RDF en action avec Java et JENA

B. NGUYEN
STI4A – INSA Centre Val de Loire

Ressources : Tutoriels JENA « officiels »

Plan

- Présentation du logiciel JENA
- Création et manipulation de données RDF
- Navigation de données RDF
- Intégration et Collections
- Littéraux
- Raisonnement

Logiciel : JENA

- Apache JENA

  https://jena.apache.org/
- Télécharger le logiciel (30 Mo) – rien à installer, on dézippe simplement. Ça marche avec Java
- Ce sont les données qui sont volumineuses !
  - Exemples à récupérer sur :

    http://jena.apache.org/tutorials/sparql_data/
  - Données pour le tuto Java :

  https://github.com/apache/jena/tree/master/jena-core/src-examples
- Variables d'environnement à régler
  - JENA_HOME = répertoire d'installation
  - Path = pour des raisons pratiques

Manipulation

- Via un programme JAVA (à la JDBC)
  - Inclusion nécessaire des bibliothèques JENA
- Directe (avec les batches) : pour de l’interrogation SPARQL.

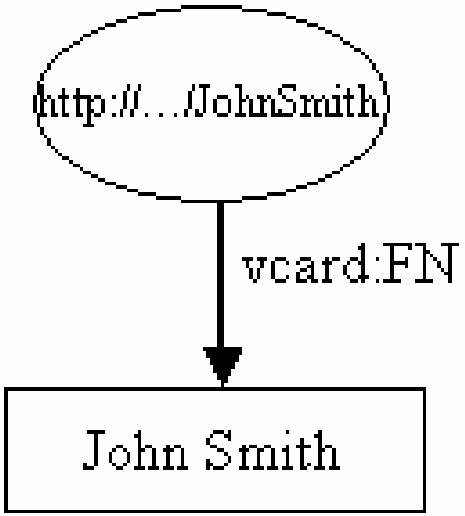
Création de données RDF

Les données

- Représentation RDF de VCard
- Voir par exemple : http://norman.walsh.name/2005/12/05/vcard

Premier exemple

- La ressource (ellipse) est identifiée par son URI
- La ressource possède des propriétés, qui sont les arcs étiquetés.
- Chaque propriété possède une valeur, ici un littéral « John Smith »



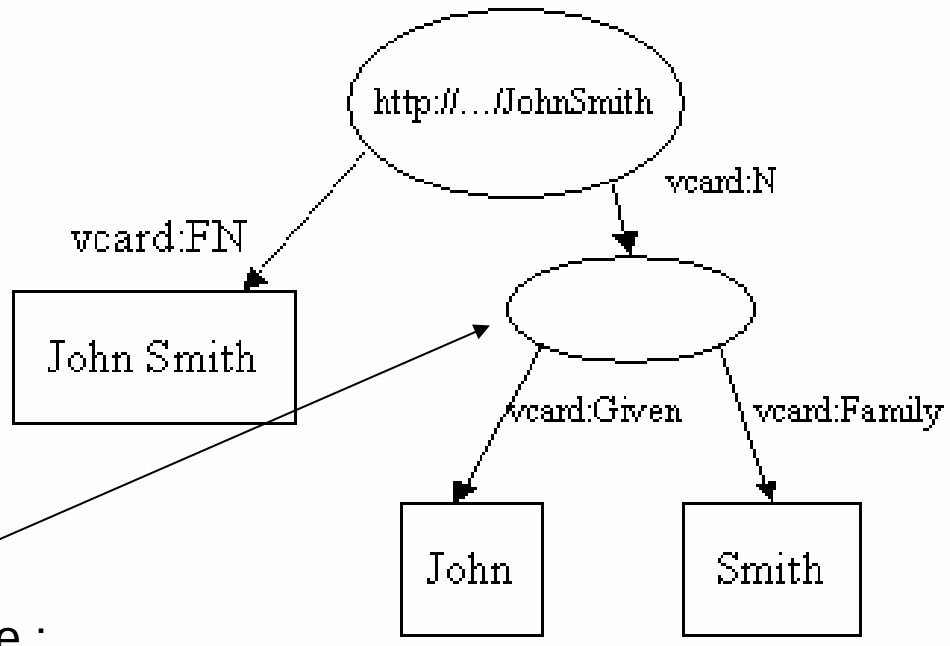
Premier exemple

- Voir sous eclipse
- Un Graphe = Model

- La ressource John Smith est alors créée et une propriété y est ajoutée. La propriété est fournie par une classe « constante » de VCARD, qui détient les objets représentant toutes les définitions dans le schéma VCARD.
- Jena fournit des classes constantes pour d'autres schémas bien connus, comme RDF et RDF Schema eux-mêmes, Dublin Core et DAML.

Deuxième exemple

Visualisons le contenu
avec un
System.out.println



Nœud anonyme :
Un identifiant sera généré par JENA

Navigation

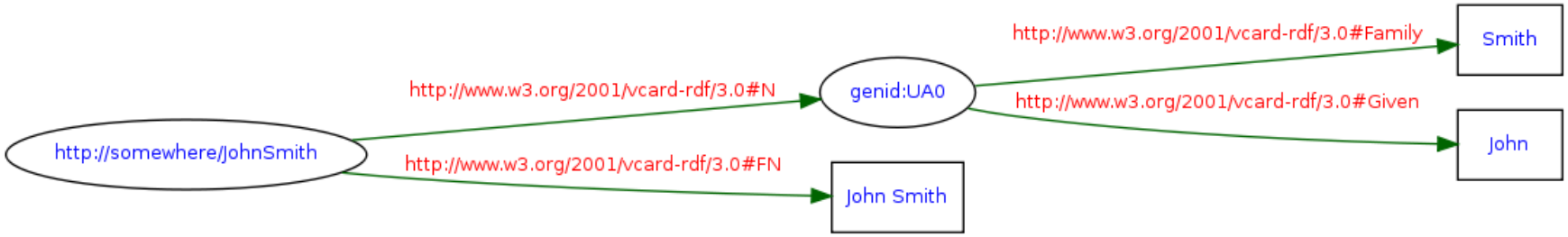
```
// liste des déclarations dans le modèle
StmtIterator iter = model.listStatements();
// affiche l'objet, le prédicat et le sujet de chaque déclaration
while (iter.hasNext()) {
        Statement stmt = iter.nextStatement();
        // obtenir la prochaine déclaration
        Resource subject = stmt.getSubject();
        // obtenir le sujet
        Property predicate = stmt.getPredicate();
        // obtenir le prédicat
        RDFNode object = stmt.getObject();
        // obtenir l'objet
        System.out.print(subject.toString());
        System.out.print(" " + predicate.toString() + " ");
        if (object instanceof Resource) { System.out.print(object.toString()); }
        else { // l'objet est un littéral
                System.out.print(" \"" + object.toString() + "\""); }
        System.out.println(" .");
}
```

Ecriture

- Tout simplement avec model.write(OutputStream)
- Plusieurs paramètres sont possibles avec ce modèle :
  - model.write(System.out, "RDF/XML-ABBREV");
  - model.write(System.out, "N-TRIPLE");
- Validité à tester sur : http://www.w3.org/RDF/Validator/

Ecriture

- Tout simplement avec model.write(OutputStream)
- Plusieurs paramètres sont possibles avec ce modèle :
  - model.write(System.out, "RDF/XML-ABBREV");
  - model.write(System.out, "N-TRIPLE");
- Validité à tester sur : http://www.w3.org/RDF/Validator/



Lecture de fichiers

- Même principe…

model.read(in, null);

- Le 2e paramètre est l'URI de base utilisée pour résoudre les URI relatives.

Définition des namespaces

Les namespaces peuvent être laissés comme par défaut (dans ce cas ils sont générés automatiqument), ou bien on peut explicitement leur donner un nom avec la commande :

setNsPrefix(String prefix, String URI)

Le préfixe doit être valide en XML.

Naviguer un graphe RDF avec JENA

Naviguer un graphe

- Etant donné une URI, on peut récupérer une ressource avec :
  - Model.getResource(String uri)

  E.g. Resource vcard = model.getResource(johnSmithURI);
- Pour récupérer une propriété :
  - Resource.getProperty(Property p)

  /!\ On récupère le TRIPLET ! On doit ensuite indiquer la partie qui nous intéresse (subject, predicate ou object)

  Resource name = (Resource) vcard.getProperty(VCARD.N) .getObject();

  Ou pour un littéral:

  String fullName = vcard.getProperty(VCARD.FN) .getString();

Propriétés identiques multiples

- Considérons la propriété « surnom »

  vcard.addProperty(VCARD.NICKNAME, "Smithy")
  vcard.addProperty(VCARD.NICKNAME, "Adman");
- L’appel à getProperty est indéterminé ! (e.g. retourne le premier… ou le dernier)
- Il faut utiliser listProperties(VCARD.NICKNQME)
- D’une manière générale listProperties() donne l’ensemble des propriétés d’une ressource.

Et ensuite … ?

- On peut naviguer !
- Condition : connaître au moins une URI d’entrée.

Interrogation sans connaître d’URI

- Si on ne connaît rien :
  - Model.listStatements() donne tout
  - Model.listSubjects() donne les sujets
- En fait model.listStatements(Selector s) peut être paramétré par un Selector. Il en existe un certain nombre définis par défaut.
- E.g. Si on cherche des ressources dont on connaît une propriété :
  - Model.listSubjectsWithProperty(Property p, RDFNode o)

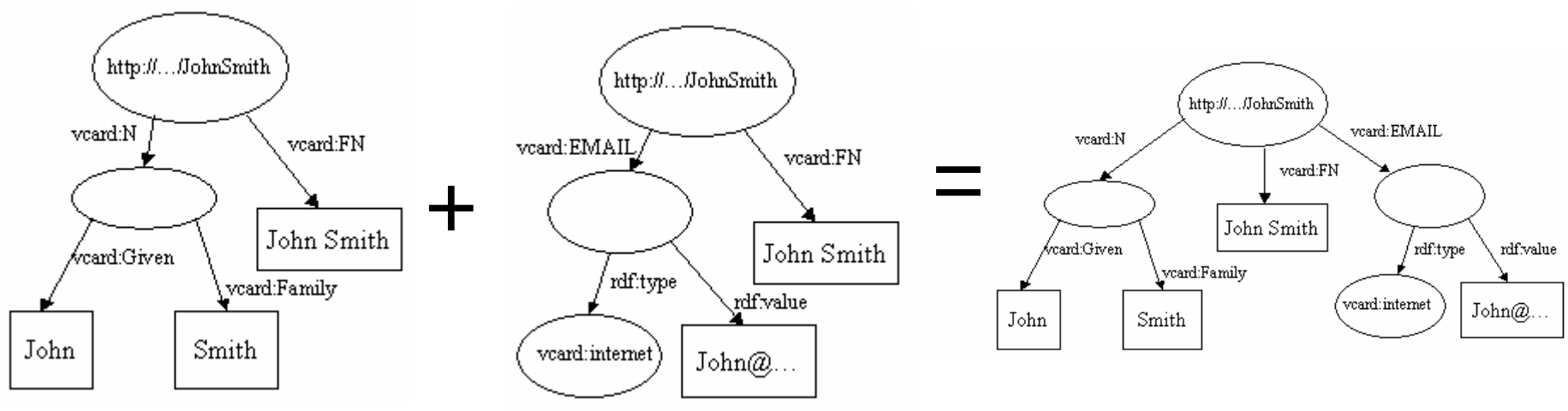
Définition d’un Selector

- Selector selector = new SimpleSelector(subject, predicate, object)
- Une valeur null correspond à aucune contrainte de sélection.
- E.g.
  - Selector selector = new SimpleSelector(null, null, null); selectionne toutes les données de la base de connaissances.
  - Selector selector = new SimpleSelector(null, VCARD.FN, null); Sélectionne tous les triplets ayant comme propriété FN

Intégration et collections

Intégration d'informations

Comme les ressources sont définies à l'aide d'URIs, il est trivial de les intégrer (fusionner).



Et c'est facile avec JENA : Model model = model1.union(model2);

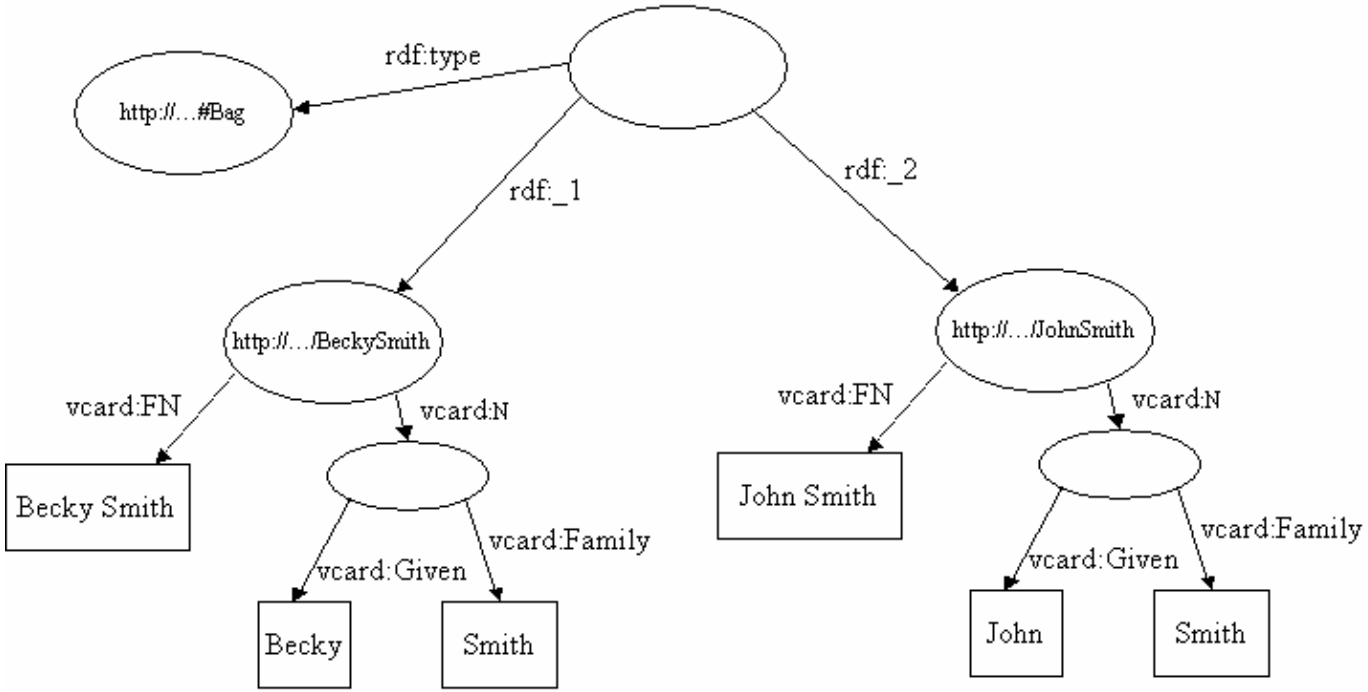
Intersection, différence

- Ces opérations ensemblistes sont possible également avec :
  - model = model1.intersection(model2)
  - model = model1.difference(model2)

Collections

- RDF définit trois manières de regrouper des ressources en collections: des containers
  - un BAG est une collection non ordonnée ;
  - une ALT est une collection non ordonnée destinée à représenter des alternatives ;
  - une SEQ est une collection ordonnée.
- Un container est représenté par une ressource qui aura comme rdf:type rdf:Bag, rdf:Alt ou rdf:Seq. On accède ensuite aux éléments du container par les propriétés rdf:_1, rdf:_2 etc
- Une collection peut être naviguée avec un iterator.

Collections



Littéraux (constantes)

Langue

- Les littéraux RDF ne sont pas de simples chaînes de caractères. Les littéraux peuvent avoir une étiquette de langue pour indiquer la langue du littéral, ou encore avoir un type, comme dans un langage de programmation.
- E.g Le littéral « chat » avec une étiquette de langue anglaise est considéré (non sans raison !) comme différent du littéral « chat » avec une étiquette de langue française.

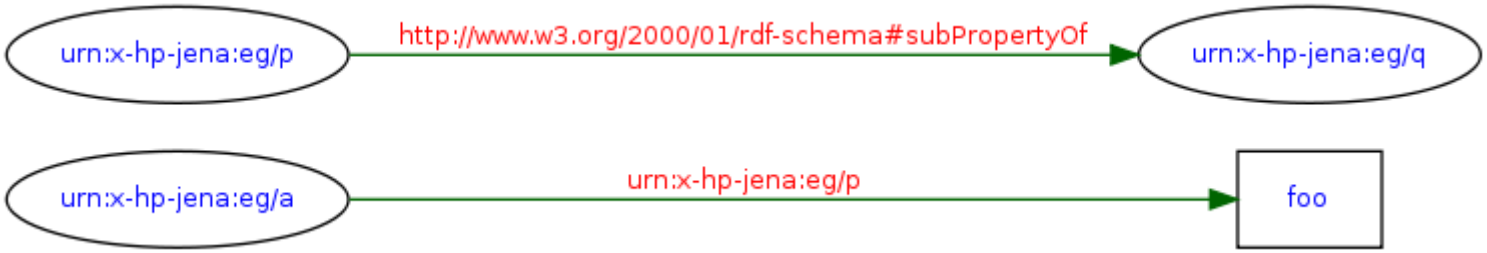
Egalité

- Pour que deux littéraux soient considérés comme égaux, ils doivent être tous les deux soit des littéraux XML soit de simples littéraux, et que leur type soit identique.
- En outre, soit les deux ne doivent avoir aucune étiquette de langue, soit les étiquettes de langues sont présentes mais elles doivent être égales.
  - Pour de simples littéraux, les chaînes de caractères doivent être égales.
  - Les littéraux XML ont deux notions d'égalité. La notion simple est que les conditions précédemment mentionnées soient vraies et les chaînes de caractères soient aussi égales. L'autre notion est qu'ils peuvent être égaux si la canonisation de leur chaîne de caractères est égale.

Raisonnement

RDFS Reasonner

- Uniquement les sous classes et sous propriétés.
- Requête : Trouver toutes les ressources ayant la propriété *q*



Sans raisonnement : PAS DE REPONSE

Avec raisonnement : *a*

Utilisation

- On n’utilise plus simplement la classe Model mais la classe InfModel
- Les méthodes utilisées sont les mêmes, le raisonnement est automatique !

Autre opérations de raisonnement

- A suivre …