versioning pour rendre l’intégration de code et la résolution des conflits plus facile à effectuer.

Gantt : Outil pour suivre l’évolution des tâches en fonction du temps.

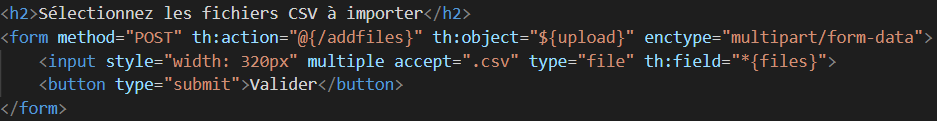
Des livrables après chaque 1 ou 2 semaines pour avoir un aperçu de la direction suivie par le projet. Ces livrables sont la plupart du temps des documents expliquant le mécanisme de l’application sans rentrer dans les détails du code.

4. Extraction des sources de données

Nous avons détaillé dans les documents précédents, la démarche qui sera suivie pour l’extraction des données à haut niveau, c’est-à-dire sans rentrer dans les détails de codage. Nous avons également élaboré la structure qui doit être respecter pour la récupération des données, ainsi que quelques exemples de source à extraire.

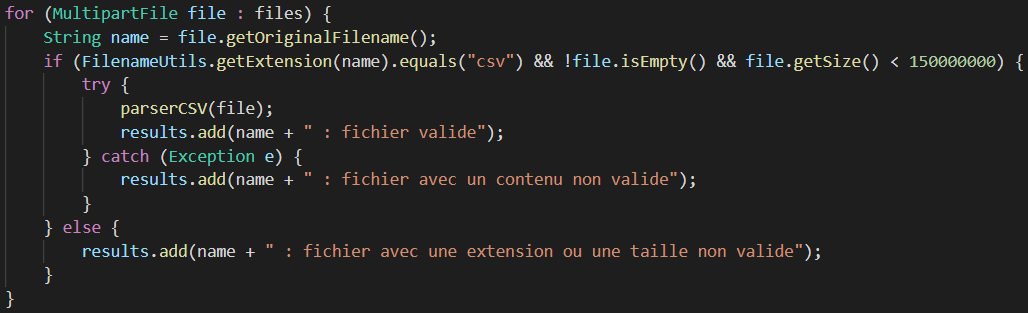
Dans cette partie, nous allons expliquer le mécanisme employé derrière l’extraction et le stockage des données. Etant donné que l’extraction des sources CSV est différente de celle des liens HTML, nous allons aborder chaque parseur en individuel. On abordera uniquement les parties clés du code.

*Extraction des sources CSV*



Ce code HTML donnera la possibilité à un utilisateur de rentrer plusieurs fichiers CSV à la fois à condition que la taille totale des fichiers ne dépasse pas le seuil des 150 mégaoctets et que les fichiers soient de type CSV.

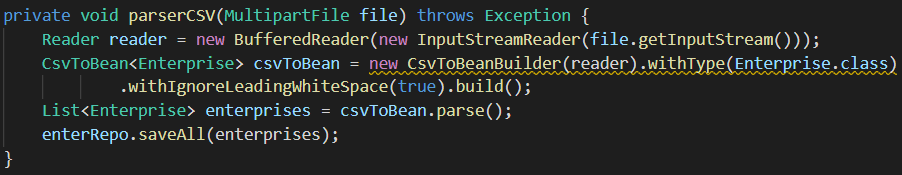
Après avoir fait le choix et l’envoi des fichiers CSV, le serveur prendra la main pour le traitement des fichiers.



Le code Java du dessus s’occupera de vérifier si le fichier respecte les contraintes imposées par notre application, à savoir la taille inférieure à 150 mégaoctets, le contenu non vide et le format CSV.

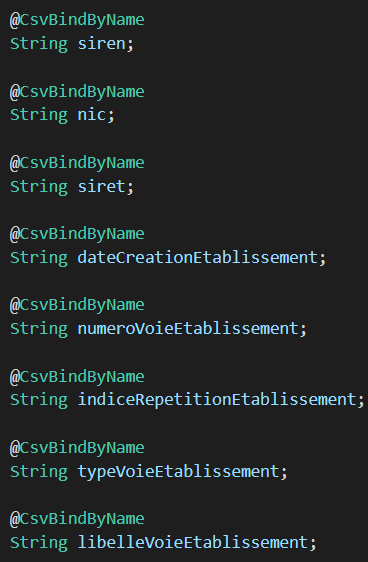
Une fois le test est validé, la méthode « parserCSV » prendra en charge le fichier rentré en paramètre. Si jamais le fichier ne respecte pas la structure mise en place pour l’extraction des données CSV, une exception sera levée et affichée sur l’interface pour alerter l’utilisateur que le fichier est invalide.

L’exception n’empêchera pas les autres fichiers (qui peuvent être valides) d’être traités, chacun à son tour.



Nous avons utilisé la libraire « OpenCSV » vu l’efficacité et la simplicité dont il fait preuve quand il s’agit de parcourir un fichier CSV et récupérer uniquement les données voulues.

On commence par lire le contenu du fichier CSV en paramètre grâce à la classe « BufferedReader ». Ensuite, on utilise la classe « Enterprise » en tant que référence pendant la création de l’objet « csvToBean », c’est-à-dire on récupérera du CSV uniquement les colonnes assignées en tant qu’attribut dans la classe « Enterprise ». Finalement, l’objet est parsé pour récupérer une liste d’objets « Enterprise » qui sera enregistré avec JPA dans la base de données H2.

Le code à gauche donne un petit aperçu des attributs qui existent dans la classe « Enterprise ».

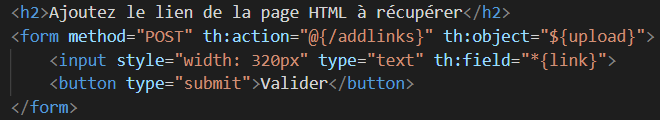
L’annotation « @CsvBindByName » est utilisé pour chercher une colonne dans le fichier CSV qui correspond au nom de l’attribut et inclure les données de cette colonne dans l’objet « csvToBean ».

Dans notre cas, on cherche à récupérer du fichier CSV les colonnes : siren, nic, siret, etc. Toutes les autres données du CSV qui n’ont pas de correspondance dans la classe « Enterprise » seront ignorées.

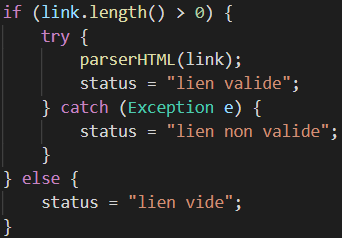
Grâce à l’annotation « @Entity » lié à la classe « Enterprise ». Une table est créée dans la base de données H2, avec les attributs liés au fichier CSV en tant que colonnes de la table.

*Extraction des sources HTML*

A la différence de l’importation en masse des fichiers CSV, on pourra rentrer qu’un seul lien à la fois vers une page HTML. On pourra effectuer cette opération avec le code HTML du dessous :



Au clic sur le bouton « Valider », le texte inséré est envoyé au serveur pour son traitement.



Le code Java du dessus permettra de récupérer dans un premier temps le texte inséré en tant qu’une chaîne de caractères « link ». Un test sera effectué sur la longueur de la chaîne pour éviter de récupérer une chaîne vide.

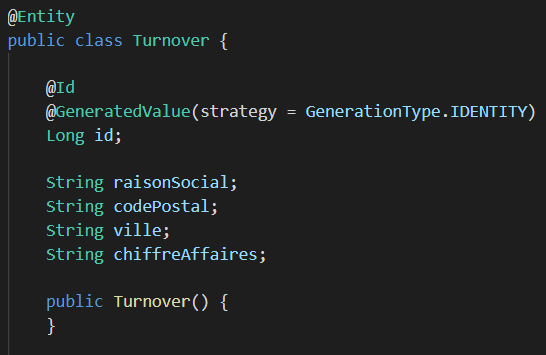
Ensuite, la méthode « parserHTML » s’occupera de traiter le lien. Si jamais le lien ne correspond pas à une page valide ou ne respecte pas la structure définie pour extraire les données d’une source HTML, une exception sera levée et affichée sur l’interface pour alerter l’utilisateur que le lien de la source n’est pas valide.



Nous avons utilisé la libraire « Jsoup » pour traiter le contenu d’un lien récupéré en tant qu’une chaîne de caractères.

On se connecte au lien et on récupère son contenu pour le mettre dans un objet Document. Le tableau qu’on veut récupérer de la source HTML est composé de 4 colonnes identifiables par la classe de la forme « verif\_colX ».

On crée 4 objets « Iterator » qui vont itérer sur les éléments de chaque colonne. A chaque itération, on crée un objet de classe « Turnover » qui va extraire le contenu de chaque élément visité par chaque objet « Iterator ». On rajoute à la fin cet objet « Turnover » à une liste, à la fin du traitement, cette liste sera insérée dans la base de données H2 en utilisant JPA.



Un aperçu des attributs qui composent la classe « Turnover » et qui permettent de récupérer le contenu des éléments qui constituent la table qu’on cherche à extraire de la source HTML.

5. Bot Wikidata

5.1. L’information dans un bocal

Il s’agit d’un espace de wikidata ouvert au seul bénéfice des élèves de Master 1 DSC. Les données insérées et les requêtes effectuées sur celles-ci sont exclusives à cet espace, ainsi nous n’aurons pas accès au reste de la base Wikidata.

5.2. Une question de repères

Dans notre application, la page permettant d’interagir avec l’espace de notre wikidata privée se situe en lien du 4ème onglet de la barre des taches à la rubrique ‘Bot Wikidata’.

Chaque formulaire affichera le résultat de la requête (information ou résultat d’une opération) sous le champ du formulaire employé.

Il n’est pas toujours possible de disposer à l’instant t, alors il est à noter pour les formulaires d’insertion et de mise à jour qu’il est possible de laisser des champs vides afin de les mettre à jour plus tard.

Afin de faciliter et guider le choix des utilisateurs quant à la langue des entités, celles-ci sont pour tous les formulaires des champs de type <select>. L’application gère en effet exclusivement (et jusqu’à une prochaine mise à jour) l’anglais et le français.

Dans tous les formulaires, il sera détecté si une entité existe ou pas et l’utilisateur en sera informé.

La page sur laquelle nous sommes redirigés met à notre disposition 5 formulaires distincts :

* Insertion dans la Wikidata :
  + - Insertion d’une nouvelle entité dans la wikibase privée.
    - Il n’est pas possible ici d’insérer une propriété.
    - Chaque entité insérée n’est relative qu’à une seule langue.
      * Anglais ou Français
      * Il faudra mettre à jour l’entité nouvellement créée pour renseigner celle-ci dans une autre langue.
    - Renseignement des champs suivant :
      * Label de l’entité :
        + Le label représente l’expression qui désignera l’entité dans la wikibase
      * La description :
        + Courte explication concernant le label
      * Langue :
        + Anglais ou français dans notre application.
        + L’entité disposera pour chaque langue d’un label, d’une description et de statements qui sera propre à la langue choisie.

Exemple - Q251 :

|  |  |  |
| --- | --- | --- |
| Langue | Label | Description |
| Français | Super Tomate | Le justicier rouge |
| Anglais | Super Tomato | The red hero |

* Mise à jour dans la Wikidata
  + - Une fois présente dans la wikibase, chacune des entités peut être modifiée à loisir
    - Encore une fois, seules les entités sont concernées et non les propriétés
    - Il ne sera possible à chaque soumission que de modifier les attributs pour une langue donnée.
      * S’il n’existe pas de label pour cette langue, alors la requête ne pourra aboutir
      * Même si un label existe pour une langue des 2 langues mais qu’elle n’est pas sélectionnée alors la requête échouera car le label sera considéré comme absent de la Wikibase.
    - Renseignement des champs suivants :
      * Label de l’entité référente :
        + Label de l’entité que l’on souhaite modifier.
      * Nouveau label :
        + Label remplaçant dorénavant celui de l’entité référente.
      * Nouvelle description :
        + Celle-ci remplacera l’ancienne.
      * Langue :
        + Anglaise ou Française
    - Exemple :
      * Sur l’entité existante Q251 - Super Tomate - Le justicier rouge :

|  |  |  |  |
| --- | --- | --- | --- |
| Label de l’entité référente | Nouveau label | Description | Langue |
| Super Tomate | L’écarlate | Les vitamines du fruit, la richesse du légume | fr |

* Transcription code et label
  + - Comme son nom ne dit pas, il s’agit d’un formulaire d’interrogation de la Wikibase
    - Il propose 2 modes majeurs :
      * Code :
        + A partir du code de l’entité, on obtient les informations de celles-ci
      * Label :
        + A partir du label de l’entité, on obtient son code Wikidata
* Insertion d’une entreprise :
  + - Similaire au principe d’une insertion classique évoquée plus haut mais centré sur les entreprises.
    - Les données supplémentaires et le type de données insérées réclament en revanche leur propre méthode dans la classe BotInsert.java
* Affichage Entreprise Code et Label :
  + - Interrogation dans la Wikibase des données des entreprises
    - Similaire au formulaire de transcription mais adapté aux entreprises
    - Si l’entité n’est pas une entreprise, l’utilisateur en sera informé
    - S’il s’agit d’une entreprise, alors nous récupérerons les informations relatives à celle-ci.

6. En arrière-plan

6.1. Salle des machines (Classe BotInsert)

Nous ne reviendrons pas ici sur le fonctionnement du formulaire de la page de l’application, celui-ci est déjà décrit plus haut dans le document. Nous nous focaliserons désormais sur le fonctionnement et l’architecture de cette partie de l’application.

6.2. Aiguillage saturé (Controller)

Le Controller du botWikidata est particulièrement fourni dans l’application, chaque type de requête réclame à lui seul une méthode propre.

6.2.1. Switch

Le méthode ‘bote’ renvoie les informations des requêtes déjà traitées de chaque formulaire.

Chaque méthode associée à un formulaire traitera la requête puis renverra le résultat, sous forme d’un message ou d’un booléen, à la méthode bote, qui elle transmettra celui-ci à la vue afin qu’il soit affiché.

6.2.2. A chaque problème sa méthode

Nous trouvons dans le Controller de l’application 7 méthodes majeures qui traitent chacune d’un type de requête particulier, elles correspondent aux 5 formulaires de la page dédiée aux BotWikidata et aux 2 fonctions chargées de traiter à la chaîne les données extraites grâce aux parsers HTML et CSV.

6.3. Les temps modernes

Nous ne ferons assez de cas du travail laborieux du petit artisan qui patiemment, répète tâche après tâche afin de faire sortir de son établi le meilleur des produits. Malheureusement, c’est long.  
Nous rappellerons que l’objectif du projet est d’extraire des données depuis des sources, de les traiter et, si celles-ci sont absentes, de les rajouter, si celles-ci sont présente, de potentiellement les mettre à jour.  
Aussi déshumanisé que nous apparait le travail à la chaîne, il aura le bon goût de répondre au cahier des charges du projet. Si l’insertion par formulaire, manuelle et consciencieuse était une avancée majeure, celle-ci n’était au final qu’un marchepied bien pratique vers notre véritable but, à savoir automatiser le processus de traitement des données.

Nous avons choisi pour notre projet de traiter le thème de l’entreprise, plus précisément celui de l’entreprise stéphanoise. Se procurer des données fiables et exploitables n’est pas toujours une mince affaire et nous aurons du mal à cacher que nous n’avons pas eu toutes les difficultés du monde à trouver des listings utilisables recensant les entreprises de Saint-Étienne mais également les informations sur elles. Nous avons à notre disposition 2 sources majeures, plus précisément un listing CVS et un listing HTML et le challenge consistera à extraire des données à grande échelle des prenant la forme de rubriques dans des colonnes et ce sur des centaines voire des milliers de lignes.

Ce traitement à la chaîne est la suite directe de tout le travail fait sur les parsers dont l’analyse a été faite plus haut dans le présent rapport.

6.3.1. Les joies du tableur : Le CSV

* Dans le Controller de l’application, cette fonctionnalité est traitées par la méthode ‘botinsertEntrepriseHTML()’.
* Du fichier CVS, sur les dizaines de types d’informations contenues pour chaque entreprise, nous n’en n’extrayons que 7 :
  + label
  + description
  + lang
  + ville
  + codePostal
  + SIREN
  + SIRET
  + CA (chiffre d’affaire mis à 0 car non présent dans le fichier)
* Pour chaque entreprise présente dans le fichier, ces 7 informations seront extraites et parfois traitées, avant d’être insérées dans la Wikibase.
* Ce processus d’insertion à la chaîne se distingue notamment par le traitement qu’il applique à certaines de ces informations.
  + Le point le plus notable ces attributs concerne le cas particulier de la ville, dont l’insertion dans la wikibase est plus complet que les autre données :
    - Si l’on se réfère à l’ontologie relative au thème de l’entreprise, chaque ville est l’un des attributs de chaque entité ‘Entreprise’.
    - Une ville est aussi une entité dans la Wikibase, aussi nous avons 2 types de traitement possible à chaque insertion d’une entreprise :
      * Si la ville est déjà présente dans la wikibase, alors l’entité ‘ville’ existante est attribuée au Statement ‘est localisé dans la ville de’ de l’entreprise inscrite.
      * Si elle n’existe pas dans la Wikibase alors l’entité de cette nouvelle ville est créée dans un 1er temps puis elle est attribuée de manière traditionnelle au Statement.

6.3.2. De balise en alise : La page HTML

L’insertion des données extraites du fichier HTML fonctionne sur un principe similaire à celui du CSV. Encore une fois, si le parsing à partir des fichiers sources est différent, les données sont à ce stade stockées dans la base de données dans une qui leur est propre. L’insertion à la chaîne se fait à partir de ces informations extraites cette fois-ci depuis la base H2. La principale différence tient ici au nombre et au type de données traitées, il n’y a ici que 4 données à insérer, le reste étant transformé en valeur nulle ou par défaut.

On trouve :

* Le label (La raison sociale)
* Le CP (code postal)
* La ville
* Le CA (chiffre d’affaire)

Le site recensant ces informations, [www.verif.com](https://www.verif.com/Hit-parade/01-CA/01-Par-departement/42-Loire), contient des erreurs de format de certaines données, notamment dans la colonne des villes. Les caractères accentués ne s’affichent pas correctement et nous avons par exemple

* Saint-Ã‰tienne -> au lieu de Saint-Étienne
* La TalaudiÃ¨re -> au lieu de La Talaudière
* L’Ã‰trat -> au lieu de L’Étrat  
  Aussi faut-il veiller à détecter ces problèmes et à les traiter avant la phase d’insertion.

6.4. Du haut de son perchoir

6.4.1. La peur du néant

Même ce qui n’existe pas peut être néfaste, l’informatique nous le prouve sans cesse.  
Le Null Pointer Exception, si l’on n’y prend pas garde, se rappellera à notre bon souvenir à toutes les occasions : interrogation, insertion, mise à jour, … tous les prétextes sont bons.

Le drame ne survient jamais au moment où l’on s’y attend, ce n’est pas lorsqu’on va cherher l’information dans la wikidata que l’on risque le pire mais lorsqu’on devra traiter ses données que les ennuis vont commencer.

La classe de l’application BotInsert.java représentera au cours de toutes ses fonctions le principale challenge car après chaque transformation en objet Java, on pourra potentiellement chercher à traiter celui-ci alors qu’il ne contient rien. Le Null Pointer Exception devra être éviter à tout prix afin d’éviter l’apparition sur son navigateur d’un message d’erreur peu compatible avec le caractère didactique d’une application de type web.

6.4.2. Usurpation d’identité

Si notre but est bien de chercher et insérer des informations dans la wikibase, il est aussi accessoirement de faire en sorte qu’elles soient les bonnes. Nous courons le risque en permanence d’obtenir les données d’une entité dont le label ou la langue nous a induit en erreur.

Mal traitées, des données risquent de prendre la place d’autres :

* Le traitement des langues (ici Anglais et Français) est un souci permanent car comme nous l’avons dit plus haut dans le rapport, chaque entité possède des attributs indépendants suivant la langue donnée.
  + L’anglais étant la langue de base, il arrivera qu’une insertion dans la Wikibase soit :
    - Effectuées mais pas détectée lors des recherches
    - Annulée car une entité au label identique sera trouvée dans la base (même d’une autre langue)

7. Conclusion

Nous avons manipulé le concept d’interopérabilité dans ce projet vu qu’on a mis en communication plusieurs technologies hétérogènes. A commencer par l’extraction des sources de données vers une base de données locale à l’application. De prendre les données dans cette base et les insérer dans une wikibase personnelle à l’Université. Pour finir par l’utilisation de l’API QAnswer pour traiter les différentes questions des utilisateurs en fouillant dans la wikibase.