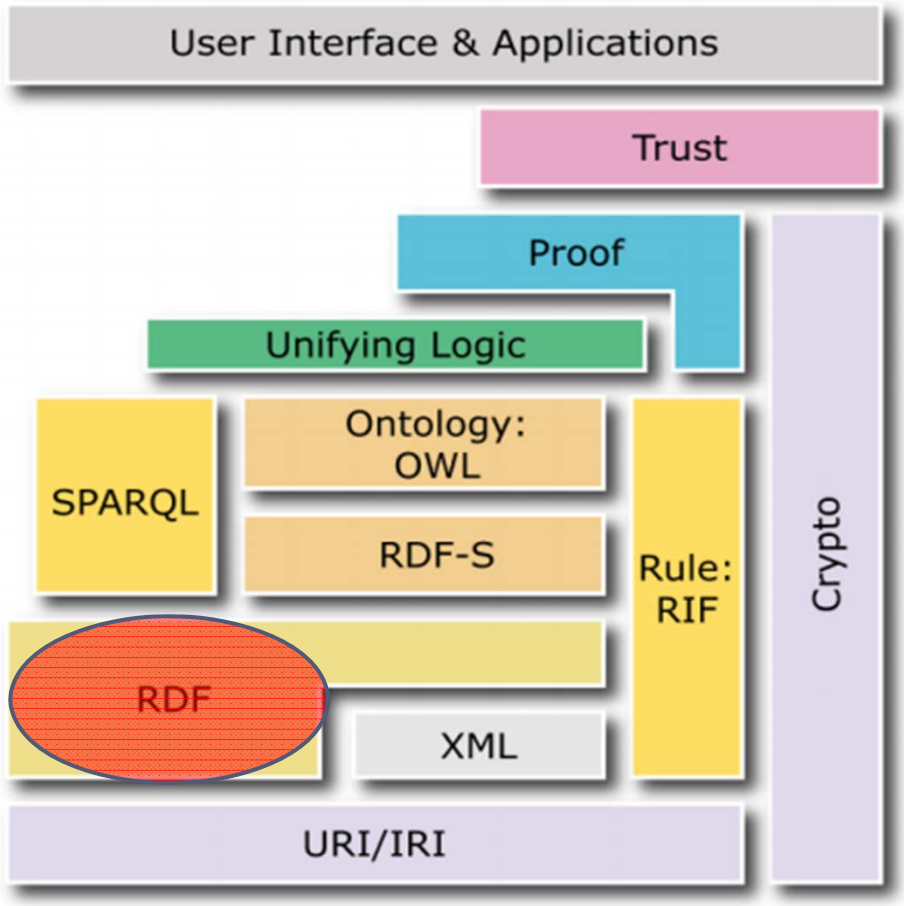
Web Sémantique & XML

Master IPS 2021/2022

Ali OUACHA

Web sémantique

Approche en couches



Plan

Chapitre 2
RDF & Syntaxes RDF

I – Introduction

II – Modèle de Données RDF

III – Sérialisation des graphes RDF

- Syntaxe – RDF/XML
- **Syntaxe – Turtle**

III - Sérialisation des graphes RDF

Syntaxe - TURTLE

Turtle

- Un graphe RDF est composé de triples
- Chaque triplet est composé d'un **sujet**, d'un **prédicat** et d'un **objet**.
- Un document **Turtle** permet d'écrire un graphe RDF sous une forme textuelle compacte.

Triplet Simple

- Un triple est représenté par une **séquence de termes** (sujet, prédicat, objet), **séparés par des espaces** et **terminés** par un **point ' . '**

```
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object1> .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Listes de prédicats

- Dans le cas ou le **même sujet** est référencé par un certain nombre de prédicats, les **couples prédicats-objet** sont séparés par le symbole **';'**

```
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object1> .
<http://one.example/subject1>  <http://one.example/predicate2>  <http://one.example/object2> .
<http://one.example/subject1>  <http://one.example/predicate3>  <http://one.example/object3> .
```

```
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object1> ;
                               <http://one.example/predicate2>  <http://one.example/object2> ;
                               <http://one.example/predicate3>  <http://one.example/object3> .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Listes d'Objets

- Dans le cas ou le même **couple sujet et prédicat** est **partagé** par plusieurs **objets**, la **liste deces objets** sont séparés par le symbole **' , '**

```
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object1> .
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object2> .
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object3> .
```

```
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object1> ,
                                                                <http://one.example/object2> ,
                                                                <http://one.example/object3> .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

- Il existe trois types de termes RDF définis dans les concepts RDF:
  - les **IRI**,
  - les **littéraux,**
  - les **nœuds vides**.

- Turtle propose plusieurs façons d'écrire chacune.

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

IRIs

- Un IRI peut être : **relative** ou **Absolue** ou **Préfixée**.
- Un IRI **relative** et **absolue** est délimitée par **'<'** et **'>'**
- Un IRI peut contenir des **séquences d'échappement numériques**.
  - Il existe trois formes utilisées dans les documents Turtle:
    1) Séquences d'échappement en **code Unicode**
    2) Séquences d'échappement de **chaîne**
    3) Séquences d'échappement de **caractères réservés**

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

IRIs

- Il existe trois formes des séquences d'échappement numériques utilisées dans les documents Turtle:

1) Séquences d'échappement en **code Unicode:**
   - '\u' hex hex hex hex
     Un caractère Unicode compris entre U+0000 et U+FFFF
   - '\U' hex hex hex hex hex hex hex hex
     Un caractère Unicode compris entre U+0000 to U+10FFFF

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

IRIs

- Il existe trois formes des séquences d'échappement numériques utilisées dans les documents Turtle:

2) Séquences d'échappement de **chaîne :**

Séquences d'échappement de chaîne	Code Unicode équivalant
'\t'	U+0009
'\b'	U+0008
'\n'	U+000A
'\r'	U+000D
'\f'	U+000C
'\"'	U+0022
'\''	U+0027
'\\'	U+005C

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

IRIs

- Il existe trois formes des séquences d'échappement numériques utilisées dans les documents Turtle:

3) Séquences d'échappement de **caractères réservés :**
   - Elle est composée du caractère '\' suivi d'un de `~ . - ! $ & ' ( ) * +,; = /? # @ % _`
   - Elle représentent le caractère à droite du ' \ '.

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

IRI relatif

- Les IRI relatives sont résolues par rapport à une **IRI de base**.
- Une nouvelle IRI de base peut être définie à l'aide de la directive '**@base**' ou '**BASE**'.

```
<http://one.example/subject1>  <http://one.example/predicate1>  <http://one.example/object1> .
```

```
@base <http://one.example/> .
<subject1>  <predicate1>  <object1> .
```

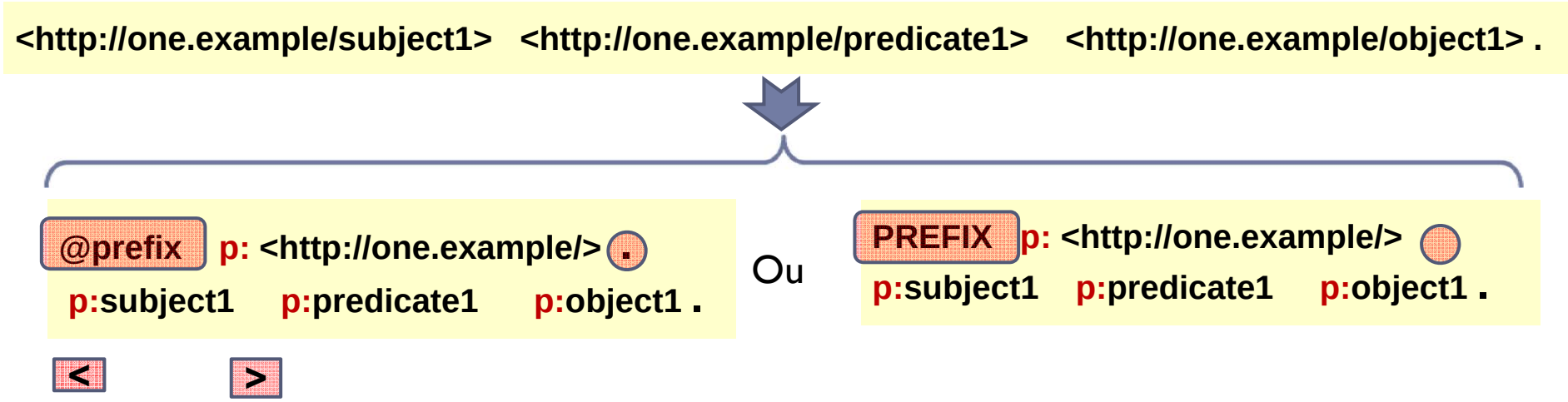
Ou

```
BASE <http://one.example/>
<subject1>  <predicate1>  <object1> .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

IRI préfixé

- Un **nom préfixé** est composé d'une **étiquète (label)** et d'une **partie locale**, séparées par la caractère ' **:** '.
- Un nom préfixé est transformé en un IRI en concaténant l'IRI associé au préfixe et la partie locale.
- La directive '**@prefix**' ou '**PREFIX**' associe une étiquette à une IRI.



III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Remarque

- Le langage Turtle n'autorisait à l'origine que la syntaxe avec le caractère '@' pour écrire les directives de **préfixe** et de **base**.

- Les formes `'PREFIX'` et `'BASE'` ont été ajoutées pour adapter la syntaxe de Turtle sur celle de SPARQL.

- Il est conseillé de sérialiser RDF en utilisant la syntaxe avec le caractère '@': `'@prefix'` et `'@base'`.

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

IRI préfixé : Préfixe Etendu et Préfixe vide

- **Préfixe étendu**

```
@base <http://one.example/> .
<subject1>  <predicate1>  <object1> .
```

→ **<http://one.example/subject1>**

```
@base     <http://one.example/> .
@prefix   p: <path/> .
p:subject2   p:predicate2   p:object2 .
```

→ **<http://one.example/path/subject2>**

- **Préfixe Vide**

```
@prefix   : <http://two.example/>.
:subject3   :predicate3   :object3 .
```

→ **<http://two.example/subject3>**

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

L'IRI : **http://www.w3.org/1999/02/22-rdf-syntax-ns#type**

- Le jeton **'a'** dans la position de prédicat d'un triplet **remplace l'IRI** **http://www.w3.org/1999/02/22-rdf-syntax-ns#type**

```
@prefix  :  <http://one.example/>  .
:subject1   a    :object2 .
```

```
@prefix  :  <http://one.example/>  .
:subject1    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>    :object2 .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

- Les littéraux sont utilisés pour identifier des valeurs telles que des chaînes, des nombres, des dates.

- Les littéraux avec ‘double cote‘ (Quoted Literals) ont une **forme lexicale** délimitée par '**"**' et contient une séquence de caractères autorisés (et parfois des séquences d'échappement numériques).

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://example.org/#green-goblin>      foaf:name      "Green Goblin" .
<http://example.org/#spiderman>         foaf:name      "Spiderman" .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

- La forme lexicale RDF correspondante est constituée des caractères entre les délimiteurs après le traitement des séquences d'échappement.
- Les littéraux ont une forme lexicale **suivie d'une étiquette de langue**, d'un **IRI type de données**, ou **ni l'un ni l'autre**.
- Si elle est présente, l'étiquette de langue est précédée d'un '@'.
- S'il n'y a pas l'étiquette de langue, il peut y avoir un IRI type de données, précédé de '^^'.
- L' IRI type de données peut être écrit en utilisant soit un IRI absolu, un IRI relatif ou un nom préfixé.
- S'il n'y a pas d'IRI de type de données ni de balise de langue, le type de données est `xsd:string`.

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

- Le caractère '\' ne peut apparaître dans aucun littéral sauf dans le cadre d'une séquence d'échappement.

- Autres restrictions dépendantes du délimiteur:
  - Les littéraux délimités par `'`,
    ne peuvent pas contenir les caractères `'`, **`LF(U+000A)`** ou **`CR(U+000D)`**.
  - Les littéraux délimités par `"`,
    ne peuvent pas contenir les caractères`"`, **`LF`** ou **`CR`**.
  - Les littéraux délimités par `'''`
    ne peuvent pas contenir la séquence de caractères `'''`.
  - Les littéraux délimités par `"""`
    ne peuvent pas contenir la séquence de caractères `"""`.

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix show: <http://example.org/vocab/show/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

show:218 rdfs:label "That Seventies Show"^^xsd:string .
 # literal with XML Schema string datatype

show:218 rdfs:label
    "That Seventies Show"^^<http://www.w3.org/2001/XMLSchema#string> .
# same as above

show:218 rdfs:label "That Seventies Show" .
# same again

show:218 show:localName "That Seventies Show"@en .
# literal with a language tag
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix show: <http://example.org/vocab/show/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

…

show:218 show:localName 'Cette Série des Années Soixante-dix'@fr .
# literal delimited by single quote

show:218 show:localName "Cette Série des Années Septante"@fr-be .
# literal with a region subtag

show:218 show:blurb '''This is a multi-line
quotes literal with many quotes (""""")
and up to two sequential apostrophes ('').''' .
# literal with embedded new lines and
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

Numéros

- Les nombres peuvent être écrits avec une forme lexicale et un type de données (par exemple **`"-5.0"^^xsd:decimal`**).

Data Type	Abbreviated	Lexical
xsd:integer	-5	"-5"^^xsd:integer
xsd:decimal	-5.0	"-5.0"^^xsd:decimal
xsd:double	4.2E9	"4.2E9"^^xsd:double

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

Numéros

- Turtle a une syntaxe abrégée pour écrire des valeurs entières, des valeurs décimales (simple précision) et des valeurs à virgule flottante (double précision).

```
@prefix : <http://example.org/elements> .

<http://en.wikipedia.org/wiki/Helium>
        :atomicNumber           2 ;             # xsd:integer
        :atomicMass             4.002602 ;      # xsd:decimal
        :specificGravity        1.663E-4 .      # xsd:double
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Les Littérales

Booléens

- Les valeurs booléennes qui peuvent être écrites sont : «**`true`**» ou «**`false`**» (sensible à la casse)
- Elles représentent des littéraux RDF avec comme type de données **`xsd:boolean`**.

```
@prefix : <http://example.org/stats> .

<http://somecountry.example/census2007>
       :isLandlocked  false .                  # xsd:boolean
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Nœud Vide

- Les nœuds vide enTurtle sont exprimés par les deux caractères `'_:'` suivi par une **étiquette**.

```
@prefix   foaf:   <http://xmlns.com/foaf/0.1/> .

_:alice       foaf:knows   _:bob .
_:bob         foaf:name    "bob".
```

- Un nœud vide peut ne pas avoir d'étiquette,
- Dans ce cas, il peut être représenté par les **crochets** `[]`

```
@prefix     foaf:    <http://xmlns.com/foaf/0.1/> .

# Someone knows someone else, who has the name "Bob".
[]     foaf:knows    [ foaf:name "Bob" ] .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Imbrication de nœuds vides non étiquetés.

- Turtle permet d'imbriquer des listes nœuds vide avec leurs liste de propriétés.
- Dans ce cas, chaque crochet ouvrant ' **[** ' établit un **nouveau nœud vide sujet** qui s'étend jusqu' au crochet fermant ' **]** ' correspondant
- Il sert de **sujet** pour la liste des objets et de prédicat.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

[ foaf:name "Alice" ]      foaf:knows [ foaf:name "Bob" ;
       foaf:knows [
              foaf:name "Eve" ] ;
       foaf:mbox <bob@example.com> ] .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Imbrication de nœuds vides non étiquetés.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

[ foaf:name "Alice" ]      foaf:knows [ foaf:name "Bob" ;
       foaf:knows [
              foaf:name "Eve" ] ;
       foaf:mbox <bob@example.com> ] .
```

- Triplets correspondants:

```
_:a    <http://xmlns.com/foaf/0.1/name>     "Alice" .
_:a    <http://xmlns.com/foaf/0.1/knows>    _:b .
_:b    <http://xmlns.com/foaf/0.1/name>     "Bob" .
_:b    <http://xmlns.com/foaf/0.1/knows>    _:c .
_:c    <http://xmlns.com/foaf/0.1/name>     "Eve" .
_:b    <http://xmlns.com/foaf/0.1/mbox>     <bob@example.com>.
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Collections.

- RDF fournit une structure **Collection** pour les listes de nœuds RDF.
- La syntaxe Turtle pour les **collections** est **une liste** (qui peut être vide) de termes RDF entourée de **( )**.
- La collection représente une structure de liste **`rdf:first`**/**`rdf:rest`**
- La séquence d'objets des instructions **`rdf:first`** sont les termes entourés par **( )** en respectant l'ordre.

- La syntaxe **(...)** **doit** apparaître à la position **sujet** ou **objet** d'un triplet.
- Le nœud vide en tête de liste est le sujet ou l'objet du triple la contenant.

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Collections.

```
@prefix : <http://example.org/foo> .

# the object of this triple is the RDF collection blank node
:subject     :predicate   ( :a :b :c ) .
```

```
@prefix : <http://example.org/foo> .

# an empty collection value - rdf:nil
:subject     :predicate2  () .
```

III - Sérialisation des graphes RDF
Syntaxe - TURTLE

Collections.

```
@prefix : <http://example.org/stuff/1.0/> .

(1 2.0 3E1)   :p     "w" .
```

- Cette exemple est une représentation syntaxique plus simple pour l'exemple en-dessous:

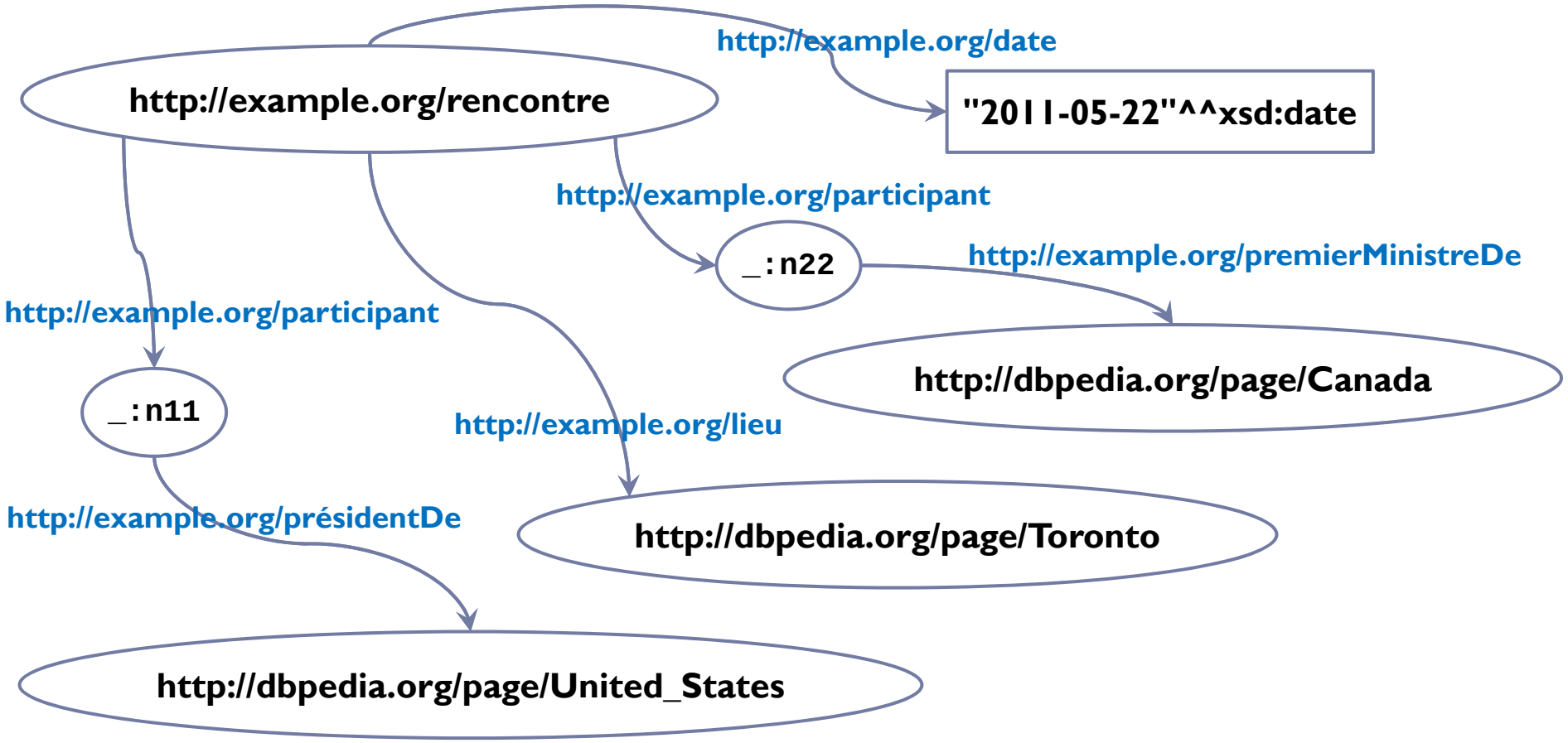
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:b0   rdf:first     1 ;
       rdf:rest      _:b1 .
_:b1   rdf:first     2.0 ;
       rdf:rest      _:b2 .
_:b2   rdf:first     3E1 ;
       rdf:rest      rdf:nil .
_:b0   :p            "w" .
```

https://ci.mines-stetienne.fr/teaching/semweb/turtle.html

Exercice 8

1. Sérialiser le graphe RDF suivant en utilisant Turtle



Exercices

- **Exercice 8**
  1. Sérialiser le graphe RDF suivant en utilisant Turtle

```
# EXERCICE 7

<http://example.org/rencontre>
  <http://example.org/date> "2011-05-22"^^xsd:date.

<http://example.org/rencontre>
  <http://example.org/lieu> <http://dbpedia.org/page/Toronto>.

<http://example.org/rencontre>
  <http://example.org/participant> <_:n11>.

<http://example.org/rencontre>
  <http://example.org/participant> <_:n22>.

<_:n11> <http://example.org/présidentDe>
    <http://dbpedia.org/page/United_States>.

<_:n22> <http://example.org/premierMinistreDe>
    <http://dbpedia.org/page/Canada>.
```

Exercices

- **Exercice 8**
  1. Sérialiser le graphe RDF suivant en utilisant Turtle

```
# EXERCICE 7
@prefix dbp: <http://dbpedia.org/page/> .
@prefix ex: <http://example.org/> .


ex:rencontre    ex:date                 "2011-05-22"^^xsd:date.
ex:rencontre    ex:lieu                 dbp:Toronto.
ex:rencontre    ex:participant          <_:n11>.
ex:rencontre    ex:participant          <_:n22>.
<_:n11>         ex:présidentDe          dbp:United_States.
<_:n22>         ex:premierMinistreDe    dbp:Canada.

```

Exercices

- **Exercice 7**
  1. Sérialiser le graphe RDF suivant en utilisant Turtle

```
# EXERCICE 7
@prefix dbp: <http://dbpedia.org/page/> .
@prefix ex: <http://example.org/> .


ex:rencontre    ex:date                 "2011-05-22"^^xsd:date;
                ex:lieu                 dbp:Toronto;
                ex:participant          <_:n22>,
                                        <_:n11>.
<_:n22>         ex:premierMinistreDe    dbp:Canada.
<_:n11>         ex:présidentDe          dbp:United_States.

```

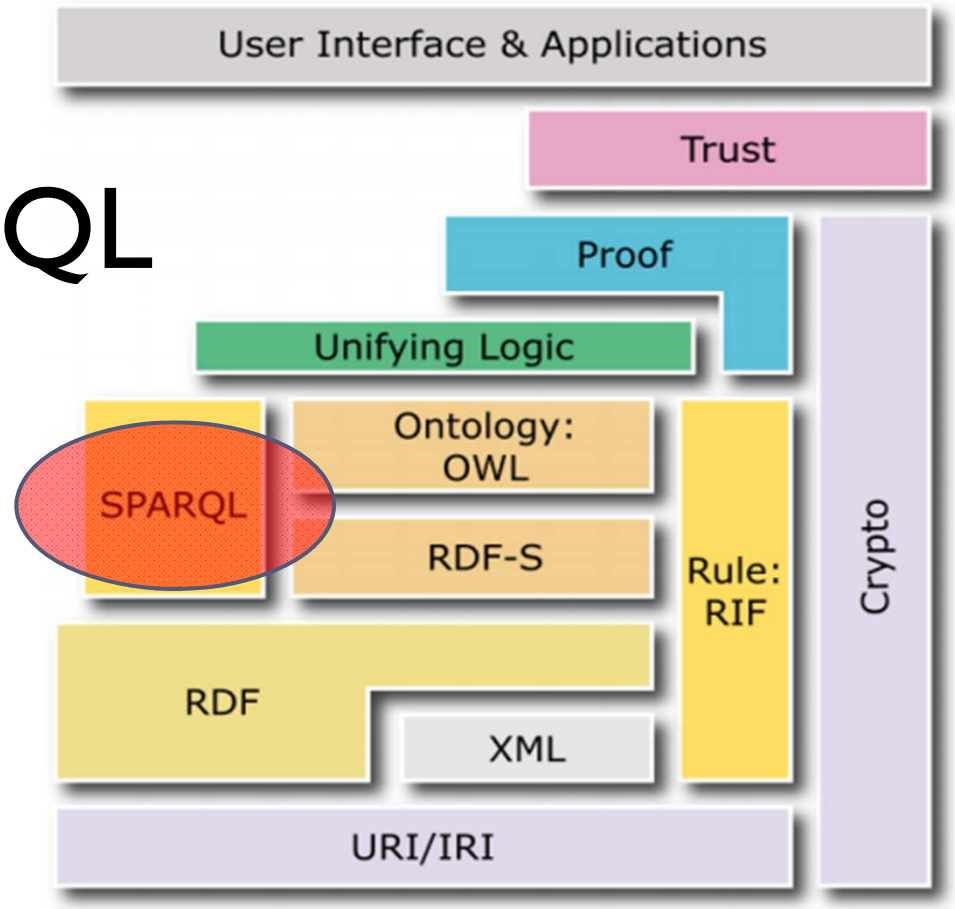
TP

RDF & RDF/Turtle & Jena

Travaille à rendre :

Le code source et fichier générés (Avant le 16-02-2022)

SPARQL


User Interface & Applications
Trust
Proof
Unifying Logic
SPARQL
Ontology: OWL
RDF-S
Rule: RIF
Crypto
RDF
XML
URI/IRI