

STTL

# Transformation Language for Linked Data RDF Graph

[olivier.corby@inria.fr](mailto:olivier.corby@inria.fr)

Université Côte d'Azur

Inria, I3S

Wimmics, Morewais

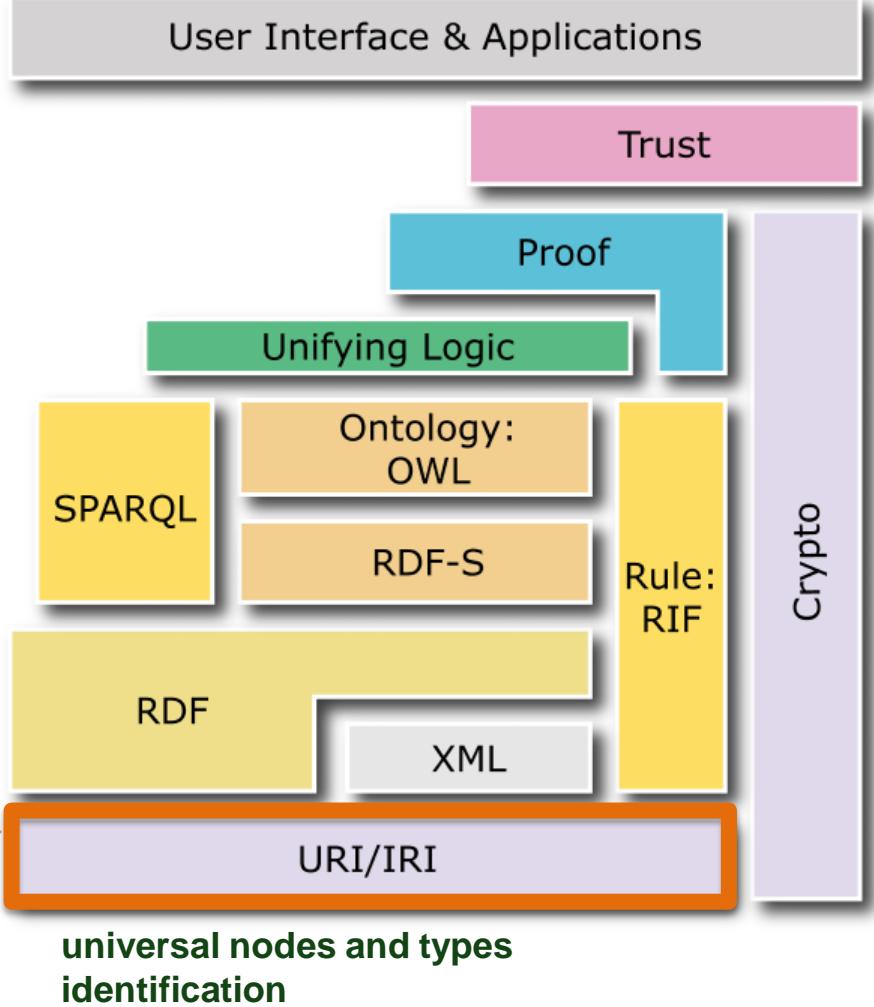
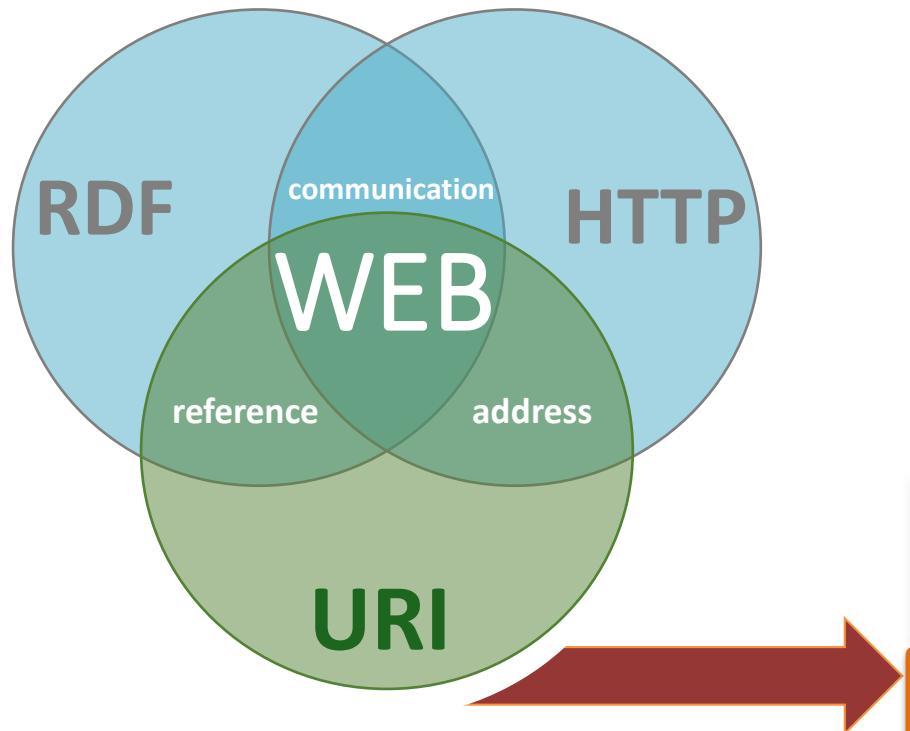
# Agenda

1. Introduction
2. STTL: SPARQL Template Transformation Language
3. STTL Server
4. Conclusion

# W3C Web of Data

1. Semantic Web
2. Web of Data
3. Linked (Open) Data

# Web of Data



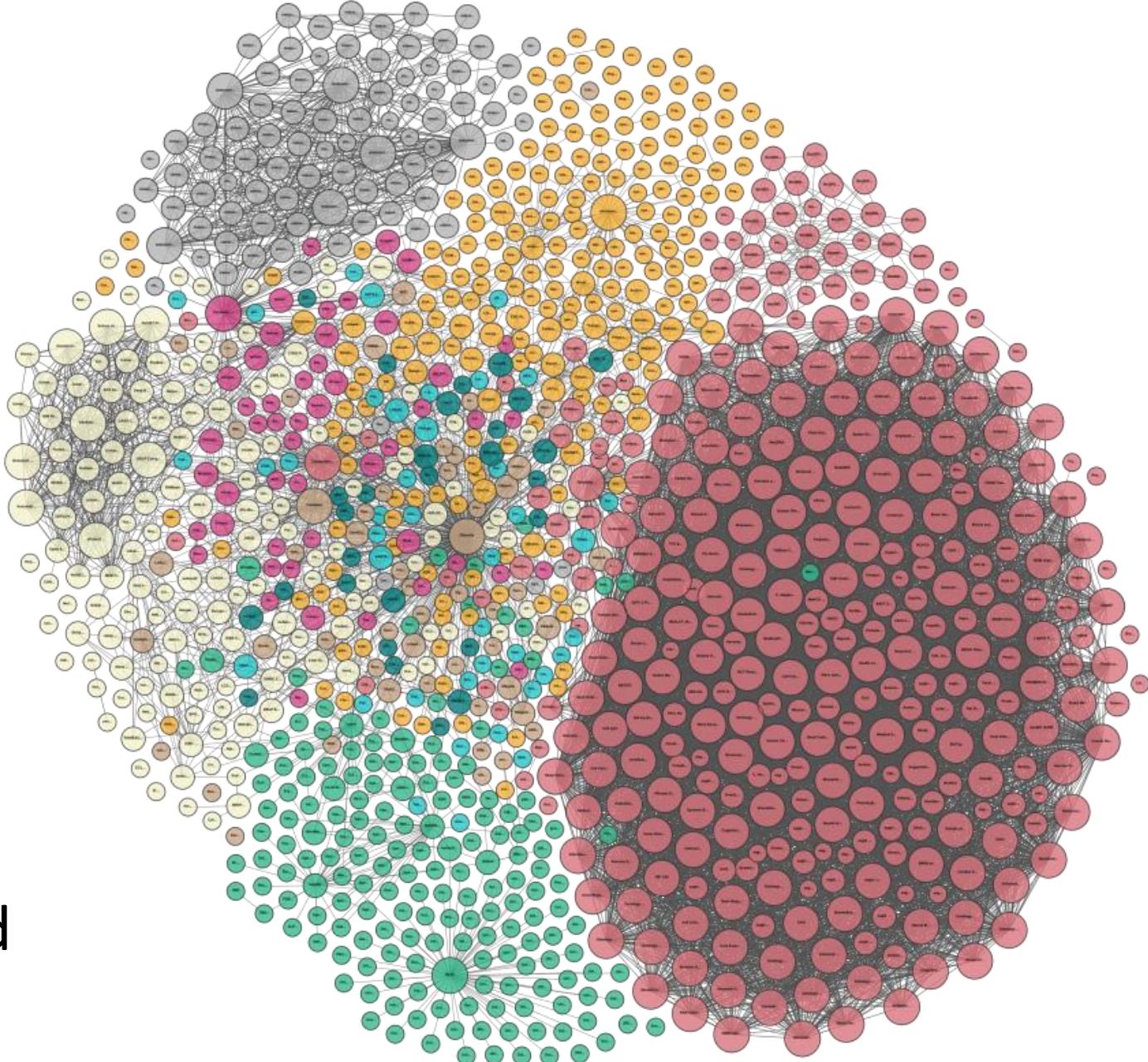
# Web of Data

- Web : Hypertext on top of the Internet (HTML)
- Web of Data : Semantic network on top of the Internet (RDF)

### Legend

Cross Domain
Geography
Government
Life Sciences
Linguistics
Media
Publications
Social Networking
User Generated
Incoming Links
Outgoing Links

# LOD Cloud



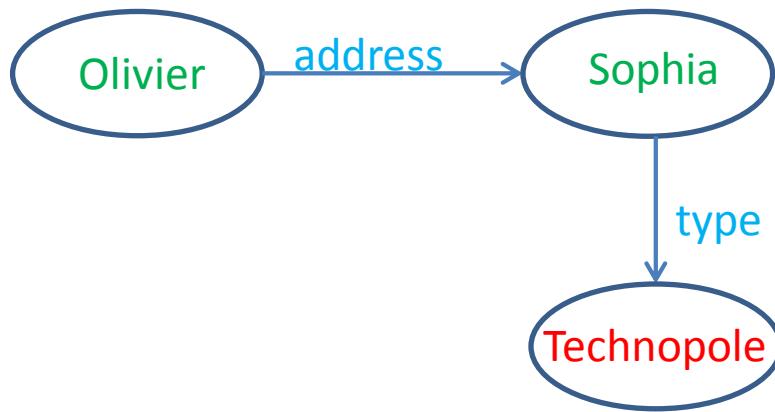
SPARQL Template Transformation Language

6

# W3C Web of Data

1. RDF: Resource Description Framework
2. RDFS: RDF Schema
3. SPARQL: RDF Query Language

# RDF : Labeled Oriented Graph



# RDF Identifier : URI

<<http://www.inria.fr/olivier.corby>>

<<http://xmlns.com/foaf/0.1/name>>

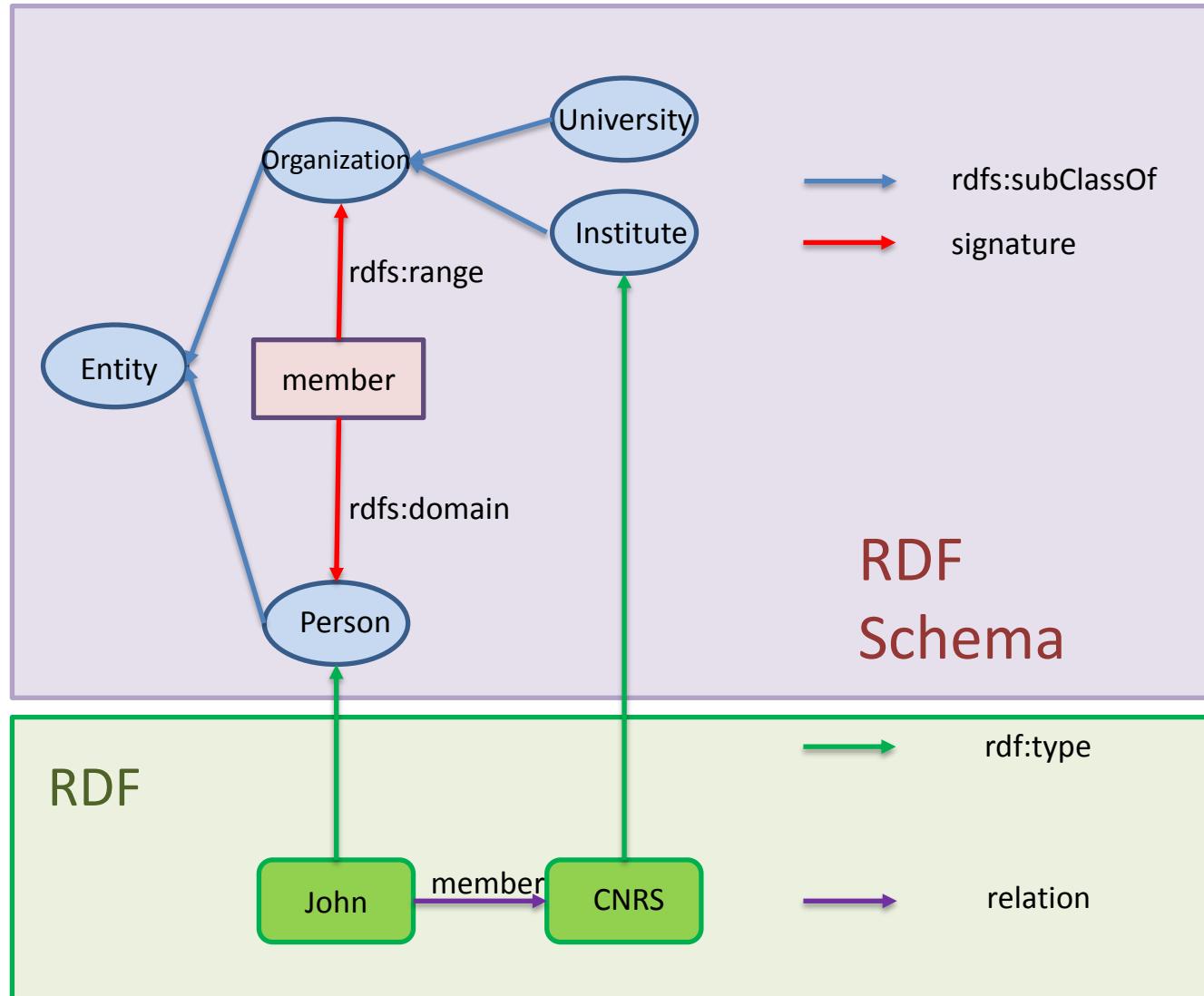
"Olivier Corby" .

<<http://www.inria.fr/olivier.corby>>

<<http://example.org/address>>

<<http://example.org/SophiaAntipolis>> .

# RDF & RDFS

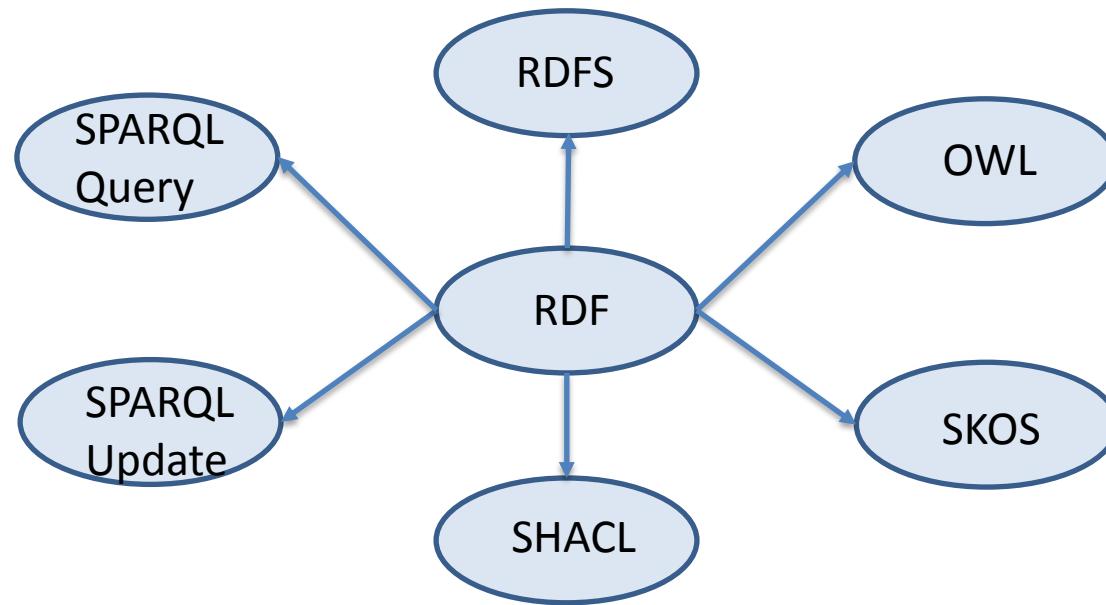


# SPARQL Query

- Linked Data

```
select * where {  
    service <http://dbpedia.org/sparql> {  
        ?x rdfs:label "Paris"@fr ;  
            geo:lat ?lat ; geo:lon ?lon  
    }  
}
```

# Web of Data Ecosystem



# SPARQL Template Transformation Language

# STTL

STTL : transformation language for RDF

*XSLT : transformation language for XML*

- Input RDF graph
- Output Text format
- SPARQL based
- Declarative transformation rules

# STTL motivating use cases

1. Transformation of RDF data from one RDF syntax to another:
  - Turtle
  - RDF/XML
  - JSON LD
2. Presentation of RDF data:
  - RDF to HTML
  - RDF to Latex
  - RDF to Natural Language
  - RDF to graphic format (GML)
3. Transformation of statements in a given language from RDF to another syntax:
  - OWL/RDF to OWL functional syntax
  - SPARQL/RDF (SPIN) to SPARQL syntax
  - AST of  $L$  in RDF to concrete syntax of  $L$
4. Constraint checking
  - OWL Profile: OWL RL
  - SHACL

# XSLT - STTL

```
<xsl:template match="person">  
  <xsl:apply-templates select="knows"/>  
</xsl:template>
```

```
template { st:apply-templates(?y) }  
where { ?in a foaf:Person ; foaf:knows ?y }
```

# XSLT - STTL

```
<xsl:template match="person">  
  <xsl:apply-templates select="knows"/>  
</xsl:template>
```

```
template { st:apply-templates(?y) }  
where { ?in a foaf:Person ; foaf:knows ?y }
```

# XSLT - STTL

```
<xsl:template match="person">  
  <xsl:apply-templates select="knows"/>  
</xsl:template>
```

```
template { st:apply-templates(?y) }  
where { ?in a foaf:Person ; foaf:knows ?y }
```

# XSLT - STTL

	XSLT	STTL
Input	XML	RDF
Output	XML	Text
Syntax	XML	SPARQL extension
Template	xsl:template	template where
Named template	xsl:template name="test"	template ex:test
Apply templates	xsl:apply-templates	st:apply-templates
Apply named template	xsl:call-template	st:call-template
Parameters	xsl:with-param	(?x, ?y)
Numbering	xsl:number	st:number
Sorting	xsl:sort	order by
Grouping	xsl:for-each-group	group by
Condition	xsl:if	if (exp, then, else)

# Differences XSLT - STTL

- XSLT :
  - XML Tree
  - Ordered edges
- STTL :
  - RDF Graph
  - Unordered edges

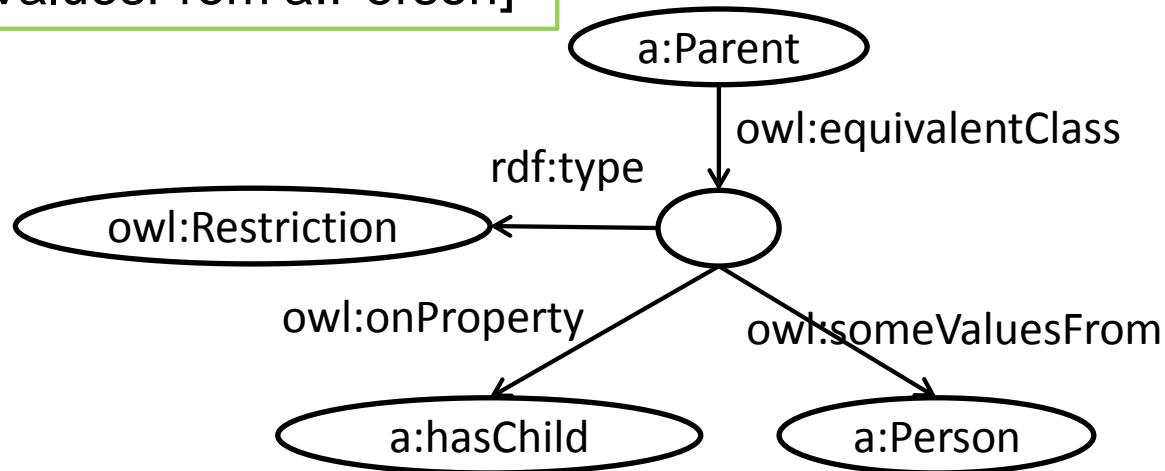
# Example use case: OWL/RDF to OWL/FS

```
a:Parent owl:equivalentClass [ a owl:Restriction ;  
    owl:onProperty a:hasChild;  
    owl:someValuesFrom a:Person]
```

OWL/RDF (Turtle)



**STTL transformation**



```
EquivalentClasses (a:Parent ObjectSomeValuesFrom(a:hasChild, a:Person) )
```

OWL/Functional Syntax

# SPARQL

## Query forms

```
SELECT WHERE { GP }  
CONSTRUCT { GP } WHERE { GP }  
ASK { GP }  
DESCRIBE WHERE { GP }
```

# SPARQL Template

## Query forms

```
SELECT WHERE { GP }  
CONSTRUCT { GP } WHERE { GP }  
ASK { GP }  
DESCRIBE WHERE { GP }
```

```
TEMPLATE { Text Pattern } WHERE { GP }
```

# SPARQL Template

An additional SPARQL query form:

```
TEMPLATE { Text Pattern } WHERE { GP }
```

with Text Pattern = ( VARIABLE | EXP | TEXT )\*

# RDF to HTML transformation

```
TEMPLATE { format {"<a href=\"%s\">%s</a>" str(?x) str(?name) } }
WHERE { ?x a foaf:Person ; foaf:name ?name }
```

```
ns:olivier a foaf:Person ; foaf:name "Olivier".
ns:catherine a foaf:Person ; foaf:name "Catherine".
```

```
<a href='http://ns.inria.fr/olivier'>Olivier</a>
<a href='http://ns.inria.fr/catherine'>Catherine</a>
```

# RDF to Turtle transformation

```
TEMPLATE { ?x " " rdfs:label " " ?name "." }
```

```
WHERE { ?x a foaf:Person ; foaf:name ?name }
```

```
ns:olivier a foaf:Person ; foaf:name "Olivier".
```

```
ns:catherine a foaf:Person ; foaf:name "Catherine".
```

```
ns:olivier rdfs:label "Olivier".
```

```
ns:catherine rdfs:label "Catherine".
```

# STTL: Transformation

A set of templates

```
TEMPLATE { "EquivalentClasses (" ?in " " ?c ")" }  
WHERE { ?in owl:equivalentClass ?c }
```

```
TEMPLATE { "SubClassOf (" ?in " " ?c ")" }  
WHERE { ?in rdfs:subClassOf ?c }
```

```
TEMPLATE { "ObjectSomeValuesFrom (" ?p " " ?c ")" }  
WHERE { ?in a owl:Restriction ;  
        owl:onProperty ?p ;  
        owl:someValuesFrom ?c }
```

# Template recursive call

```
TEMPLATE { "EquivalentClasses ("  
    ?in " " ?c ")" }  
  
WHERE { ?in owl:equivalentClass ?c . }
```

# Template recursive call

```
TEMPLATE { "EquivalentClasses ("  
  st:apply-templates(?in) " " ?c ")" }  
WHERE { ?in owl:equivalentClass ?c . }
```

# Template recursive call

```
TEMPLATE { "EquivalentClasses ("  
    st:apply-templates(?in) " " st:apply-templates(?c) ")" }  
WHERE { ?in owl:equivalentClass ?c . }
```

# STTL

1. SPARQL Template Query form
2. Transformation: a set of Templates
3. Extension functions: `st:apply-templates`, `st:call-template`

# Focus Node ?in

```
template {  
    st:apply-templates(?y)  
}  
where { ?in foaf:knows ?y }
```

# Focus Node ?in

```
template {  
    st:apply-templates(?y)
```

```
}
```

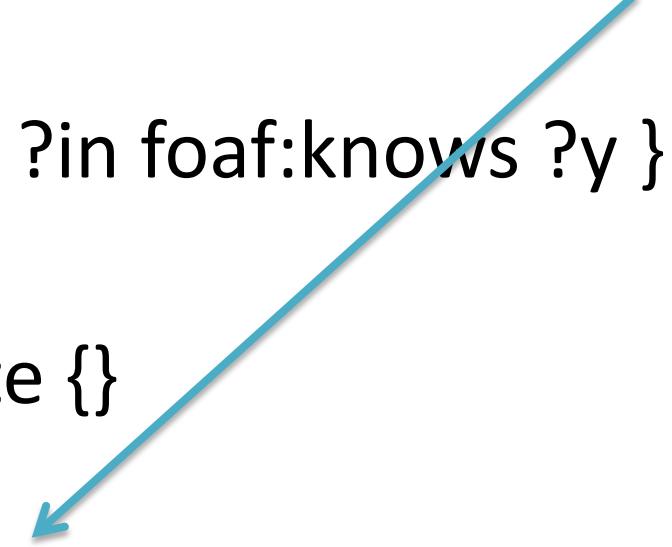
```
where { ?in foaf:knows ?y }
```

```
template {}
```

```
where {
```

?in a foaf:Person

```
}
```



# Named Template

```
template {  
    st:call-template(st:title)  
}  
where {}
```

# Named Template

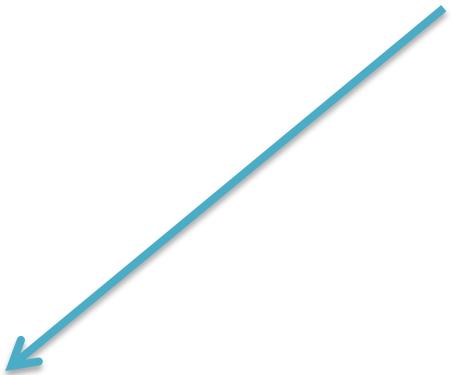
```
template {  
    st:call-template(st:title)
```

```
}
```

```
where {}
```

```
template st:title {}
```

```
where {}
```

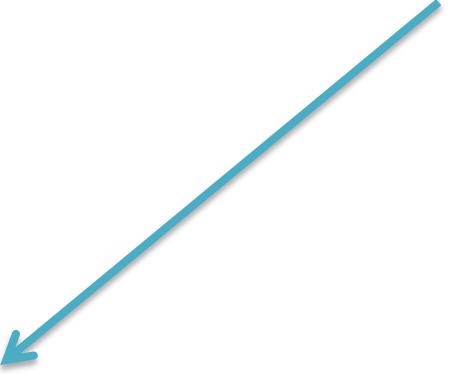


# Named Template

```
template {  
    st:call-template(st:title, ?y)
```

```
}
```

```
where {}
```



```
template st:title (?x) {}
```

```
where {}
```

# STTL Features

# STTL Extension Functions

prefix st: <<http://ns.inria.fr/sparql-template/>>

st:apply-templates(term)

st:apply-templates-with(transform-uri, term)

st:call-template(template-uri, term)

st:call-template-with(transform-uri, template-uri, term)

st:turtle(term)

st:set(term, term)

st:get(term)

# Template Statements

- Separator
- Format
- Group
- Box
- Numbering
- Values Unnest

# Format

```
template {  
  format {  
    "<h2>%1$s</h2><p>%2$s</p>"  
  
    st:apply-templates(?x)  
    st:apply-templates(?y)  
  }  
}  
where {  
}
```

# External Format

```
template {  
  format {  
    <http://example.org/format/test.html>  
  
    st:apply-templates(?x)  
    st:apply-templates(?y)  
  }  
}  
where {  
}
```

# Numbering

```
template {  
    st:number() " " st:apply-templates(?x)  
}  
  
where {  
    ?in foaf:knows ?y  
}  
  
order by ?x
```

# Compiling STTL

```
template { E1 .. En }
where {}
```

compiled as :

```
select (concat(cp(E1), .. cp(En)) as ?out)
where {}
+
aggregate( $\Omega$ , group_concat, ?out)
```

# STTL Compilation

$\text{cp}(\text{Var}(x)) = \text{st:process}(x)$

Default:

$\text{st:process}(\text{?x}) = \text{st:turtle}(\text{?x})$

Overloaded:

```
function st:process(?x) {  
    st:apply-templates(?x)  
}
```

# Constraint Checking with STTL

- OWL Profile checking
  - OWL ontology conforms to OWL RL ?
- SHACL Validation
  - RDF Graph conforms to SHAPE ?

# Constraint Checking with STTL

- Template returns a boolean true/false whether a constraint is verified/not verified
- Aggregate operator is boolean AND

# SHACL Validation

```
template { ?suc }
where {
    graph sh:shape {
        ?sh sh:property [
            sh:path ?p ; sh:class ?c ]
    }
    values ?val { unnest(sh:path(?in, ?p)) }
    bind (exists { ?val rdf:type/rdfs:subClassOf* ?c }
        as ?suc)
}
```

# LDScript: Linked Data Script Language

- Simple language to define extension functions
- For SPARQL and STTL
- On top of SPARQL Filter language

# Function Definition

```
function us:fac(?n) {  
    if (?n = 0, 1, ?n * us:fac(?n - 1))  
}  
}
```

# Function Definition

```
function us:fac(?n) { LDScript  
  if (?n = 0, 1, ?n * us:fac(?n - 1)) SPARQL  
}  
}
```

# LDScript

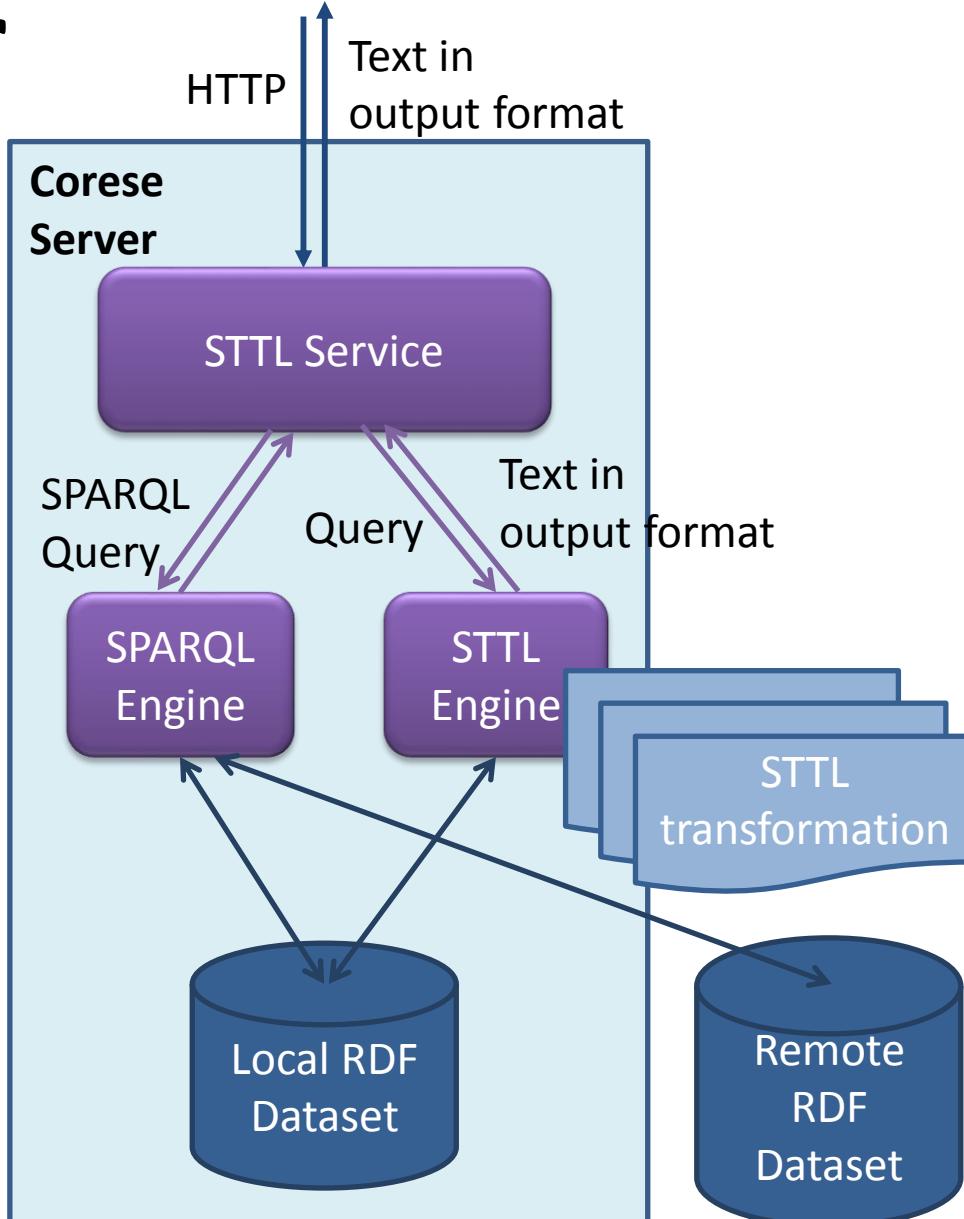
- Function definition & Lambda expression
- Second order functions: apply, funcall, map, reduce
- Pattern matching
- SPARQL Query
- Extension datatypes: dt:graph, dt:triple, dt:list
- Statements: let, for, return

# STTL engine

available in the Corese Semantic Web Factory

- Free download: <http://wimmics.inria.fr/corese>
  - SPARQL engine
  - STTL engine
  - Standalone environment to develop transformation
  - SPARQL endpoint
  - STTL server
- Web Server

# STTL Server



Corese Web Server Demo

localhost:8080

Corese SPARQL Tutorial SPARQL-SPIN Converter OWL Others Sudoku Solver

 CORESE Web Server

Corese is a Semantic Web Factory implementing RDF, RDFS, SPARQL and Inference Rules. This site presents demos of Semantic Web servers and Linked Data Navigators designed with **SPARQL Template Transformation Language**.

## Linked data browsers



## Online services

**SPARQL Query**

Server

```
select * where {  
?x ?p ?y  
}
```

Query

**DBpedia Query**

STD

```
select * where {  
service <http://fr.dbpedia.org/sparql> {  
<http://fr.dbpedia.org/resource/Antibes> ?p ?y  
}  
}  
limit 10  
offset 10
```

Query

**Self Service**

RDF graph URI:

Format: st:turtle

Transform



Copyright © 2015  
Contact: Olivier Corby

Designed by Fuqi Song using Simplex css style



# Auguste



<b>Naissance</b>	-63-09-23+02:00
<b>Décès</b>	14-08-19+02:00
<b>Prédécesseur</b>	Jules César
<b>Successeur</b>	Tibère
<b>Père</b>	Gaius Octavius
<b>Mère</b>	Atia Balba Caesonia
<b>Conjoints</b>	Scribonia (épouse d'Octavien) Clodia Pulchra Livie
<b>Enfants</b>	Julia Caesaris filia
<b>Résumé</b>	Auguste, né sous le nom de Caius Octavius le 23 septembre 63 av. J.-C. à Rome, d'abord appelé Octave puis Octavien, porte le nom de Imperator Caesar Divi Filius Augustus à sa mort le 19 août 14 ap. J.-C. à Nola. Il est le premier empereur romain, du 16 janvier 27 av. J.-C. au 19 août 14 ap. J.-C. Issu d'une ancienne et riche famille de rang équestre appartenant à la gens plébéienne des Octavii, il devient fils adoptif posthume de son grand-oncle maternel Jules César en 44 av.
<b>Wikipedia</b>	<a href="http://fr.wikipedia.org/wiki/Auguste">http://fr.wikipedia.org/wiki/Auguste</a>
<b>DBpedia</b>	<a href="http://fr.dbpedia.org/resource/Auguste">http://fr.dbpedia.org/resource/Auguste</a>

File Edit View History Bookmarks ScrapBook Tools Help

Corese Web Server Demo × +

corese.inria.fr/srv/template?uri=http://fr.dbpedia.org/res ↻ C g Google

**Nord** Colomars Falicon Saint-André-de-la-Roche

**Nord Est** La Trinité (Alpes-Maritimes)

**Est** Villefranche-sur-Mer

**Sud Est**

**Sud**

**Sud Ouest**

**Ouest** Saint-Jeannet (Alpes-Maritimes) La Gaude

**Nord** Gattières

**Ouest**

**Latitude** 43.6959

**Longitude** 7.27141

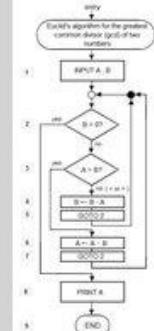
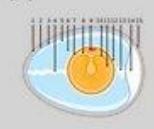
**Wikipedia** <http://fr.wikipedia.org/wiki/Nice>

**DBpedia** <http://fr.dbpedia.org/resource/Nice>



## DBpedia History 01/2002

01/2001 << 12/2001 << 01/2002 >> 02/2002 >> 01/2003

<p><b>1</b> Clotaire Ier (2)</p> 	<p><b>Algorithmique</b> (1)</p> 	<p><b>Carl Sagan</b> (1)</p> 	<p><b>Dagobert Ier</b> (1)</p> 
<p><b>2</b> Espéranto (1)</p> 	<p><b>GNU</b> (1)</p> 	<p><b>Iron Maiden</b> (1)</p> 	<p><b>Linus Torvalds</b> (1)</p> 
<p><b>3</b> Blanc d'œuf (1)</p> 	<p><b>Modèle standard (physique des particules)</b> (1)</p> 	<p><b>Tourisme</b> (1)</p> 	

## 2015 - 2016 - 2017

January

Mo	Tu	We	Th	Fr	Sa	Su
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

February

Mo	Tu	We	Th	Fr	Sa	Su
				1	2	3
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

March

Mo	Tu	We	Th	Fr	Sa	Su
				1	2	3
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

April

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

May

Mo	Tu	We	Th	Fr	Sa	Su
				1		
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

June

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	4
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

July

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

August

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

September

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

October

Mo	Tu	We	Th	Fr	Sa	Su
			1	2		
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

November

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

December

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	



i3S

sophia antipolis

Load: /data/primer.owl

Transform: st.owl

### Result

```

Ontology(<http://example.com/owl/families>
Import(<http://example.org/otherOntologies/families.owl>
SubClassOf(Annotation(rdfs:comment "States that every man is a person.")
<http://example.com/owl/families/Man> <http://example.com/owl/families/Person>
SubClassOf(
  ObjectIntersectionOf(
    ObjectOneOf(<http://example.com/owl/families/Mary> <http://example.com
/owl/families/Bill> <http://example.com/owl/families/Meg>) <http://example.com
/owl/families/Female>
    ObjectIntersectionOf(<http://example.com/owl/families/Parent>
      ObjectMaxCardinality(1 <http://example.com/owl/families/hasChild>
      ObjectAllValuesFrom(<http://example.com/owl/families/hasChild>
<http://example.com/owl/families/Female>))
    )
  DisjointClasses(<http://example.com/owl/families/Woman> <http://example.com
/owl/families/Man>
DisjointClasses(<http://example.com/owl/families/Mother> <http://example.com
/owl/families/Father> <http://example.com/owl/families/YoungChild>
NegativeObjectPropertyAssertion(<http://example.com/owl/families/hasWife>
<http://example.com/owl/families/Bill> <http://example.com/owl/families/Mary>
NegativeDataPropertyAssertion(<http://example.com/owl/families/hasAge>
<http://example.com/owl/families/Jack> "53"^^xsd:integer)
NegativeObjectPropertyAssertion(<http://example.com/owl/families/hasDaughter>
<http://example.com/owl/families/Bill> <http://example.com/owl/families/Susan>
Declaration(Class(<http://example.com/owl/families/Adult>))
EquivalentClasses(<http://example.com/owl/families/Adult> <http://example.org
/otherOntologies/families/Grownup>
Declaration(Class(<http://example.com/owl/families/ChildlessPerson>))
EquivalentClasses(<http://example.com/owl/families/ChildlessPerson>

```

# SPARQL Tutorial

## Select a query

[Previous](#)

13. Count

[Next](#)

### Count

Compter le nombre de solutions avec un opérateur d'agrégation.

(<http://www.w3.org/TR/sparql11-query/#aggregates>)

[Solution](#)[Template](#)[submit](#)

```
prefix h: <http://www.inria.fr/2015/humans#>
select (count(*) as ?c) where {
    ?x ?p ?y
}
```

Corese

SPARQL Tutorial

SPARQL-SPIN Converter

OWL

Others

Sudoku Solver

## SPARQL Sudoku Solver

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	1	4	3	6	5	8	9	7
3	6	5	8	9	7	2	1	4
8	9	7	2	1	4	3	6	5
5	3	1	6	4	2	9	7	8
6	4	2	9	7	8	5	3	1
9	7	8	5	3	1	6	4	2

Generated by Corese server using SPARQL Template Transformation.

2015-06-30T16:18:58

# Conclusion

- STTL Transformation Language for RDF
- Based on SPARQL
- Script Language
- XSLT like
- Hypertext navigation on Linked Data
- [corese.inria.fr](http://corese.inria.fr)