STTL
SPARQL Template Transformation Language for RDF Graphs

Catherine Faron Zucker
Olivier Corby

olivier.corby@inria.fr
faron@i3s.unice.fr

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)

Differences XSLT - STTL

- XSLT :
  - XML Tree
  - Edges are ordered
- STTL :
  - RDF Graph
  - Edges are not ordered

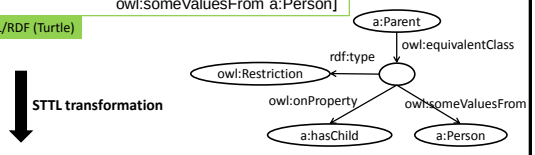
STTL motivating use cases

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

1. SELECT WHERE { GP }
2. CONSTRUCT { GP } WHERE { GP }
3. ASK { GP }
4. DESCRIBE WHERE { GP }

SPARQL Template

Query forms

1. SELECT WHERE { GP }
2. CONSTRUCT { GP } WHERE { GP }
3. ASK { GP }
4. DESCRIBE WHERE { GP }

5. 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 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".
```


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>
```

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. **Template**: SPARQL Template Query form
2. **Transformation**: set of templates
3. **Extension function**: st:apply-templates, st:call-template

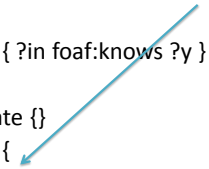
Focus Node

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

Focus Node

```
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)
```

Priority

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


Start template

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


Extension Functions

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


Profile: Define Extension Functions

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

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


Variable Processing

```
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))
}
```

Template Statements

- Separator
- Format
- Group
- Box
- Numbering

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
}
```


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
```


Compiling STTL

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

compiled as :

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


Compiling STTL

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

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

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

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

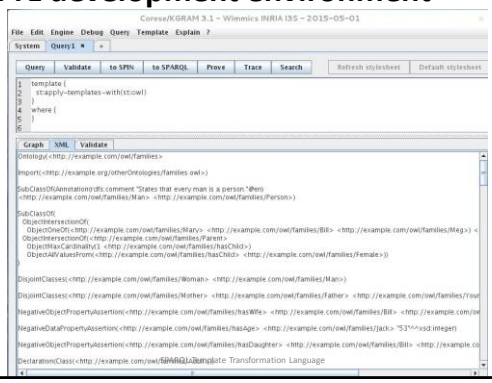
Usage

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



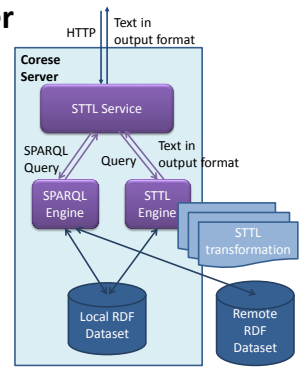
Application

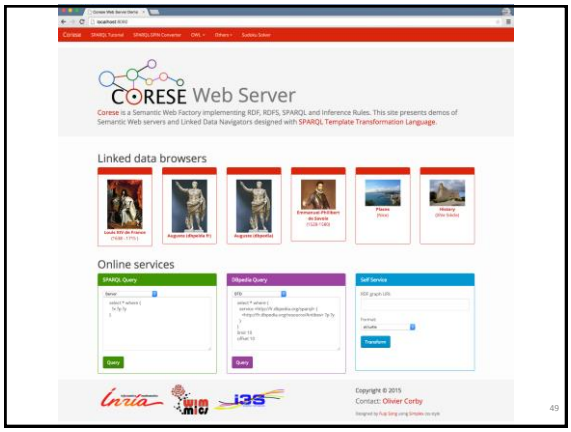
STTL engine

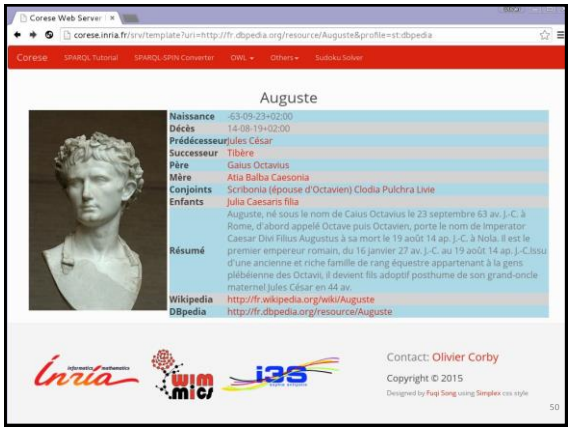
available in the Corese Semantic Web Factory

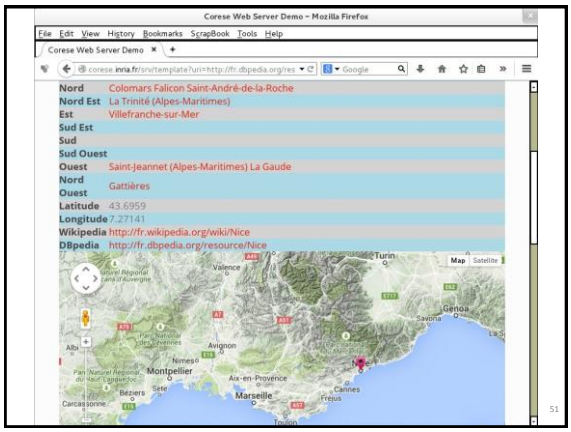
- Free download: http://wimmics.inria.fr/corese
  - SPARQL engine
  - STTL engine
  - Standalone environment to develop transformation
  - SPARQL endpoint
  - STTL server
- Web Server: http://corese.inria.fr

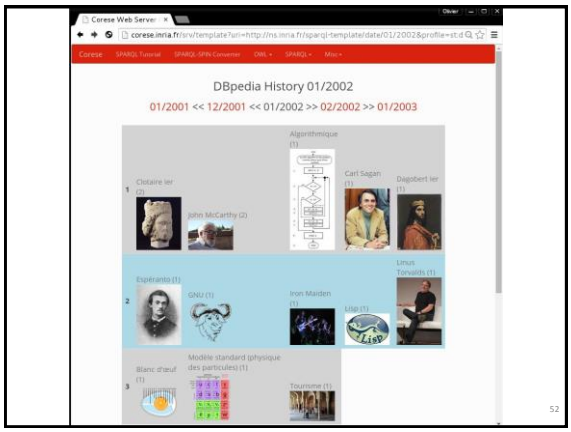
STTL Server

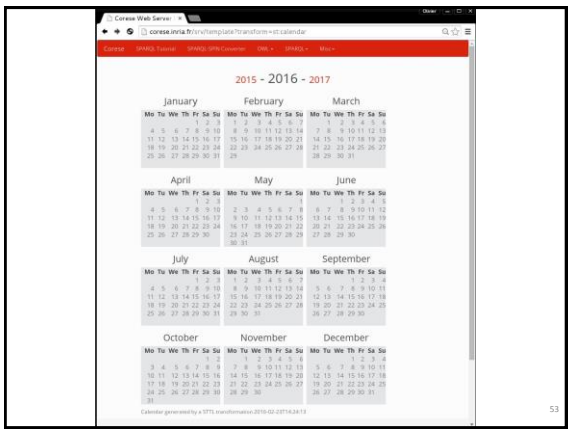


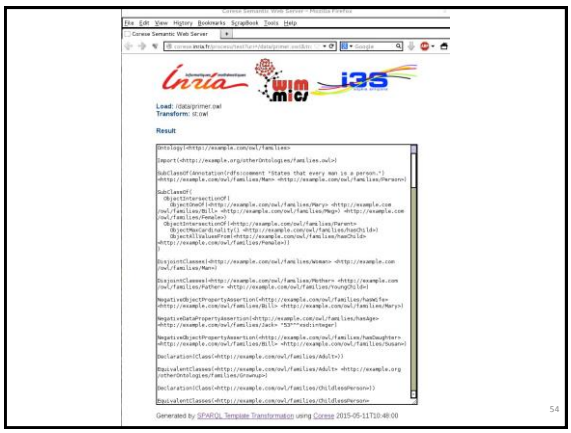

CORESE Web Server
Linked data browsers
Online services
SPARQL Query
DBpedia Query
Self Service
Contact: Olivier Corby
49

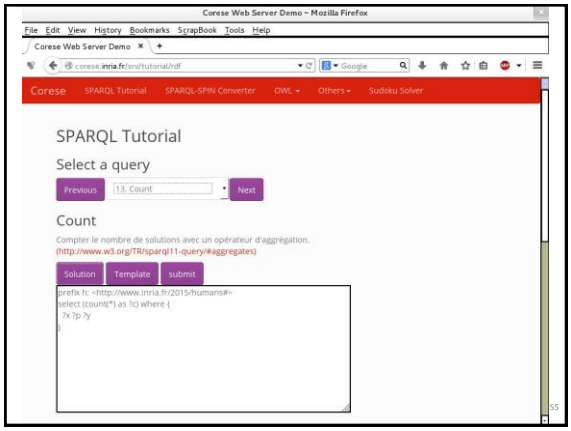

Auguste
Naissance -63-09-23+02:00
Décès 14-08-19+02:00
Prédécesseur Jules César
Successeur Tibère
Père Gaius Octavius
Mère Atia Balba Caesonia
Conjoints Scribonia (épouse d'Octavien) Clodia Pulchra Livie
Enfants Julia Caesaris filia
Résumé Auguste, né sous le nom de Caius Octavius le 23 septembre 63 av. J.-C. à Rome, d'abord appelé Octave puis Octavien, porte le nom de Imperator Caesar Divi Fillius 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
Contact: Olivier Corby
Copyright © 2015
50

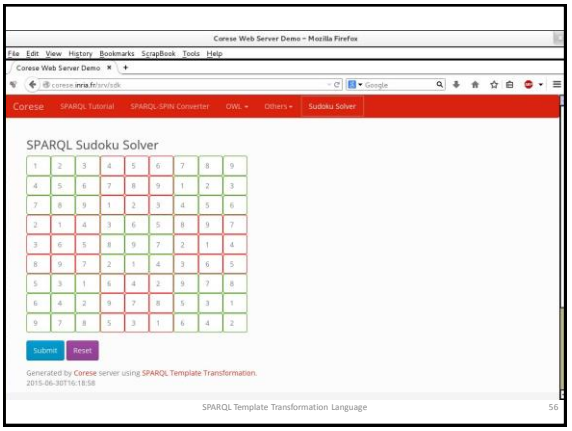

Corese Web Server Demo - Mozilla Firefox
Nord Colomars Falicon Saint-André-de-la-Roche
Nord Est La Trinité (Alpes-Maritimes)
Est Villefranche-sur-Mer
Sud Est
Sud
Sud Ouest
Ouest Saint-Jeannet (Alpes-Maritimes) La Gaude
Nord Gattières
Ouest
Latitude 43.6959
Longitude 7.27141
Wikipedia http://fr.wikipedia.org/wiki/Nice
DBpedia http://fr.dbpedia.org/resource/Nice
51


DBpedia History 01/2002
01/2001 << 12/2001 << 01/2002 >> 02/2002 >> 01/2003
52


2015 - 2016 - 2017
53


54





Conclusion

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