
WEB DES DONNÉES

TP N° 7 – SPARQL

Romain LELONG – romain.lelong@gmail.com

Avant de commencer ...

Outre le présent sujet, les fichiers suivant doivent être récupérés avant de débiter le TP :

- `guillaume_canet_simple.rdf`
- `marion_cotillard_simple.rdf`

Ce TP ne sera pas noté et aucun compte-rendu n'est demandé. Vous pouvez le réaliser en binôme ou seul. Les divers éléments abordés dans celui-ci pourront néanmoins faire l'objet d'une micro-évaluation prochaine.

1 Outil DBPedia en ligne

L'outil en ligne accessible via l'URL <https://dbpedia.org/sparql> mène à l'endpoint SPARQL de *DBpedia*, un service en ligne qui permet d'interroger la base de connaissances de *DBpedia* en utilisant le langage de requête SPARQL.

Afin d'alléger les requêtes SPARQL on emploiera les namespaces suivant :

namespace	URL
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
res	http://dbpedia.org/resource/
prop	http://dbpedia.org/property/
xsd	http://www.w3.org/2001/XMLSchema#
onto	http://dbpedia.org/ontology/

Étape n° 1 : Que permet de rechercher la requête suivante :

```
SELECT ?s ?p
WHERE {
  ?s ?p <http://dbpedia.org/resource/Marion_Cotillard>
}
```

Étape n° 2 : Réécrire la requête précédente en faisant usage du namespace **res**.

Étape n° 3 : Parmi les résultats renvoyés, observez ceux qui font usage des prédicats suivants :

- <http://dbpedia.org/property/partner>
- <http://dbpedia.org/ontology/starring> ou <http://dbpedia.org/property/starring>

En déduire la « signification » de ces prédicats.

Étape n° 4 : Quelle requête permet de lister les films dans lesquels Marion Cotillard **ou bien** Guillaume Canet ont **joué l'un ou l'autre** ?

Étape n° 5 : Quelle requête permet de lister les films dans lesquels Marion Cotillard **et** Guillaume Canet ont **tous les deux joués** ?

Étape n° 6 : Modifier la requête précédente pour obtenir les titres français de ces films.

Indication : Utiliser le prédicat <http://www.w3.org/2000/01/rdf-schema#label> et la fonction `lang(...)`

Étape n° 7 : Déterminer une requête permettant de construire un graphe RDF préservant uniquement les informations relatives à *Marion Cotillard* et *Guillaume Canet* véhiculées par trois prédicats suivants :

- <http://dbpedia.org/property/partner>
- <http://dbpedia.org/ontology/starring>
- <http://dbpedia.org/property/starring>

Étape n° 8 : Enregistrer, au format RDF/XML, la réponse renvoyée par le moteur SPARQL dans un fichier nommé `playsin.rdf`.

Étape n° 9 : Vérifier la validité de votre graphe RDF avec le service de validation RDF du W3C ¹

2 Le triplestore Apache Jena Fuseki

Apache Jena Fuseki est un serveur SPARQL. Apache Jena Fuseki c'est donc un triplestore, c'est à dire agit un serveur de base de données RDF qui permet de stocker, interroger et manipuler des données RDF via le langage de requête SPARQL. Il permet donc d'interroger un/des graphe(s) RDF préalablement chargé(s) dans le serveur. Il dispose d'une interface d'administration. Fuseki fournit les protocoles SPARQL 1.1 pour les requêtes (i.e. partie SPARQL Query) mais permet également les mises à jour (i.e. la partie SPARQL Update), ainsi que le protocole SPARQL Graph Store.

¹<https://www.w3.org/RDF/Validator/>

Étape n° 10 : Téléchargée la dernière version d'Apache Jena Fuseki à partir de l'URL : <https://jena.apache.org/>. L'archive téléchargé devrait alors posséder un nom du type `apache-jena-fuseki-x.x.x.tar.gz` ou bien `apache-jena-fuseki-x.x.x.zip`.

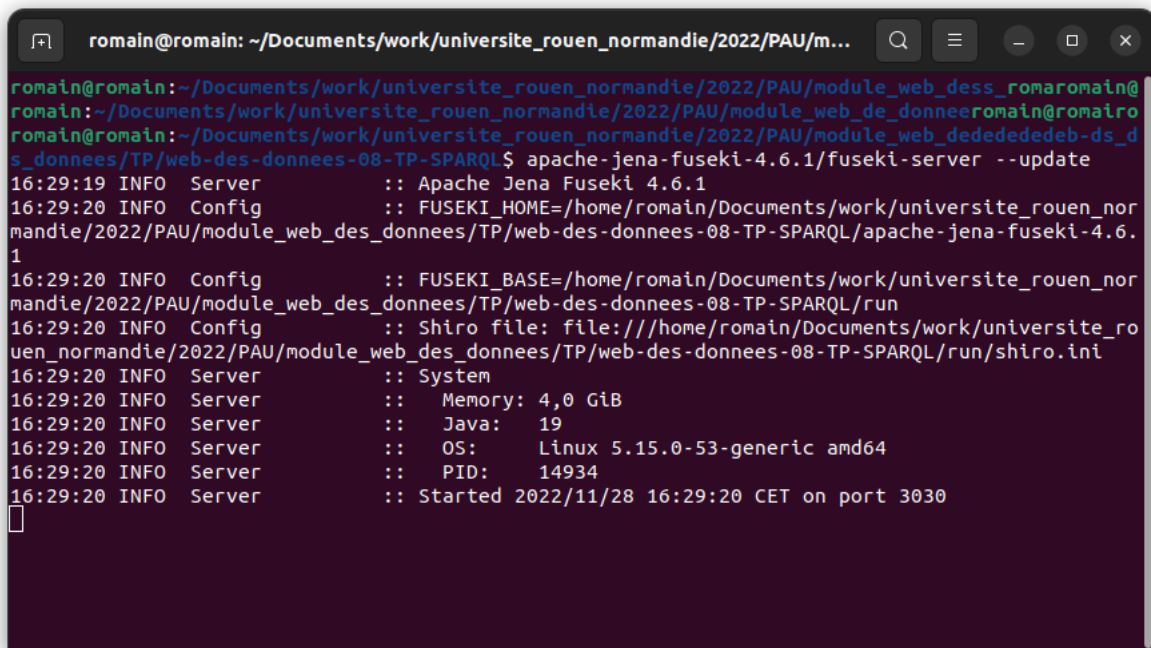
Attention : *Jena5 nécessite Java 17 ou une version plus récente de Java pour fonctionner. Si vous possédez une version antérieure de Java, vous pouvez, si vous ne souhaitez pas mettre à jour votre installation Java, utiliser une version antérieure de l'outil par exemple `apache-jena-fuseki-3.9.0.zip` disponible dans la section [Previous releases](#).*

Étape n° 11 : Décompresser l'archive `apache-jena-fuseki-x.x.x.tar.gz`.

Étape n° 12 : Lancer le script `fuseki-server` qui se trouve dans le dossier obtenu après décompression. Cela peut être fait à l'aide de la ligne de commande :

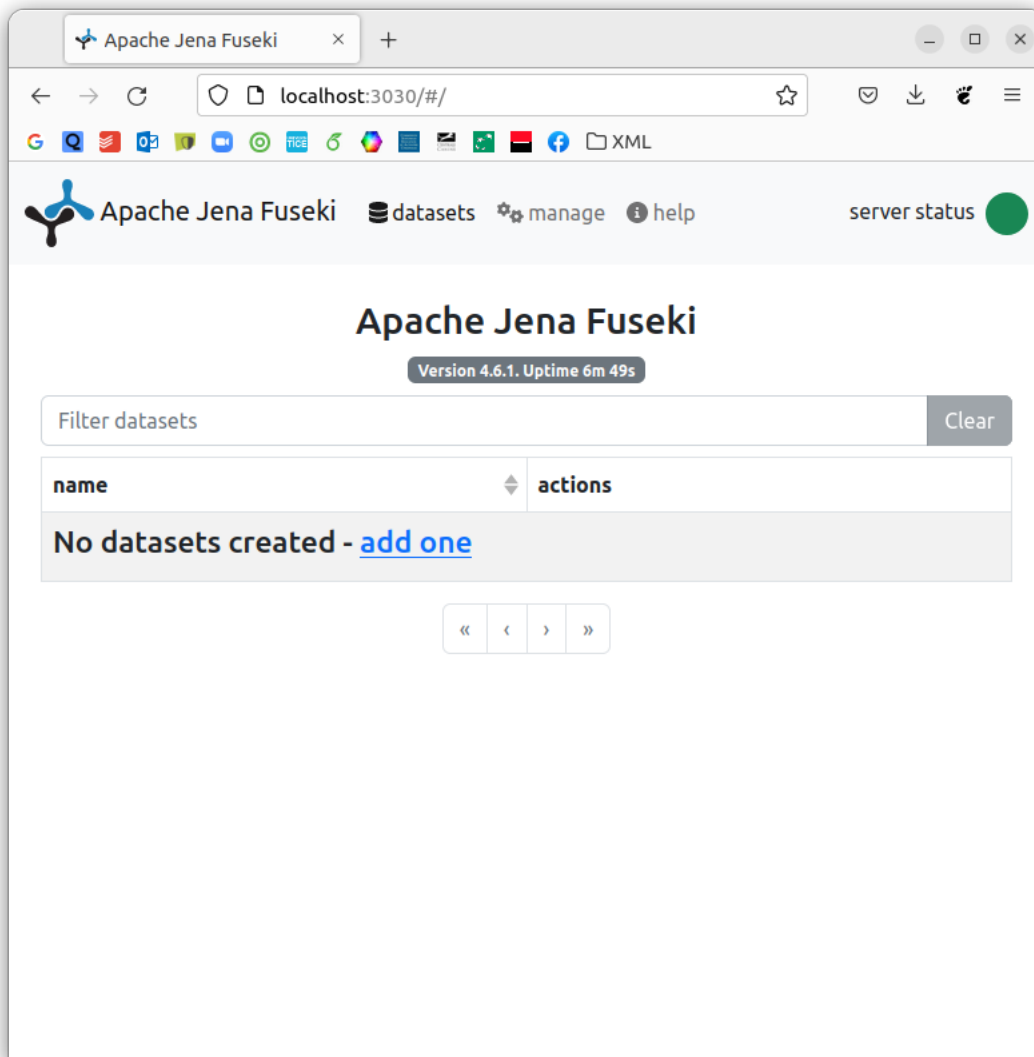
```
[...]/apache-jena-fuseki-4.6.1/fuseki-server --update
```

Un terminal s'ouvre alors vous indiquant le démarrage du serveur et le port sur lequel il est accessible :



```
romain@romain: ~/Documents/work/universite_rouen_normandie/2022/PAU/m...
romain@romain:~/Documents/work/universite_rouen_normandie/2022/PAU/module_web_dess_romaromain@
romain:~/Documents/work/universite_rouen_normandie/2022/PAU/module_web_de_donneeromain@romairo
romain@romain:~/Documents/work/universite_rouen_normandie/2022/PAU/module_web_dededededeb-ds_d
s_donnees/TP/web-des-donnees-08-TP-SPARQL$ apache-jena-fuseki-4.6.1/fuseki-server --update
16:29:19 INFO Server      :: Apache Jena Fuseki 4.6.1
16:29:20 INFO Config      :: FUSEKI_HOME=/home/romain/Documents/work/universite_rouen_nor
mandie/2022/PAU/module_web_des_donnees/TP/web-des-donnees-08-TP-SPARQL/apache-jena-fuseki-4.6.
1
16:29:20 INFO Config      :: FUSEKI_BASE=/home/romain/Documents/work/universite_rouen_nor
mandie/2022/PAU/module_web_des_donnees/TP/web-des-donnees-08-TP-SPARQL/run
16:29:20 INFO Config      :: Shiro file: file:///home/romain/Documents/work/universite_ro
uen_normandie/2022/PAU/module_web_des_donnees/TP/web-des-donnees-08-TP-SPARQL/run/shiro.ini
16:29:20 INFO Server      :: System
16:29:20 INFO Server      :: Memory: 4,0 GiB
16:29:20 INFO Server      :: Java: 19
16:29:20 INFO Server      :: OS: Linux 5.15.0-53-generic amd64
16:29:20 INFO Server      :: PID: 14934
16:29:20 INFO Server      :: Started 2022/11/28 16:29:20 CET on port 3030
```

Étape n° 13 : Accéder à l'interface d'administration du serveur via l'URL : <http://localhost:3030>



Étape n° 14 : Créer une base de donnée RDF ou plus généralement un *dataset* persistant qui sera dédié au TP.

Étape n° 15 : Charger dans ce dataset le graphe RDF constitué et enregistré aux étapes n° 7 et 8. Vous pouvez si vous préférez intégrer à la place les deux graphes suivants :

- guillaume_canet_simple.rdf
- marion_cotillard_simple.rdf.

Étape n° 16 : On souhaite maintenant simplifier le graph RDF contenu dans notre dataset. En utilisant un nouveau prédicat afin notamment d'utiliser un même et unique prédicat à la place de `prop:starring` et `onto:starring`. En somme, dans le nouveau graphe, on souhaite ne plus utiliser les prédicats d'origine mais les remplacer comme suit :

Prédicat DBpedia	→	Nouveau prédicat
<code>http://dbpedia.org/property/partner</code>	→	<code>http://www.univ-rouen.fr/conjoint</code>
<code>http://dbpedia.org/ontology/starring</code>	→	<code>http://www.univ-rouen.fr/playsIn</code>
<code>http://dbpedia.org/property/starring</code>	→	<code>http://www.univ-rouen.fr/playsIn</code>

Déterminer la requête permettant de construire ce graphe.

Étape n° 17 : Intégrer, à l'aide d'une requête SPARQL, ces triplets au dataset du TP.

Attention : Pour tous les ordres SPARQL Update, il est nécessaire de positionner le champ « SPARQL Endpoint » sur « `/dataset/update` » et non sur « `/dataset/query` »

Étape n° 18 : Supprimer, à l'aide d'une requête SPARQL, les triplets dont définis à l'aide les prédicats d'origine (i.e. colonne « Prédicat DBPedia »).