

Protection des Méta-Données.

Leo Letouzey

Séminaire du Laboratoire Gepasud, Novembre 2009

11 novembre 2009

Plan

Présentation des Méta-Données

- Définition

- But

- Exemples

- Mise en place

Sécurité des Méta-Données

- Dans quel but ?

- Quelle solution ?

Conclusion

Que sont les méta-Données ?

Il s'agit de données qui décrivent ou définissent une ou plusieurs autres donnée(s).

Il peut s'agir de toutes sortes de précisions (documentation, définition. . .) sur n'importe quel type de donnée.

À quoi peuvent servir les méta-données ?

- Informations complémentaires ;

À quoi peuvent servir les méta-données ?

- ▶ Informations complémentaires ;
- ▶ outils supplémentaires pour faciliter les recherches ;

À quoi peuvent servir les méta-données ?

- ▶ Informations complémentaires ;
- ▶ outils supplémentaires pour faciliter les recherches ;
- ▶ interopérabilités des données.

Informations pouvant apporter des précisions à un morceau de musique :

Informations pouvant apporter des précisions à un morceau de musique :

- ▶ Titre ;
- ▶ numéro de piste ;
- ▶ nom de l'album ;
- ▶ nom de l'artiste ;

Informations pouvant apporter des précisions à un morceau de musique :

- ▶ Titre ;
- ▶ numéro de piste ;
- ▶ nom de l'album ;
- ▶ nom de l'artiste ;
- ▶ genre ;
- ▶ durée ;
- ▶ image de la pochette ;
- ▶ lien vers le site de l'artiste.

Informations pour un livre :

Informations pour un livre :

- ▶ Titre ;
- ▶ auteur ;
- ▶ éditeur ;
- ▶ numéro ISBN ;
- ▶ langue de l'ouvrage ;
- ▶ résumé.

Présentes aussi en programmation.

Présentes aussi en programmation.

En HTML : `<h1>Titre</h1>`

Présentes aussi en programmation.

En HTML : `<h1>Titre</h1>`

Titre est la donnée et `<h1>` est une méta-donnée qui indique la taille de la police.

Présentes aussi en programmation.

En HTML : `<h1>Titre</h1>`

Titre est la donnée et `<h1>` est une méta-donnée qui indique la taille de la police.

→ utilisées pour le Web Sémantic.



Comment mettre en place ces méta-données ?

Comment mettre en place ces méta-données ?
Syntaxe simple et efficace : utilisation de triplets.

Comment mettre en place ces méta-données ?
Syntaxe simple et efficace : utilisation de triplets.

< sujet > < prédicat > < objet >

Comment mettre en place ces méta-données ?
Syntaxe simple et efficace : utilisation de triplets.

< sujet > < prédicat > < objet >

- ▶ <**sujet**> est le document ou l'entité auquel s'attache la méta-donnée.
- ▶ <**prédicat**> est une propriété que possède le sujet.
- ▶ <**objet**> est la valeur de la propriété.

Comment mettre en place ces méta-données ?

Syntaxe simple et efficace : utilisation de triplets.

< sujet > < prédicat > < objet >

- ▶ <**sujet**> est le document ou l'entité auquel s'attache la méta-donnée.
- ▶ <**prédicat**> est une propriété que possède le sujet.
- ▶ <**objet**> est la valeur de la propriété.

Exemple : <doc01> <auteur> <Alice>

ce triplet peut se lire "Alice est l'auteur du document **doc01**".

Quels outils ?

But : Interopérabilité !

Quels outils ?

But : Interopérabilité !

Nécessité d'un vocabulaire commun, facilement accessible et clairement définit.

Une syntaxe principalement orientée pour les communications *machine to machine*.

Quels outils ?

But : Interopérabilité !

Nécessité d'un vocabulaire commun, facilement accessible et clairement définit.

Une syntaxe principalement orientée pour les communications *machine to machine*.

Utilisation d'ontologies.

Ontologie

Ensemble de termes dont la sémantique est clairement établie.

Ontologie

Ensemble de termes dont la sémantique est clairement établie.
Le plus précis possible pour éviter toute ambiguïté.

Ontologie

Ensemble de termes dont la sémantique est clairement établie.
Le plus précis possible pour éviter toute ambiguïté. Forme un vocabulaire se rapportant à un domaine précis. . .

- ▶ Dublin Core
- ▶ FOAF
- ▶ ...

Ontologie

Ensemble de termes dont la sémantique est clairement établie.
Le plus précis possible pour éviter toute ambiguïté. Forme un vocabulaire se rapportant à un domaine précis. . .

- ▶ Dublin Core
- ▶ FOAF
- ▶ ...

ou non :

- ▶ RDFS ;
- ▶ OWL ;

Ontologie

Ensemble de termes dont la sémantique est clairement établie.
Le plus précis possible pour éviter toute ambiguïté. Forme un vocabulaire se rapportant à un domaine précis. . .

- ▶ Dublin Core
- ▶ FOAF
- ▶ ...

ou non :

- ▶ RDFS ;
- ▶ OWL ;

Chacun peut définir sa propre ontologie s'il ne trouve rien de satisfaisant.

Syntaxe

Les ontologies définissent le vocabulaire.

Syntaxe

Les ontologies définissent le vocabulaire.

RDF définit la syntaxe. Ressource Description Framework.

(Recommandation w3c depuis 2004)

RDF

RDF se base sur une représentation de l'information par triplets.

RDF

RDF se base sur une représentation de l'information par triplets.

RDF n'est pas dépendant d'un langage particulier :

exemple :

| Notation N3 | RDF/XML |
|---|---|
| <pre>@prefix foaf : <http :...> . @prefix rdf : <http :...> . _:bnode0 rdf :type foaf :Person . _:bnode0 foaf :name "Robert" _:bnode0 foaf :mail "rob@home.com"</pre> | <pre><rdf :RDF xmlns :foaf="http :..." xmlns :rdf="http :..."> <foaf :Person> <foaf :name>Robert</foaf :name> <foaf :mbox>rob@home.com</foaf :mbox> </foaf :Person> </rdf :RDF></pre> |

Lecture

SPARQL : SPARQL Protocol And RDF Query language
Recommandation w3c depuis mi - 2008.

Lecture

SPARQL : SPARQL Protocol And RDF Query language
Recommandation w3c depuis mi - 2008.
langage de requêtes pour les documents RDF.

Lecture

SPARQL : SPARQL Protocol And RDF Query language

Recommandation w3c depuis mi - 2008.

langage de requêtes pour les documents RDF.

Utilisation de **Patterns** pour sélectionner les éléments à l'intérieur d'un ou plusieurs document(s) RDF.

Lecture

SPARQL : SPARQL Protocol And RDF Query language

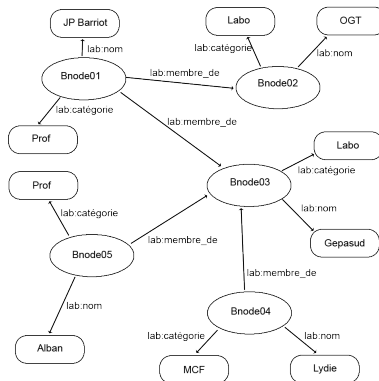
Recommandation w3c depuis mi - 2008.

langage de requêtes pour les documents RDF.

Utilisation de **Patterns** pour sélectionner les éléments à l'intérieur d'un ou plusieurs document(s) RDF.

le pattern `<?s> <préd> <obj>` sélectionnera tout les triplets dont le prédicat vaut `<préd>` et dont l'objet vaut `<obj>`.

Exemples



Exemples

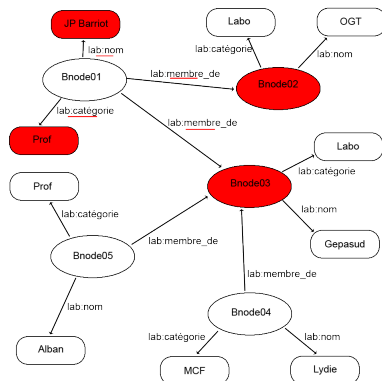
```
SELECT ?p ?o WHERE {  
  ?s lab :nom "JP Barriot" .  
  ?s ?p ?o .  
}
```

Exemples

```
SELECT ?p ?o WHERE {  
  ?s lab :nom "JP Barriot" .  
  ?s ?p ?o .  
}
```

Sélectionne toute les informations sur Jean-Pierre

Exemples



| ?p | ?o |
|----------------|---------------|
| lab :membre_de | _ :Bnode02 |
| lab :membre_de | _ :Bnode03 |
| lab :nom | " JP Barriot" |
| lab :catégorie | " Prof" |

Exemples

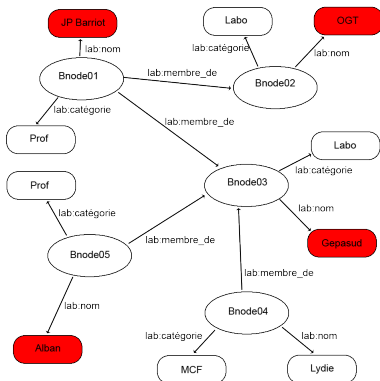
```
SELECT ?nom ?nomLab WHERE {  
  ?s lab :catégorie "Prof" .  
  ?s lab :nom ?nom .  
  ?s lab :membre_de ?labo .  
  ?labo lab :nom ?nomLab .  
}
```

Exemples

```
SELECT ?nom ?nomLab WHERE {  
  ?s lab :catégorie "Prof" .  
  ?s lab :nom ?nom .  
  ?s lab :membre_de ?labo .  
  ?labo lab :nom ?nomLab .  
}
```

Sélectionne le nom et le laboratoire des Professeurs.

Exemples



| ?nom | ?lab |
|--------------|-----------|
| "JP Barriot" | "OGT" |
| "JP Barriot" | "Gepasud" |
| "Alban" | "Gepasud" |

Résumé

- ▶ RDF + Ontologies pour l'écriture des Méta-Données.
- ▶ SPARQL pour la recherche et la lecture.

Plan

Présentation des Méta-Données

Définition

But

Exemples

Mise en place

Sécurité des Méta-Données

Dans quel but ?

Quelle solution ?

Conclusion

Motivations

RDF permet de décrire toutes sortes d'informations.
Certaines sont peut-être personnelles, sensibles et/ou
confidentielles.

Motivations

RDF permet de décrire toutes sortes d'informations.

Certaines sont peut-être personnelles, sensibles et/ou confidentielles.

Exemple de document RDF utilisant l'ontologie FOAF :

- _ :b0 foaf :name "Robert" .
- _ :b0 foaf :phone "31 41 59" . ← information personnelle
- _ :b0 foaf :knows _ :b1 .
- _ :b1 foaf :name "Alice" .
- _ :b1 foaf :phone "27 18 28" . ← information personnelle

Besoins

Il faut une solution :

- ▶ qui gère les contextes ;

Besoins

Il faut une solution :

- ▶ qui gère les contextes ;
- ▶ indépendante du document RDF ;

Besoins

Il faut une solution :

- ▶ qui gère les contextes ;
- ▶ indépendante du document RDF ;
- ▶ qui utilise pleinement la sémantique des documents RDF.

Proposition

Solution proche de ce qui peut se faire avec les bases de données.

Proposition

Solution proche de ce qui peut se faire avec les bases de données.
Utilisation de Vues.

Proposition

Solution proche de ce qui peut se faire avec les bases de données.

Utilisation de Vues.

Une vue = un morceau du document RDF de départ.

Proposition

Solution proche de ce qui peut se faire avec les bases de données.

Utilisation de Vues.

Une vue = un morceau du document RDF de départ.

Les différents utilisateurs se voient ensuite autoriser telles ou telles vues en fonction de leur rôle, du contexte de la requête. . .

Utilisation d'ACL pour définir les autorisations.

Proposition

Comment définir une vue sur un document RDF ?

Proposition

Comment définir une vue sur un document RDF ?

→ avec une requête SPARQL !

Proposition

Comment définir une vue sur un document RDF ?

→ avec une requête SPARQL !

On utilise pour cela la clause **CONSTRUCT** à la place de la clause **SELECT**.

Proposition

Comment définir une vue sur un document RDF ?

→ avec une requête SPARQL !

On utilise pour cela la clause **CONSTRUCT** à la place de la clause **SELECT**.

SPARQL permet de prendre en compte la sémantique lors de la construction des vues.

Exemples

Exemples :

V_1 :

```
CONSTRUCT { ?s med :age ?a ; med :nom ?n ; med :maladie ?m ;  
med :prescription ?p }  
WHERE {  
  ?s rdf :type med :patient ;  
  med :age ?a ;  
  med :nom ?n ;  
  med :maladie ?m ;  
  med :prescription ?p .  
}
```

Exemples

Exemples :

V_1 :

```
CONSTRUCT { ?s med :age ?a ; med :nom ?n ; med :maladie ?m ;
med :prescription ?p }
```

```
WHERE {
```

```
?s rdf :type med :patient ;
```

```
med :age ?a ;
```

```
med :nom ?n ;
```

```
med :maladie ?m ;
```

```
med :prescription ?p .
```

```
}
```

→ GRANT SELECT ON V_1 TO MEDECIN.

Exemples

Exemples :

V_2 :

```
CONSTRUCT { ?s med :nom ?n ; med :prenom ?p ;  
med :adresse ?a ; med :medecin_traitant ?mt }  
WHERE {  
  ?s rdf :type med :patient ;  
  med :nom ?n ;  
  med :prenom ?p ;  
  med :adresse ?a ;  
  med :medecin_traitant ?mt .  
}
```

Exemples

Exemples :

V_2 :

```
CONSTRUCT { ?s med :nom ?n ; med :prenom ?p ;  
med :adresse ?a ; med :medecin_traitant ?mt }
```

```
WHERE {
```

```
?s rdf :type med :patient ;
```

```
med :nom ?n ;
```

```
med :prenom ?p ;
```

```
med :adresse ?a ;
```

```
med :medecin_traitant ?mt .
```

```
}
```

→ GRANT SELECT ON V_2 TO SECRETAIRE.

Exemples

Med_1 est un médecin et exprime une requête sur V_1 .

SELECT ...FROM V_1 WHERE ...

Exemples

Med_1 est un médecin et exprime une requête sur V_1 .

SELECT ...FROM V_1 WHERE ...

Est ce que Med_1 est autorisé à consulter V_1 ? (on vérifie sur l'ACL de V_1)

Exemples

Med_1 est un médecin et exprime une requête sur V_1 .

SELECT ...FROM V_1 WHERE ...

Est ce que Med_1 est autorisé à consulter V_1 ? (on vérifie sur l'ACL de V_1) \leftarrow Oui.

Exemples

Med_1 est un médecin et exprime une requête sur V_1 .

SELECT ...FROM V_1 WHERE ...

Est ce que Med_1 est autorisé à consulter V_1 ? (on vérifie sur l'ACL de V_1) \leftarrow Oui.

La requête est modifiée en :

SELECT ...FROM {CONSTRUCT ...} WHERE ...

Limite

Problème :

SPARQL ne supporte pas les clauses CONSTRUCT à l'intérieur d'une clause FROM.

Limite

Problème :

SPARQL ne supporte pas les clauses CONSTRUCT à l'intérieur d'une clause FROM.

Bonne nouvelle :

C'est prévu pour la prochaine version de SPARQL.

Résumé

- ▶ Modèle View-Based Access Control (V-Bac) ;
- ▶ Utilisation d'Access Control List (ACL) ;
- ▶ Utilise toutes les informations disponibles pour définir les différentes vues ;
- ▶ Filtrage de requête ;
- ▶ Les requêtes ne sont plus évaluées contre le document RDF entier, mais uniquement contre les morceaux autorisés.

Plan

Présentation des Méta-Données

Définition

But

Exemples

Mise en place

Sécurité des Méta-Données

Dans quel but ?

Quelle solution ?

Conclusion

- Solution indépendante des données RDF. Important pour l'évolution de SPARQL qui permettra l'altération des données RDF (ajout, suppression, mise à jour) ;

- ▶ Solution indépendante des données RDF. Important pour l'évolution de SPARQL qui permettra l'altération des données RDF (ajout, suppression, mise à jour) ;
- ▶ Utilisation de SPARQL pour sécuriser SPARQL.

Ensuite ?

- ▶ Finir l'implémentation de cette solution (notamment FROM {CONSTRUCT...});

Ensuite ?

- ▶ Finir l'implémentation de cette solution (notamment FROM {CONSTRUCT...});
- ▶ Suivre l'évolution de SPARQL pour utiliser les prochaines fonctionnalités.

Ensuite ?

- ▶ Finir l'implémentation de cette solution (notamment FROM {CONSTRUCT...});
- ▶ Suivre l'évolution de SPARQL pour utiliser les prochaines fonctionnalités. UPDATE

Ensuite ?

- ▶ Finir l'implémentation de cette solution (notamment FROM {CONSTRUCT...});
- ▶ Suivre l'évolution de SPARQL pour utiliser les prochaines fonctionnalités. UPDATE MINUS

Ensuite ?

- ▶ Finir l'implémentation de cette solution (notamment FROM {CONSTRUCT...});
- ▶ Suivre l'évolution de SPARQL pour utiliser les prochaines fonctionnalités. UPDATE MINUS fonctions.

Merci
Questions ?