|  |
| --- |
|  |
| Rapport - Sujet B3 : Requêtes et visualisation de données issues de Dbpedia |
|  |
|  |
| **Peter Serge** |
| **9/1/2015** |

Formation MAS-ICT CAS : WEB SEMANTIQUE

Julien Tscherrig, Omar Abou Khaled et Maria Sokhn

|  |
| --- |
| Rapport de réalisation d’une application de mise en relation de données issues de dbpedia. |

|  |
| --- |
|  |

Table des matières

[1. Introduction 2](#_Toc412025018)

[1.1 Rappel sur les objectifs 2](#_Toc412025019)

[1.2 Rappel sur les contraintes techniques 2](#_Toc412025020)

[2. Analyse 2](#_Toc412025021)

[2.1 Liste des artistes (Query 1.1) 3](#_Toc412025022)

[2.2 Détail sur un artiste (Query 1.2) 4](#_Toc412025023)

[2.3 Recherche d’artistes ou bands liés (Query 1.3) 4](#_Toc412025024)

[2.4 Recherche des genres (Query 1.4) 5](#_Toc412025025)

[3. Architecture technique 5](#_Toc412025026)

[3.1 Composants 6](#_Toc412025027)

[3.2 Dbpedia et sparql 6](#_Toc412025028)

[3.3 Jena 6](#_Toc412025029)

[3.4 Spring Core & Ctx 6](#_Toc412025030)

[3.5 Spring MVC 6](#_Toc412025031)

[3.6 Junit & Spring test 6](#_Toc412025032)

[3.7 AngularJS 6](#_Toc412025033)

[4. Mise en route 6](#_Toc412025034)

[4.1 Installation de Java 6](#_Toc412025035)

[4.2 Installation de Maven 6](#_Toc412025036)

[4.3 Installation de Eclipse 6](#_Toc412025037)

[4.4 Installation de Tomcat 6](#_Toc412025038)

[5. Réalisation 7](#_Toc412025039)

[5.1 Backend 7](#_Toc412025040)

[5.2 Frontend 7](#_Toc412025041)

[6. Conclusion 7](#_Toc412025042)

[7. Références 8](#_Toc412025043)

# Introduction

Ce document est un rapport pour la réalisation d’un mini projet ; dans le cadre de la formation MAS-ICT CAS : WEB SEMANTIQUE Julien Tscherrig, Omar Abou Khaled et Maria Sokhn.

Notre application est composée de deux parties principales, le backend, application web java permettant d’opérer comme fournisseur de données et cachant la complexité des requêtes sparql ; et un frontend écrit en JavaScript avec l’aide du Framework structurant AngularJS. Ces deux principales composantes seront détaillées plus tard dans ce document.

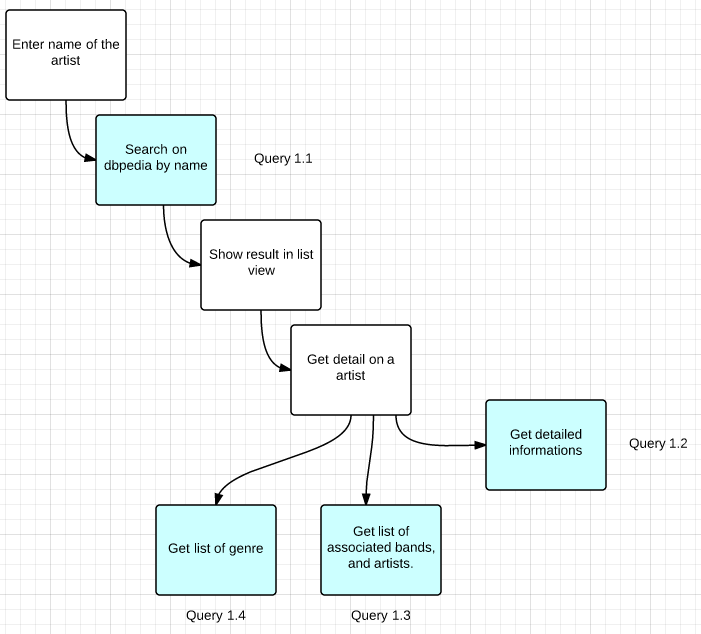
## Rappel sur les objectifs

## Rappel sur les contraintes techniques

# Analyse

L’application obtiendra ces données via des requêtes sparql, faite directement sur les sources de données de dbpedia.org.

Le workflow principal correspond à ceci :



L’application commence par une recherche sur la source de données <http://dbpedia.org/ontology/MusicalArtist> et affichera une liste d’artiste (Query 1.1). On pourra ensuite afficher des détails supplémentaires dans un écran dédié, afficher la liste de genre (Query 1.4) associés ainsi que les artistes (Query 1.2) et groupes liés (Query 1.3).

## Liste des artistes (Query 1.1)

Nous voulons dans un premier temps obtenir la liste des artistes par leur nom (ou fragment).

prefix prop: <http://dbpedia.org/property/>

prefix owl: <http://dbpedia.org/ontology/>

prefix res: <http://dbpedia.org/resource/>

prefix rdfs: [http://www.w3.org/2000/01/rdf-schema#](http://www.w3.org/2000/01/rdf-schema)

select distinct ?artist ?name

where {

?artist a owl:MusicalArtist .

?artist rdfs:label ?name .

FILTER (regex(?name, 'Mark Knopfler', 'i')

&& langMatches(lang(?name), 'en'))

} LIMIT 100

A noter que notre source de données est : <http://dbpedia.org/ontology/MusicalArtist> qui est une sous-classe de <http://dbpedia.org/ontology/Artist>



Dans cette requête nous filtrons les résultats qui sont en anglais en utilisant la fonction FILTER

FILTER (langMatches(lang(?name), 'en'))

La sélection des données se fait aussi en utilisant FILTER avec l’option i pour ne pas tenir compte de la case:

FILTER (regex(?name, 'Mark Knopfler', 'i'))

## Détail sur un artiste (Query 1.2)

Pour ce faire il faut utiliser l’URI de la ressource dans la sélection des données comme ceci et la propriété sameAs

?artist owl:sameAs? **<http://dbpedia.org/resource/Mark\_Knopfler>** .

Attention aussi aux champs null, qui ici sont traité avec la fonction OPTIONAL afin de sortir les données partielles de la requête.

OPTIONAL { ?artist owl:birthPlace ?birthPlace }

Ce qui fait une requête comme ceci:

prefix prop: <http://dbpedia.org/property/>

prefix owl: <http://dbpedia.org/ontology/>

prefix res: <http://dbpedia.org/resource/>

prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select \*

where {

?artist owl:sameAs? **<http://dbpedia.org/resource/Mark\_Knopfler>** .

?artist rdfs:label ?name .

OPTIONAL { ?artist owl:birthDate ?birthDate }

OPTIONAL { ?artist owl:birthPlace ?birthPlace }

OPTIONAL { ?artist owl:thumbnail ?image}

OPTIONAL { ?artist prop:shortDescription ?shortDescription }

OPTIONAL { ?artist owl:abstract ?abstract }

OPTIONAL { ?artist prop:website ?website }

OPTIONAL { ?artist prop:yearsActive ?yearsActive }

FILTER (langMatches(lang(?name), 'en')

&& langMatches(lang(?abstract), 'en'))

} LIMIT 1

## Recherche d’artistes ou bands liés (Query 1.3)

Là aussi la propriété sameAs permet de faire la sélection au niveau de la ressource. Ou procèdera de la même manière pour les propriétés fournissant une liste de valeurs come owl :associatedBand, owl :associatedArtist.

prefix prop: <http://dbpedia.org/property/>

prefix owl: <http://dbpedia.org/ontology/>

prefix res: <http://dbpedia.org/resource/>

prefix rdfs: [http://www.w3.org/2000/01/rdf-schema#](http://www.w3.org/2000/01/rdf-schema)

select distinct \* where { ?artist owl:sameAs? http://dbpedia.org/resource/Mark\_Knopfler> .

?artist rdfs:label ?name .

?artist owl:associatedBand ?associatedBand.

?associatedBand rdfs:label ?assName .

FILTER (langMatches(lang(?name), 'en')

&& langMatches(lang(?assName), 'en'))

} LIMIT 100

## Recherche des genres (Query 1.4)

Voici un exemple de requête, similaire au point précédent en utilisant la propriété owl:genre.

prefix prop: <http://dbpedia.org/property/>

prefix owl: <http://dbpedia.org/ontology/>

prefix res: <http://dbpedia.org/resource/>

prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select \*

where {

?artist owl:sameAs? <http://dbpedia.org/resource/Mark\_Knopfler> .

?artist rdfs:label ?name .

OPTIONAL { ?artist owl:genre ?genre}

OPTIONAL { ?genre rdfs:label ?genrelabel}

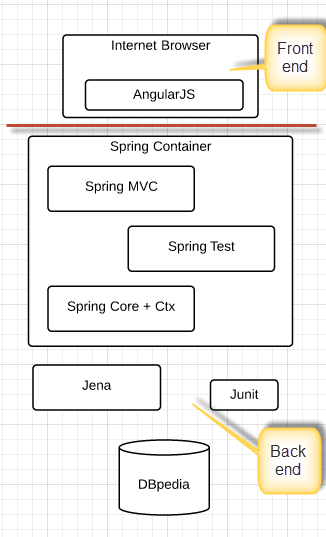
FILTER (langMatches(lang(?name), 'en')

&& langMatches(lang(?genrelabel), 'en') )

} LIMIT 10

# Architecture technique

Nous utiliserons ici Spring pour créer une application Web permettant de mettre en relation les diverses couches. Spring MVC nous permet de construire une couche service et de l’exposer pour qu’elle puisse être utilisée du côté front.



## Composants

### Dbpedia et sparql

### Jena

### Spring Core & Ctx

### Spring MVC

### Junit & Spring test

### AngularJS

# Mise en route

Les outils nécessaires pour effectuer ces développements sont :

|  |  |  |
| --- | --- | --- |
| Composants | Version | Location |
| Java JDK | 1.7+ | <http://www.oracle.com/technetwork/java/javase/downloads/index.html> |
| Maven | 3.0 et plus | <http://maven.apache.org/download.cgi> |
| Tomcat | 8 | <http://tomcat.apache.org/download-80.cgi> |
| Eclipse | Luna | <https://www.eclipse.org/downloads/> |

## Installation de Java

## Installation de Maven

## Installation de Eclipse

## Installation de Tomcat

# Réalisation

## Backend

## Frontend

# Conclusion

# Références

Dbpedia :

Livre sur AngularJS

* AngularJS de Brad Green, Shyam Seshadri, publié par O'Reilly en avril 2013; ISBN:978-1-4493-4485-6
* Mastering Web Application Development with AngularJS de Pawel Kozlowski et Peter Bacon Darwin, publié par Packt Publishing en Août 2013; ISBN 978-1-78216-182-0

Spring

* http://projects.spring.io/spring-data/
* http://www.programming-free.com/2014/07/spring-data-rest-with-angularjs-crud.html