



STTL

Un langage de transformation de graphe RDF

`olivier.corby@inria.fr`

Université Côte d'Azur

Inria, I3S

Wimmics

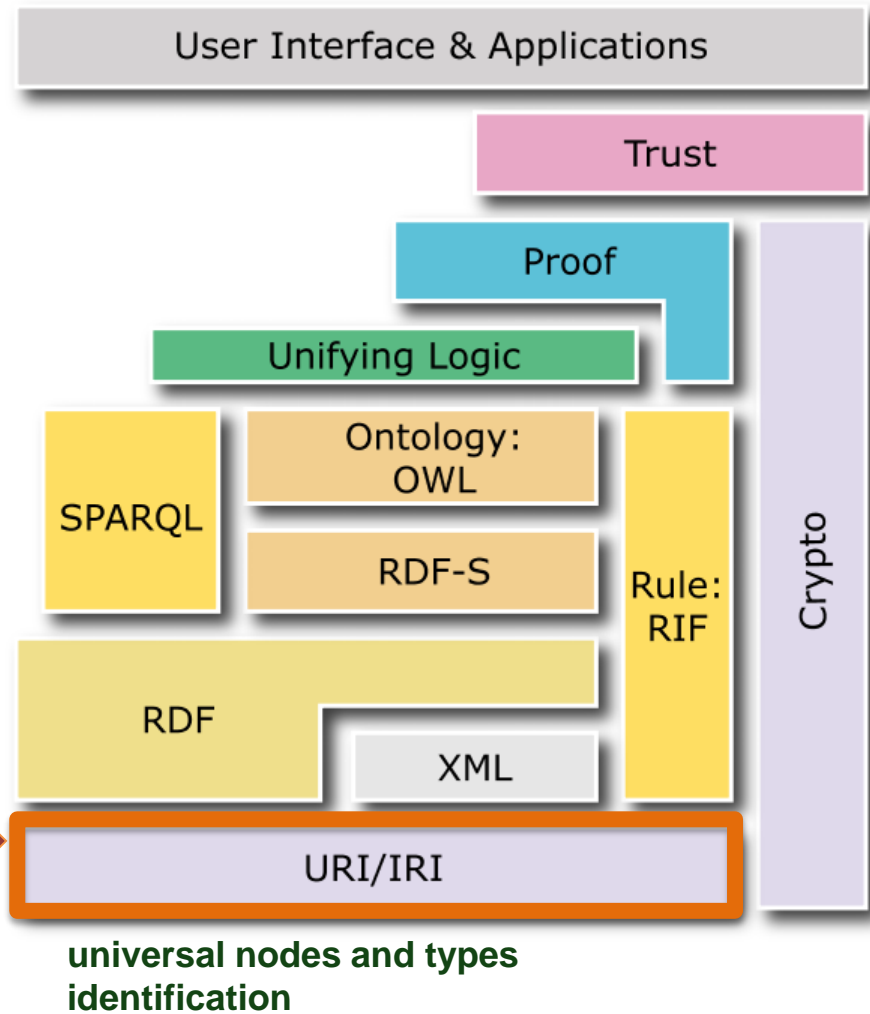
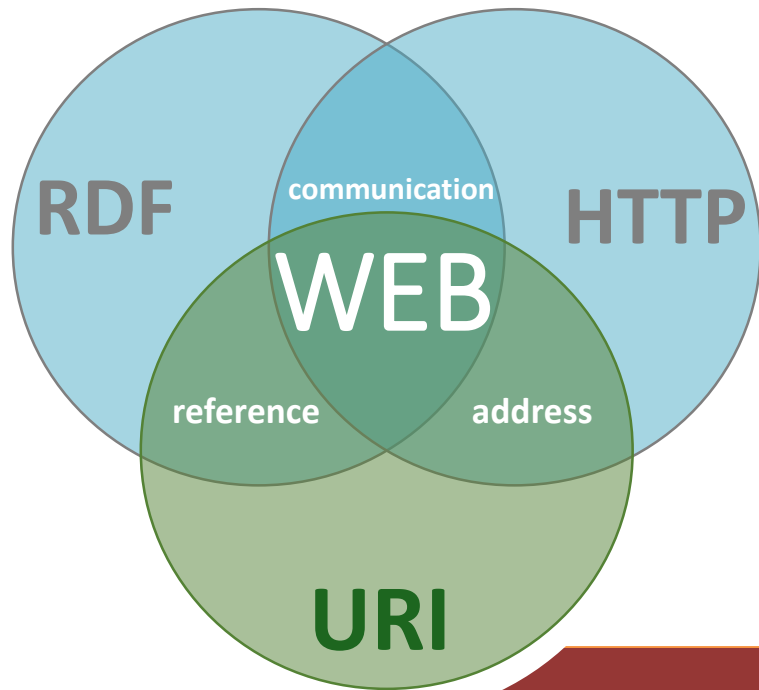
Agenda

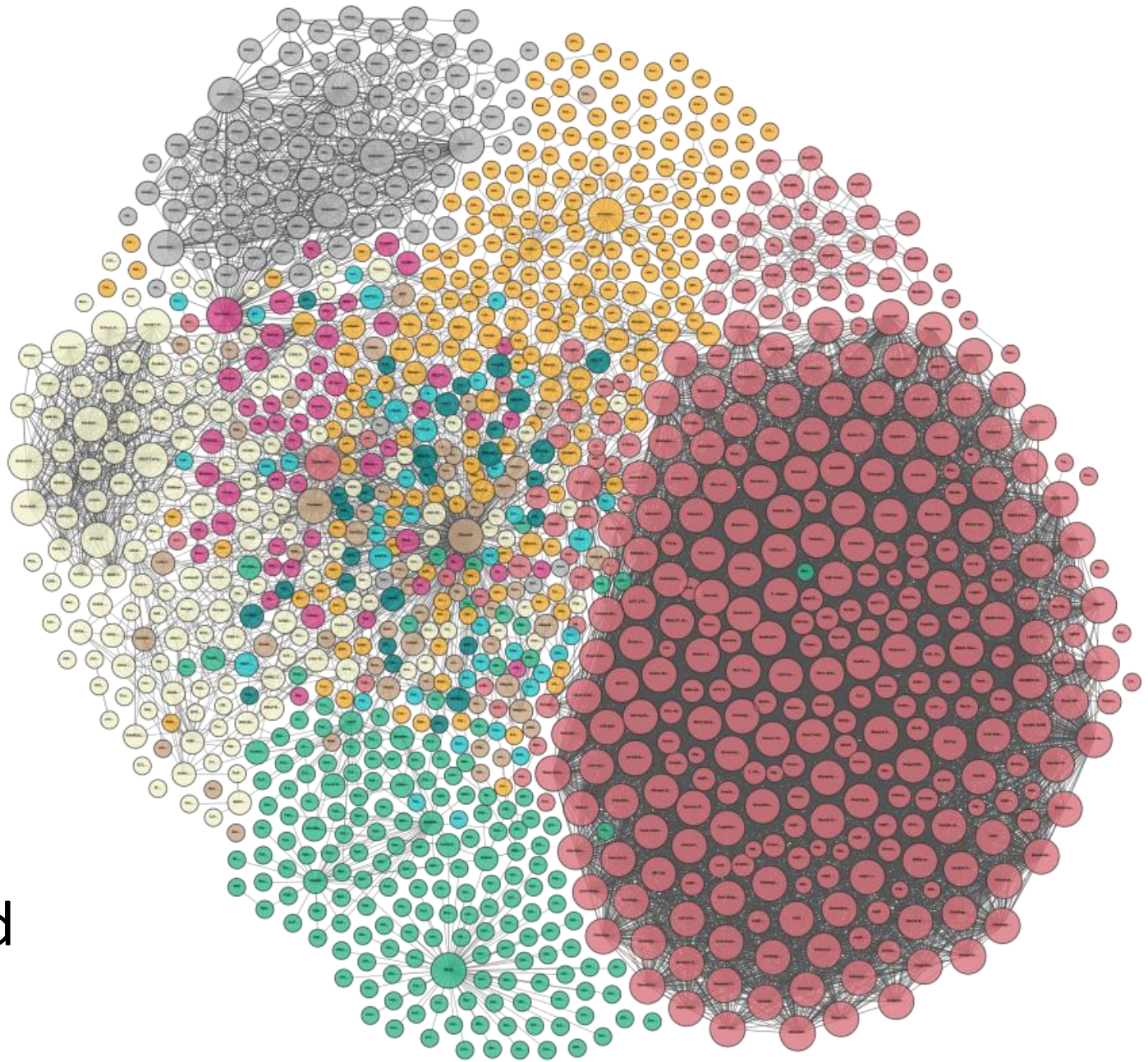
1. Introduction
 2. STTL: SPARQL Template Transformation Language
 3. LDScript: Linked Data Script Language
 4. STTL Server
 5. TD
- <http://wimmics.inria.fr/lectures>

W3C Semantic Web

1. Semantic Web : connaissances
2. Web of Data : données
3. Linked (Open) Data : données liées

Semantic Web





LOD Cloud

SPARQL Template Transformation Language

W3C Web of Data

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

Wimmics Web of Data

1. RDF: Resource Description Framework
2. RDFS: RDF Schema
3. SPARQL: RDF Query Language
4. SPARQL Rule: Inference Rules
5. SPARQL Template: RDF Graph Transformation
6. LDScript : SPARQL based Script Language

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

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)

Différences XSLT - STTL

- XSLT :
 - Arbre XML
 - Les arcs sont ordonnés
- STTL :
 - Graphe RDF
 - Les arcs ne sont pas ordonnés

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

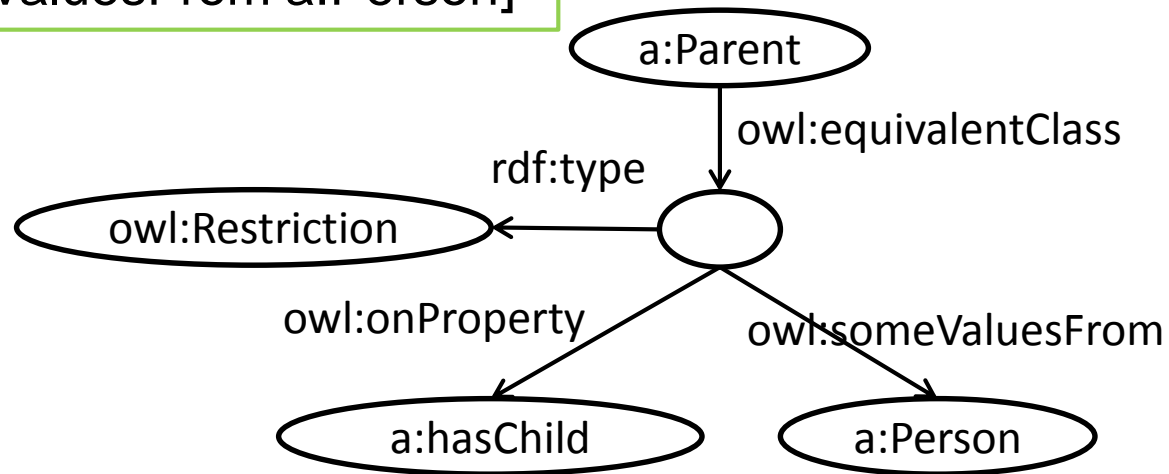
Example use case: OWL/RDF to OWL/FS

```
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

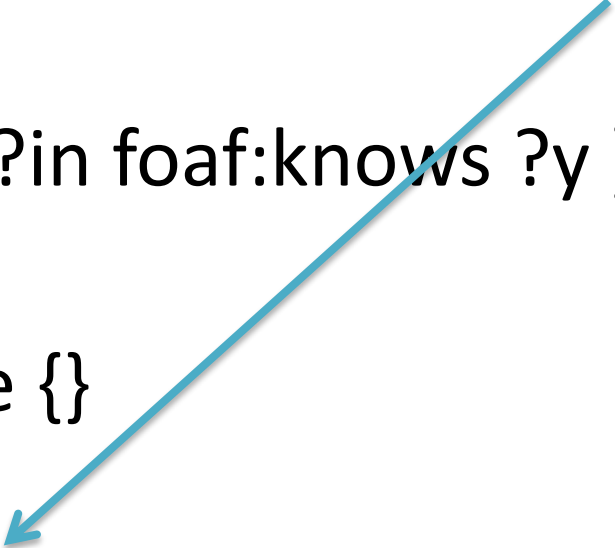
```
template {  
    st:apply-templates(?y)  
}  
where { ?in foaf:knows ?y }
```

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 {}
```

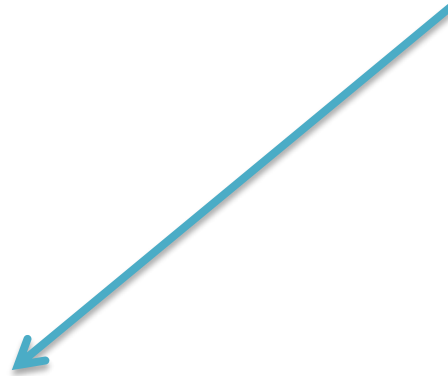
Named Template

```
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)

Priority

```
template { }  
where { }  
pragma { st:template st:priority 200 }
```


Start template

```
template st:start {  
    st:apply-templates(?x)  
}  
where {  
    ?x a foaf:Person  
}
```

Profile template: Declare Functions

```
template st:profile {}  
where {}
```

```
function st:display(?x) {  
  if (isBlank(?x),  
    concat("bnode: " , ?x),  
    st:turtle(?x))  
}
```

Variable Processing

Function `st:process` processes variables

```
template { ?y }  
where { ?in ?p ?y }
```

Compiled into:

```
template { st:process(?y) }  
where { ?in ?p ?y }
```

Overloading Variable Processing

```
function st:process(?x) {  
  if (isBlank(?x),  
    st:apply-templates(?x),  
    st:turtle(?x))  
}
```

Overloading Template Aggregate

Function st:aggregate aggregates template results

```
function st:aggregate(?x) {  
    aggregate(?x, us:merge)  
}
```

```
function us:merge(?list) {  
    apply(rq:and, ?list)  
}
```

Template Statements

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

Separator

```
template {  
    ?y  
    ; separator = ", "  
}  
where {  
    ?in foaf:knows ?y  
}
```

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 {  
}
```

Format Function

`st:format(format, exp+)`

Group

group { E1 .. En }

::=

group_concat(concat(E1, .. En))

Group

```
template {  
    ?in " : " group { ?y }  
}  
where {  
    ?in foaf:knows ?y  
}
```

Group

```
group { E1 .. En ; separator = "--" }
```

Box

box { E1 .. En }

::=

concat(E1, .. En)

st:nl()

box | sbox | ibox

Box

box: nl(+1) exp nl(-1)

sbox: nl(+1) exp indent(-1)

ibox: indent(+1) exp indent(-1)

Numbering

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


Values Unnest

- Extend SPARQL values with expressions

```
template { }
```

```
where {
```

```
    values ?val { unnest (exp) }
```

```
}
```

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)  
    }  
}
```

Compiling STTL

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

compiled as :

```
select (concat(cp(E1), .. cp(En)) as ?out)  
where {}
```

+

```
aggregate( $\Omega$ , group_concat, ?out)
```

STTL Compilation

`cp(Var(x)) = st:process(x)`

Default:

`st:process(?x) = st:turtle(?x)`

Overloaded:

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

LDScript: Linked Data Script Language

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

LDScript

SPARQL

Locable Variable

```
let (?x = us:foo(?y)) {  
    us:bar(?x)  
}
```

List datatype

```
xt:list(1, 2, 3)
```

```
let (?l = @(1 2 3)) {  
}
```

```
xt:iota(5) = xt:list(1, 2, 3, 4, 5)
```

Iterator

```
for (?x in ?list) {  
}
```

```
map(us:fun, ?list)
```

```
maplist(us:fun, ?list)
```

```
mapselect(us:test, ?list)
```

LDScript & SPARQL Query

```
let (?sol = select * where { ?x foaf:knows ?y } ) {
```

LDScript & SPARQL Query

```
let (?sol = select * where { ?x foaf:knows ?y } ) {  
  for ((?x, ?y) in ?sol) {  
    xt:display(?x, ?y)  
  }  
}
```

LDScript & SPARQL Query

```
let (?g = construct where { ?x foaf:knows ?y} ) {  
  
  
  
  
  
  
  
  
  
}
```


LDScript & SPARQL Query

```
let (?g = construct where { ?x foaf:knows ?y} ) {  
    for ((?s, ?p, ?o) in ?g) {  
        xt:display(?s, ?o)  
    }  
}
```

STTL Transformations

1.	RDF to Turtle	st:turtle
2.	RDF to RDF/XML	st:rdfxml
3.	RDF to JSON-LD	st:jsonld
4.	OWL to Functional Syntax	st:owl
5.	SPIN to SPARQL	st:spin
6.	SPARQL Query Result	st:sparql
7.	SPARQL Tutorial	st:web
8.	DBpedia Navigator	st:navlab
9.	Wikipedia Edit History Navigator	st:dbedit
10.	Calendar	st:calendar
11.	History Timeline	
12.	Sudoku (1 template)	
13.	OWL Profile check	st:owlrl
14.	SHACL Validation	st:dsmain

Usage

Create a directory e.g. sttl

Write templates in separate files, with extension .rq

Use:

```
st:apply-templates-with("/home/myself/sttl/")
```

Use in Java:

```
Transformer t = Transformer.create(g, "/home/myself/sttl/");  
String str = t.transform();
```

STTL development environment

Corese/KGRAM 3.1 - Wimmics INRIA I3S - 2015-05-01

File Edit Engine Debug Query Template Explain ?

System Query1 x +

Query Validate to SPIN to SPARQL Prove Trace Search Refresh stylesheet Default stylesheet

```
1 template {
2   st:apply-templates-with(st:owl)
3 }
4 where {
5 }
6
```

Graph XML Validate

Ontology(<http://example.com/owl/families>

Import(<http://example.org/otherOntologies/families.owl>)

SubClassOf(Annotation(rdfs:comment "States that every man is a person."@en)
<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>) <
 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/Your

NegativeObjectPropertyAssertion(<http://example.com/owl/families/hasWife> <http://example.com/owl/families/Bill> <http://example.com/ov

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.co

Declaration(Class(<http://example.com/owl/families/Adult>))

SPARQL Template Transformation Language

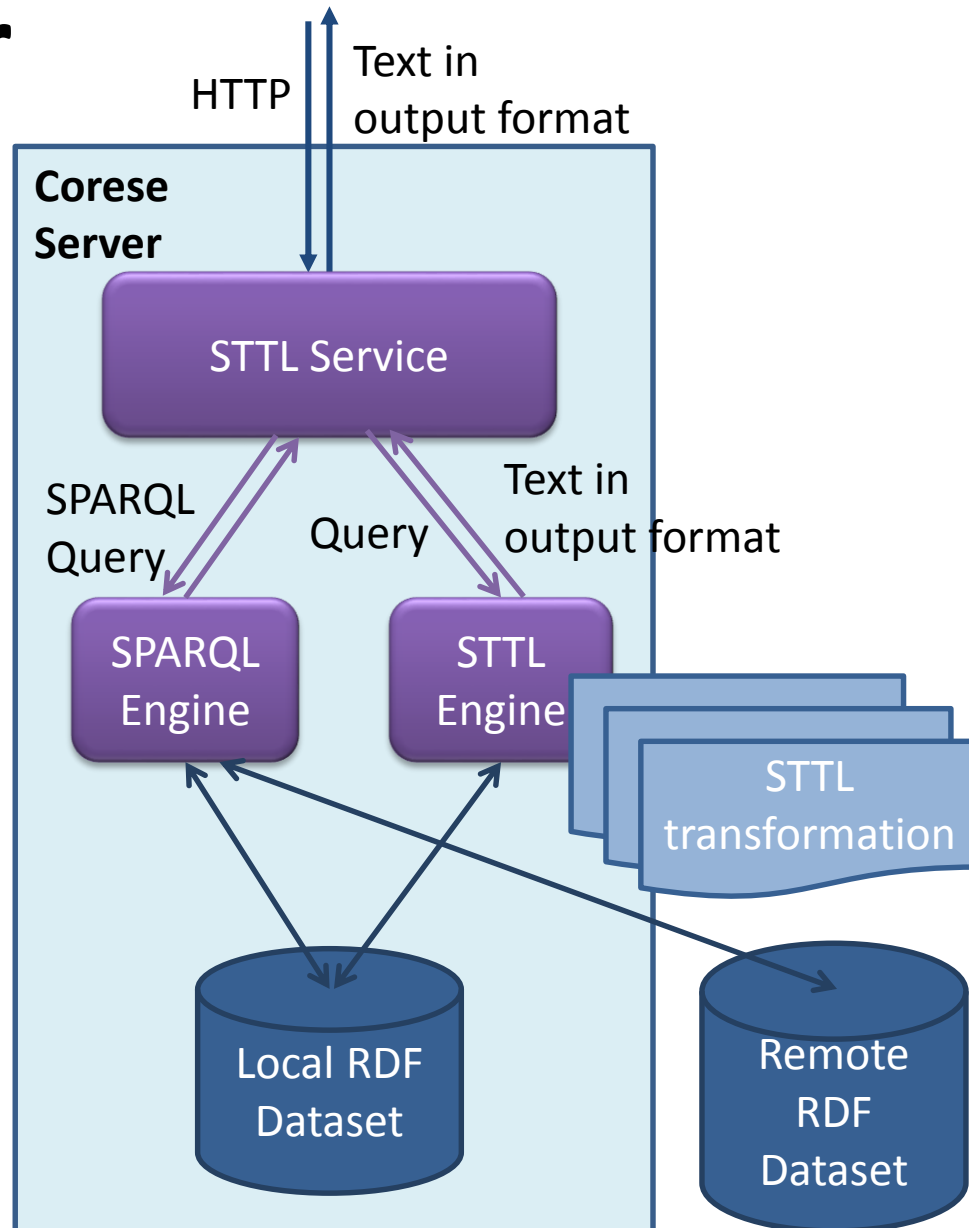
Application

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



Workflow

1. SPARQL Query → Result → Transformation → HTML
2. Service SPARQL Query → Result → Transformation → HTML
3. Transformation(Graph) → HTML
4. Transformation(Graph, URI) → HTML
5. Transformation avec clause service e.g. à DBpedia (long, lat)

Example

`http://corese.inria.fr/srv/template?`
 `uri=http://fr.dbpedia.org/resource/Auguste&`
 `profile=st:dbpedia`

`profile st:dbpedia :`
 `query = construct where {`
 `bind (st:get(st:uri) as ?uri)`
 `service <http://fr.dbpedia.org/sparql> {`
 `?uri rdfs:label ?l ; ... }}`
 `transform = st:navlab`



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



Louis XIV de France
(1638 - 1715)



Auguste (dbpedia fr)



Auguste (dbpedia)



Emmanuel-Philibert de Savoie
(1528-1580)



Places
(Nice)



History
(XIVe Siècle)

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

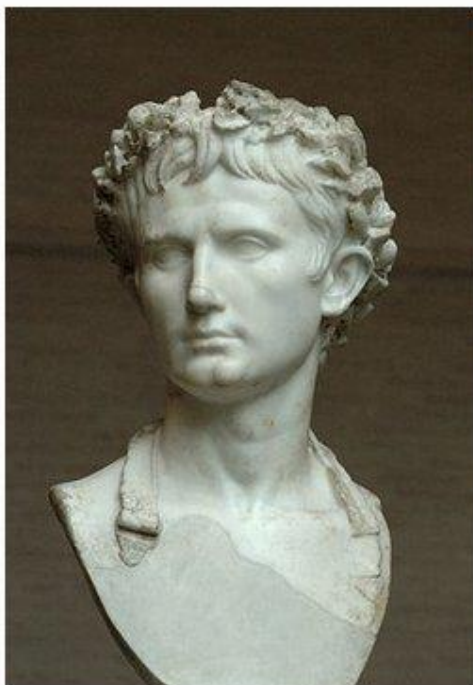
Self Service

RDF graph URI:

Format:

Transform

Auguste



Naissance -63-09-23+02:00

Décès 14-08-19+02:00

Prédécesseur Jules César

Successeur Tibère

Père Gaius Octavius

Mère Atia Balba Caesonia

Conjoints Scribonia (épouse d'Octavien) Clodia Pulchra Livie

Enfants Julia Caesaris filia

Résumé

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.

Wikipedia <http://fr.wikipedia.org/wiki/Auguste>

DBpedia <http://fr.dbpedia.org/resource/Auguste>

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

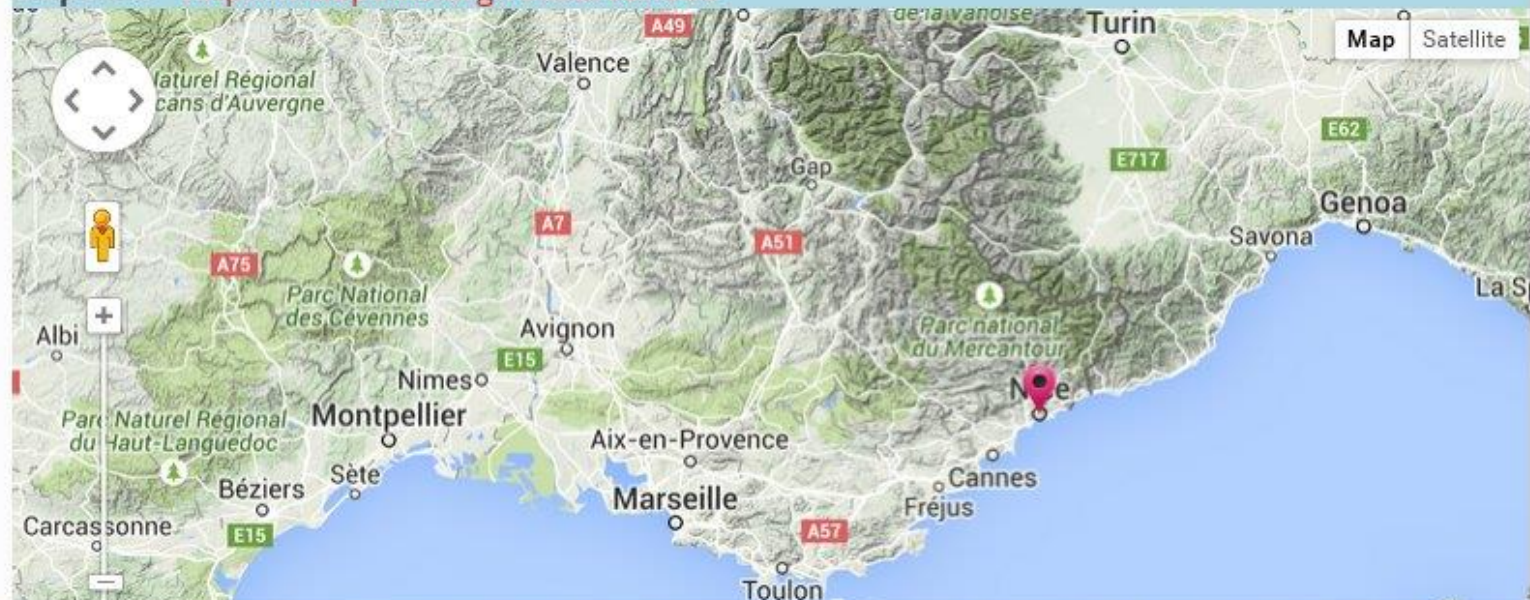
Nord Ouest Gattières

Latitude 43.6959

Longitude 7.27141

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

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



Corese

SPARQL Tutorial

SPARQL-SPIN Converter

OWL

SPARQL


Misc

DBpedia History 01/2002


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

1

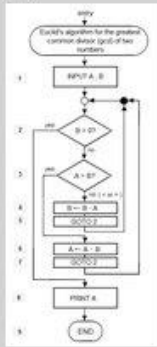
Clotaire Ier (2)




John McCarthy (2)




Algorithmique (1)



Carl Sagan (1)




Dagobert Ier (1)




2


Esperanto (1)




GNU (1)




Iron Maiden (1)



Lisp (1)




Linus Torvalds (1)




3


Blanc d'œuf (1)



Modèle standard (physique des particules) (1)



Tourisme (1)



Corese Web Server

corese.inria.fr/srv/template?transform=st:calendar

Corese
SPARQL Tutorial
SPARQL-SPIN Converter
OWL
SPARQL
Misc

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	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						

March

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			

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	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			

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	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				

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	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				

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	

Calendar generated by a STTL transformation 2016-02-23T14:24:13



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'aggrégation.

(<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
}
```


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

Submit

Reset

Generated by Corese server using SPARQL Template Transformation.

2015-06-30T16:18:58

Conclusion

- STTL Transformation Language for RDF
- Based on SPARQL
- XSLT like

TP

Navigateur hypertexte HTML pour le Web de données