Web Sémantique

Semaine 03 : le langage de requête SPARQL

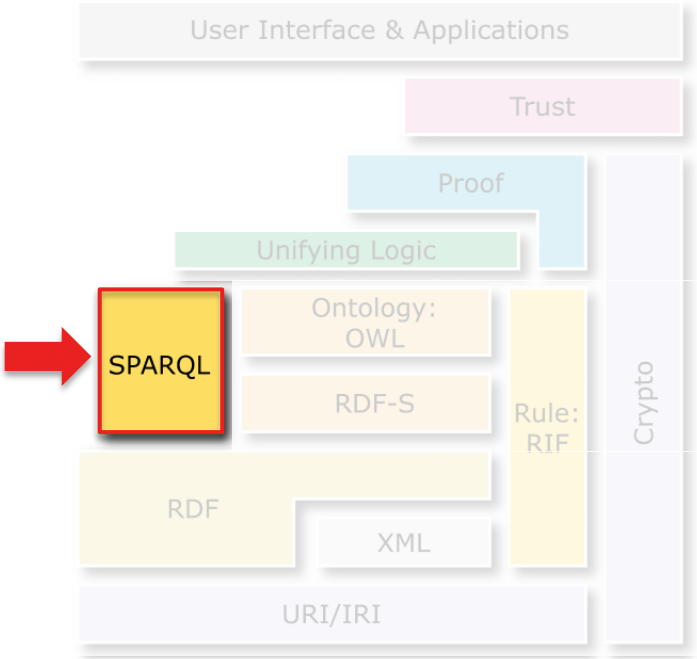
« accéder aux sources de
données du Web »

Olivier Corby

informatiques mathématiques
Inria

SEMAINE 03 : le langage de requête SPARQL

interroger des bases RDF publiées sur le Web



SEMAINE 03 : le langage de requête SPARQL

1. Appariement de graphes RDF
2. Filtres, contraintes et fonctions
3. Pré et post traitements
4. Différentes formes de requêtes
5. Formats et protocole de retours
6. Modification des bases

SEMAINE 03 : le langage de requête SPARQL

1. **Appariement de graphes RDF**
2. Filtres, contraintes et fonctions
3. Pré et post traitements
4. Différentes formes de requêtes
5. Formats et protocole de retours
6. Modification des bases

SPARQL Protocol And RDF Query Language

1. Langage de requête (syntaxe Turtle)
   - SPARQL 1.1 Query Language - W3C REC 21 Mar. 2013

SPARQL Protocol And RDF Query Language

1. Langage de requête (syntaxe Turtle)
   - SPARQL 1.1 Query Language - W3C REC 21 Mar. 2013
   - SPARQL 1.1 Update - W3C REC 21 Mar. 2013

SPARQL Protocol And RDF Query Language

1. Langage de requête (syntaxe Turtle)
   - SPARQL 1.1 Query Language - W3C REC 21 Mar. 2013
   - SPARQL 1.1 Update - W3C REC 21 Mar. 2013
2. Langage de présentation des résultats
   - SPARQL Query Results XML Format - W3C REC 21 Mar. 2013

interroger avec SPARQL

```
SELECT   ce que vous voulez
FROM     où vous voulez
WHERE    { comme vous voulez }
```

les triplets en SPARQL

- syntaxe Turtle avec des points d'interrogation pour les variables:

```
?x    rdf:type    ex:Person
```

les triplets en SPARQL

- syntaxe Turtle avec des points d'interrogation pour les variables:

```
?x   rdf:type   ex:Person
```

- décrire des patrons de graphes à trouver:

```
SELECT ?subject ?property ?value
WHERE { ?subject ?property ?value }
```

les triplets en SPARQL

- syntaxe Turtle avec des points d'interrogation pour les variables:

```
?x   rdf:type    ex:Person
```

- décrire des patrons de graphes à trouver:

```
SELECT ?subject ?property ?value
WHERE { ?subject ?property ?value }
```

- un patron est par défaut une conjonction de triplets:

```
SELECT ?x WHERE
{ ?x     rdf:type    ex:Person .
  ?x     ex:name     ?name . }
```

mêmes abréviations qu'en Turtle

- triplets ayant un sujet commun :

```
SELECT ?name ?fname
WHERE {?x a ex:Person;
      ex:name ?name ;
      ex:firstname ?fname ;
      ex:author ?y . }
```

```
SELECT ?name ?fname
WHERE{?x rdf:type ex:Person.
      ?x ex:name ?name .
      ?x ex:firstname ?fname .
      ?x ex:author ?y . }
```

mêmes abréviations qu'en Turtle

- triplets ayant un sujet commun :

```
SELECT ?name ?fname
WHERE {?x a ex:Person;
      ex:name ?name ;
      ex:firstname ?fname ;
      ex:author ?y . }
```

```
SELECT ?name ?fname
WHERE{?x rdf:type ex:Person.
      ?x ex:name ?name .
      ?x ex:firstname ?fname .
      ?x ex:author ?y . }
```

- plusieurs valeurs

```
?x ex:firstname "Fabien", "Lucien" .
```

mêmes abréviations qu'en Turtle

- triplets ayant un sujet commun :

```
SELECT ?name ?fname
WHERE {?x a ex:Person;
      ex:name ?name ;
      ex:firstname ?fname ;
      ex:author ?y . }
```

```
SELECT ?name ?fname
WHERE{?x rdf:type ex:Person.
      ?x ex:name ?name .
      ?x ex:firstname ?fname .
      ?x ex:author ?y . }
```

- plusieurs valeurs

```
?x ex:firstname "Fabien", "Lucien" .
```

- ressource anonyme

```
[ ex:firstname "Fabien" ] ou
[] ex:firstname "Fabien" .
```

déclarer les préfixes des espaces de noms

- déclarer des préfixes pour les vocabulaires utilisés dans la requête:

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE {
 ?student mit:registeredAt ?x .
}
```

déclarer les préfixes des espaces de noms

- déclarer des préfixes pour les vocabulaires utilisés dans la requête:

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE {
 ?student <http://www.mit.edu#registeredAt> ?x .
}
```

déclarer les préfixes des espaces de noms

- déclarer des préfixes pour les vocabulaires utilisés dans la requête:

```
PREFIX mit:  <http://www.mit.edu#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student
WHERE {
 ?student mit:registeredAt ?x .
 ?x foaf:homepage <http://www.mit.edu> .
}
```

déclarer les préfixes des espaces de noms

- déclarer des préfixes pour les vocabulaires utilisés dans la requête:

```
PREFIX mit:  <http://www.mit.edu#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student
WHERE {
 ?student mit:registeredAt ?x .
 ?x foaf:homepage <http://www.mit.edu> .
}
```

- déclarer un espace de base par défaut i.e. pour les URI relatifs

```
BASE <...>
```

spécifier la langue et le type des littéraux

pour obliger à respecter un attribut : `@fr , ^^xsd:integer`

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x ?f WHERE {
 ?x foaf:name "Fabien"@fr ; foaf:knows ?f .
}
```

spécifier la langue et le type des littéraux

pour obliger à respecter un attribut : `@fr` , `^^xsd:integer`

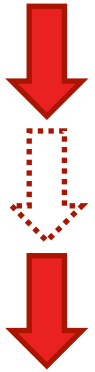
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x ?f WHERE {
 ?x foaf:name "Fabien"@fr ; foaf:knows ?f .
}
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x WHERE {
 ?x foaf:name "Fabien"@fr ;
    foaf:age "21"^^xsd:integer .
}
```

déclarer un motif optionnel

une partie du patron de graphe n'est pas obligatoire.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person foaf:homepage <http://fabien.info> .
 OPTIONAL { ?person foaf:name ?name . }
}
```

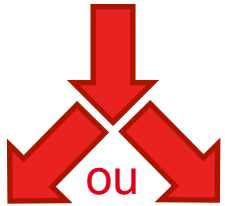


La variable ?name est potentiellement « unbound »

déclarer deux patrons comme alternatifs

faire *l'union* des résultats de deux motifs de graphes

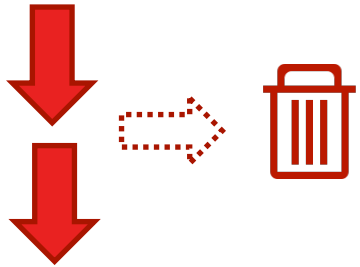
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person foaf:name ?name .
  {
   ?person foaf:homepage <http://fabien.info> .
  }
  UNION
  {
   ?person foaf:homepage <http://bafien.org> .
  }
}
```



soustraire un motif des résultats

supprimer des résultats correspondants à un pattern

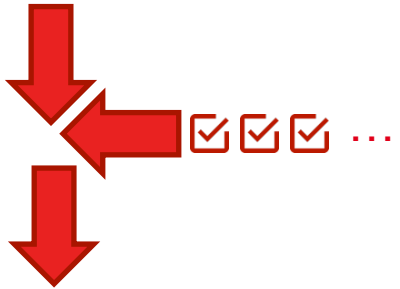
```
PREFIX ex: <http://www.example.abc#>
SELECT ?x
WHERE {
    ?x a ex:Person
    MINUS { ?x a ex:Man }
}
```



spécifier des valeurs prédéfinies pour les variables

fournir des résultats ou une partie du *binding* est prédéfinie

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name
WHERE {
 ?person foaf:name ?name .
}
VALUES ?name  { "Peter" "Pedro" "Pierre" }
```

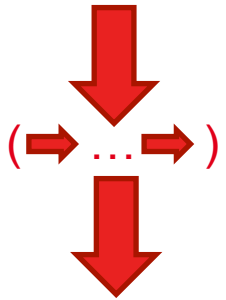


motifs de chemins dans les graphes

des expressions régulières sur des chemins entre ressources

```
/ : sequence          | : alternative
+ : one or several    * : zero or several
? : optional          ^ : reverse
! : negation
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?friend WHERE {
 ?x foaf:name "Fabien Gandon" ;
    foaf:knows+ ?friend .
}
```



ne garder que les résultats distincts

un seul exemplaire des réponses ayant les mêmes valeurs pour les mêmes variables

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name
WHERE { ?person foaf:name ?name . }
```

SEMAINE 03 : le langage de requête SPARQL

1. Appariement de graphes RDF
2. **Filtres, contraintes et fonctions**
3. Pré et post traitements
4. Différentes formes de requêtes
5. Formats et protocole de retours
6. Modification des bases

filtrer les résultats sur les valeurs

déclarer des contraintes supplémentaires notamment sur les variables

- `select` = clause sélectionnant les valeurs à retourner
- `where` = patron de graphe à apparier
- **`filter`** = contraintes ajoutées dans la clause `where` exprimées avec des fonctions de tests XPath 2.0 ou externes

ex. personne ayant au moins 18 ans

```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?person ?name
WHERE {
 ?person    rdf:type ex:Person ;
            ex:name   ?name ;
            ex:age    ?age .
 FILTER (xsd:integer(?age) >= 18)
}
```

test sur les valeurs

tester et comparer constantes et variables

- Comparateurs : `<, >, =, <=, >=, !=`
- Expressions régulières : `regex(?x, "A.*")`
- Tests sur les valeurs des variables : `isURI(?x), isBlank(?x), isLiteral(?x), bound(?x)`

accès et transformation des types et langues

attributs langage et type, re-typage, et valeur brute

- Attributs et valeurs : `lang(), datatype(), str()`
- Fonctions de (re-)typage (*casting XML Schema*): `xsd:integer(?x)`

fonctions chaînes & littéraux

`CONTAINS(lit1,lit2),STRSTARTS(lit1,lit2),STRENDS(lit1,lit2)`	tester l'inclusion d'une chaîne
`STRDT(value, type)`	construire un littéral typé
`STRLANG(value, lang)`	construire un littéral d'une langue
`CONCAT(lit1,…,litn)`	concaténer des littéraux
`SUBSTR(lit, start [,length])`	extraire une sous-chaîne
`ENCODE_FOR_URI(str)`	encoder une chaîne en URI
`UCASE(str), LCASE(str)`	changer la casse
`STRLEN(str)`	obtenir la longueur

fonctions sur les dates

```
YEAR(Date), MONTH(Date), DAY(Date)
HOURS(Date), MINUTES(Date), SECONDS(Date)
TIMEZONE(Date), TZ(Date)
NOW()
```

accesseurs éléments de dates

date du jour

fonctions numériques

`ABS(Val), CEIL(Val), FLOOR(Val), ROUND(Val)`	valeur absolue, arrondis
`isNumeric(Val)`	tester si c'est une valeur numérique
`RAND()`	valeur aléatoire entre 0 et 1

autres fonctions

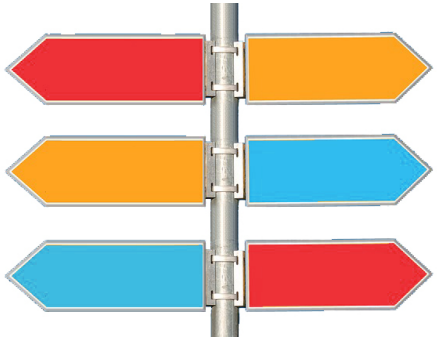
`MD5(val), SHA1(val), SHA256(val), SHA384(val), SHA512(val)`	fonctions de hachage
`coalesce(val`$_1$`,…, val`$_n$`)`	première valeur valide
`IRI(str), URI(str)`	construire un iri/uri d'une chaîne
`BNODE(ID)`	construire un blank node

combinaison et extension

- combinaisons booléennes de tests `&&`, `||`, `!`, `()`
- appel à des fonctions externes / extensions

expression d'embranchement

instruction de test classique: **if… then… else…**



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * where {
 ?x foaf:name ?name ; foaf:age ?age .
 FILTER( if (langMatches(lang(?name), "FR"),
            ?age>=18, ?age>=21) )
}
```

présence ou absence d'une valeur dans une liste

deux instructions vérifiant la présence (`in`) ou l'absence (`not in`)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * where {
  ?x foaf:name ?n .
  FILTER (?n IN ("fabien", "olivier", "catherine") )
}
```

vérifier qu'un motif n'existe pas

l'instruction (not exists) vérifie l'absence d'un motif de graphe

```
SELECT ?name
WHERE {
 ?x foaf:name ?name .
 FILTER NOT EXISTS { ?x foaf:age -1 }
}
```

SEMAINE 03 : le langage de requête SPARQL

1. Appariement de graphes RDF
2. Filtres, contraintes et fonctions
3. **Pré et post traitements**
4. Différentes formes de requêtes
5. Formats et protocole de retours
6. Modification des bases

cibler un ou plusieurs graphes à fouiller

controler les graphes sources utilisés pour répondre

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
FROM <http//www.mit.edu/data.rdf>
WHERE { ?student mit:registeredAt ?x . }
```

cibler un ou plusieurs graphes à fouiller

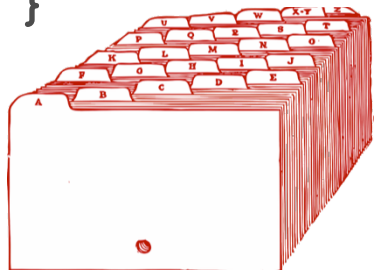
controler les graphes sources utilisés pour répondre

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
FROM NAMED <http//www.mit.edu/data1.rdf>
FROM NAMED <http//www.mit.edu/data2.rdf>
WHERE {
 GRAPH ?g {
  ?student mit:registeredAt ?x .
 }
}
```

trier, filtrer et limiter les réponses

ex. trier par nom les réponses de la n°21 à la n°40

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name . }
ORDER BY ?name
LIMIT 20
OFFSET 20
```



agréger les résultats

grouper sur une ou plusieurs variables: `group by`
agréger des valeurs: `count, sum, min, max, avg, group_concat, sample`
filtrer sur des valeurs groupées: `having`

```
PREFIX mit: <http://www.mit.edu#>
SELECT ?student
WHERE { ?student mit:score ?score . }
GROUP BY ?student
HAVING(AVG(?score) >= 10)
```

requêtes imbriquées

utiliser les résultats d'une requête pour formuler une autre requête

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name WHERE {
  { SELECT (max(?age) as ?max)
    WHERE { ?person foaf:age ?age } }
  ?senior foaf:age ?max .
  ?senior foaf:name ?name
}
```

expressions calculées

définir des variables dont la valeur est calculée

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x (month(?date) as ?month)
WHERE { ?x foaf:birthday ?date . }
```

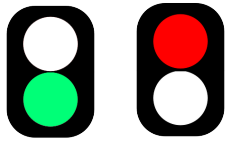
SEMAINE 03 : le langage de requête SPARQL

1. Appariement de graphes RDF
2. Filtres, contraintes et fonctions
3. Pré et post traitements
4. **Différentes formes de requêtes**
5. Formats et protocole de retours
6. Modification des bases

vérifier l’existence d’au moins une réponse

ne pas énumérer tous les résultats mais juste savoir si cela existe (vrai/faux)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?person foaf:age 111 . }
```



construire un graphe RDF en sortie

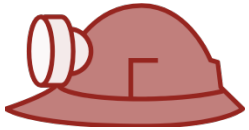
une requête peut avoir comme résultat un nouveau graphe RDF

```
PREFIX mit:  <http://www.mit.edu#>
PREFIX corp: <http://mycorp.com/schema#>

CONSTRUCT { ?student a corp:FuturExecutive . }

WHERE     { ?student a mit:Student . }
```

description libre à la discrétion du serveur

pour découvrir des données dont on ne sait pas grand chose

```
DESCRIBE <http://fabien.info>
```



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
DESCRIBE ?x WHERE { ?x foaf:name "Fabien" }
```

SEMAINE 03 : le langage de requête SPARQL

1. Appariement de graphes RDF
2. Filtres, contraintes et fonctions
3. Pré et post traitements
4. Différentes formes de requêtes
5. **Formats et protocole de retours**
6. Modification des bases

résultats de requêtes SPARQL

valeurs trouvées dans format indiqué

- bindings *:* valeurs sélectionnées, au format XML
- graphe RDF
- JSON (appels AJAX dans applications Web)
- CSV/TSV (ex. export)

exemple de *binding* en XML

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>  <variable name="student"/> </head>
 <results>
  <result>
   <binding name="student">
    <uri>http//www.mit.edu/data.rdf#ndieng</uri>
   </binding>
  </result>
  <result>
   <binding name="student">
    <uri>http//www.mit.edu/data.rdf#jdoe</uri>
   </binding>
  </result>
 </results>
</sparql>
```

protocole SPARQL

- envoyer des requêtes
- recevoir les réponses
- reposer sur l'architecture Web

exemple de *binding* en HTTP

comment envoyer une requête SPARQL à un serveur dans un appel HTTP

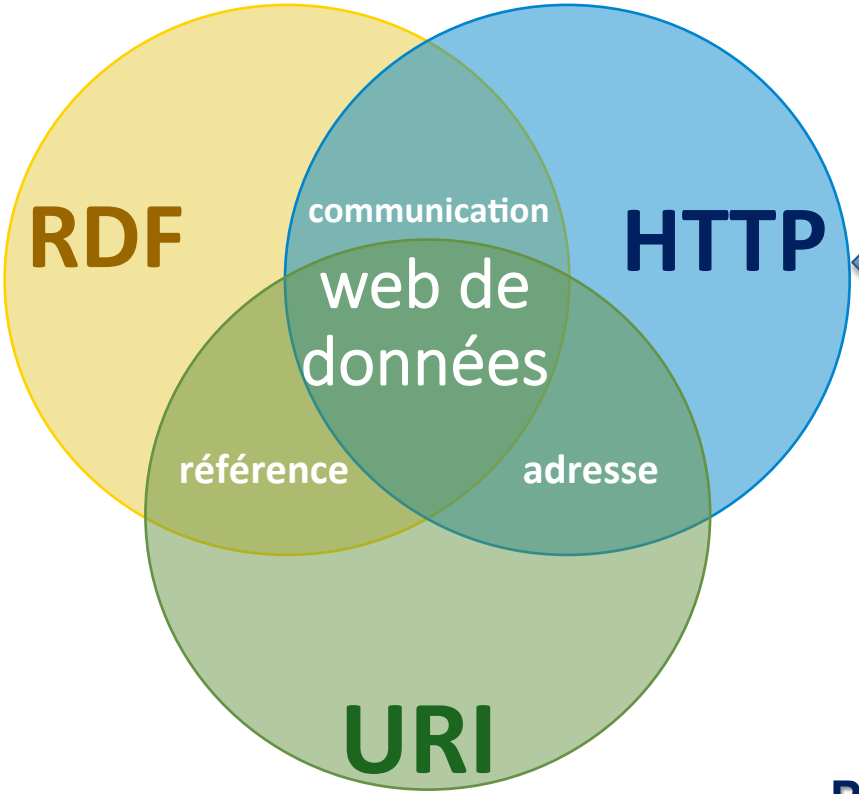
```
GET /sparql?query=<encoded query> HTTP/1.1
Host: fr.dbpedia.org
User-agent: my-sparql-client/0.1
```

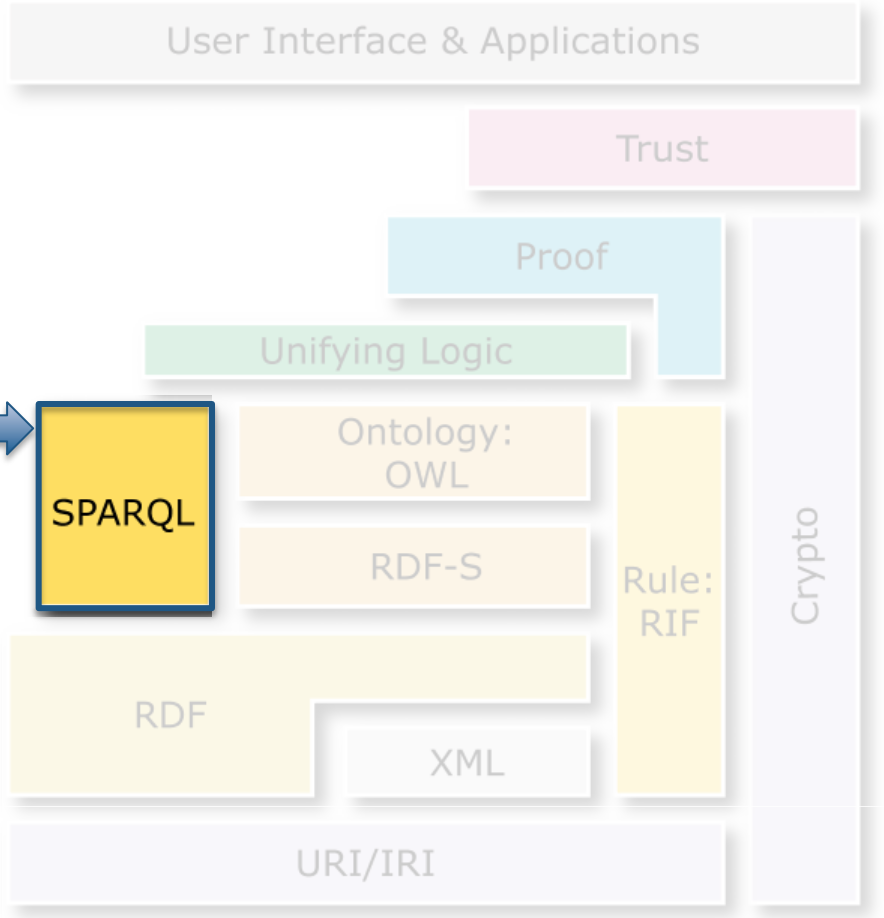
exemple de *binding* en SOAP

comment envoyer une requête SPARQL à un service web dédié

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <soapenv:Body>
  <query-request xmlns="http://www.w3.org/2005/09/sparql-protocol-types/#">
   <query>SELECT ?x ?p ?y WHERE {?x ?p ?y}</query>
  </query-request>
 </soapenv:Body>

</soapenv:Envelope>
```

pile de standardisation





Pile des standards du Web de données W3C®

SEMAINE 03 : le langage de requête SPARQL

1. Appariement de graphes RDF
2. Filtres, contraintes et fonctions
3. Pré et post traitements
4. Différentes formes de requêtes
5. Formats et protocole de retours
6. **Modification des bases**

SPARQL Update

Ajouter ou retirer des triplets ou des graphes

charger des triplets RDF

```
LOAD <http://example.org/dataset>
```

ajouter/retirer des triplets explicités

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex:   <http://example.org/>

INSERT DATA {
   ex:Jimmy foaf:name "Jimmy" ;
      foaf:knows ex:John, ex:Robert .
}
```

ajouter/retirer des triplets explicités

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex:   <http://example.org/>

INSERT DATA {
   ex:Jimmy foaf:name "Jimmy" ;
      foaf:knows ex:John, ex:Robert .
}

;

DELETE DATA {
   ex:Jimmy foaf:knows ex:Mick .
}
```

ajouter/retirer des triplets calculés

```
PREFIX ex: <http://example.org/>
DELETE {
    ?x a ex:Musician
}
WHERE {
    ?x a ex:Musician
}
```

ajouter/retirer des triplets calculés

```
PREFIX ex: <http://example.org/>
INSERT {
    ?x a ex:Artist
}
WHERE {
    ?x a ex:Musician
}
```

ajouter/retirer des triplets calculés

```
PREFIX ex: <http://example.org/>
DELETE {
    ?x a ex:Musician
}
INSERT {
    ?x a ex:Artist
}
WHERE {
    ?x a ex:Musician
}
```

SEMAINE 03 : le langage de requête SPARQL

interroger des bases RDF publiées sur le Web