Reasonable Ontology Templates

Modelling patterns for RDF

By Veronika Heimsbakk

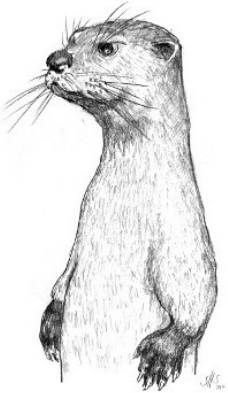
Reasonable Ontology Templates

Veronika Heimsbakk

Capgemini

- vheimsbakk
- veleda
- veronikaheim
- veronahe.no

Reasonable Ontology Templates



A box of language(s) & tools for representing and instantiating RDF modelling patterns.
It is designed to improve the efficiency and quality of building, using, and maintaining knowledge bases.

Benefits of OTTR

- Abstraction layer on top of RDF.
- Easier for non-semantic actors to use.
- Well known tools.
- CLI or automated serialisation.

Specifications

mOTTR	Concepts and Abstract Model for Reasonable Ontology Templates
rOTTR	Adapting Reasonable Ontology Templates to RDF
wOTTR	Web Reasonable Ontology Templates
stOTTR	Terse Syntax for Reasonable Ontology Templates
tabOTTR	Tabular Reasonable Ontology Template Instances
bOTTR	Batch Instantiation of OTTR templates

Specifications

mOTTR Concepts and Abstract Model for Reasonable Ontology Templates

rOTTR Adapting Reasonable Ontology Templates to RDF

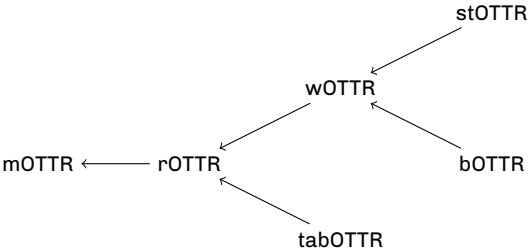
wOTTR Web Reasonable Ontology Templates

stOTTR Terse Syntax for Reasonable Ontology Templates

tabOTTR Tabular Reasonable Ontology Template Instances

bOTTR Batch Instantiation of OTTR templates

Specification dependency



Web Reasonable Ontology Templates

With wOTTR it is possible to express templates and template instances in an RDF format that is designed to be compact and readable

Terse Syntax for Reasonable Ontology Templates

This specification defines the Terse Syntax for Reasonable Ontology Templates (stOTTR) for serialising OTTR templates and instances of OTTR templates, as defined by rOTTR.

stOTTR Terms

A term is, syntactically, either a variable, a constant or a list of terms.

```
?chocolateCake
?CAKE

<http://example.com/carrotcake>
ex:bananabread
:bløtkake

[]
_:blankcake
```

stOTTR Terms

A term is, syntactically, either a variable, a constant or a list of terms.

```
"coffee"
"tea"^^xsd:normalisedString
42
true
3.14

("coffee", ex:bananabread, 42)
(("coffee", 42), ex:bananabread, (3.14) )
```

stOTTR Types

A type, i.e., the type of a term, is either a basic type, a list type or a LUB-type (least upper bound).

```
xsd:double
owl:Class
rdfs:Resource
ottt:Bot

List<xsd:string>
List<NEList<xsd:int>>
```

stOTTR Template signatures

```
ex:PlanetTemplate [ ?planet , ?label ] .
```

Two parameters, no modifiers.

```
ex:PlanetTemplate [ ! owl:Class ?planet , ? xsd:integer ?numSatellites = 0 ] .
```

`!`	non-blank modifier
`?`	optional modifier
`owl:Class`	type
`?x = 0`	default value

```
ex:PlanetTemplate [ !??planet ] .
```

`!?` non-blank optional

stOTTR Instances

```
ex:Template(ex:Coffee , ex:Tea) .

ex:Template( , ) .

ex:Template(?cake , "coffee") .

cross | ex:Template( ++(ex:Coffee , ex:Tea) ) .
```

stOTTR Example template

```
@prefix ottr: <http://ns.ottr.xyz/0.4/> .
@prefix o-rdf: <http://tpl.ottr.xyz/rdf/0.1/> .
@prefix o-rdfs: <http://tpl.ottr.xyz/rdfs/0.2/> .
@prefix ex: <http://example.org/> .

ex:PlanetTemplate[
  ottr:IRI ?identifier ,
  NEList<xsd:string> ?label ,
  ? ?aphelion
] :: {
  o-rdf:Type(?identifier, ex:Planet) ,
  cross | o-rdfs:Label(?identifier, ++?label) ,
  ottr:Triple(?identifier, ex:aphelion, ?aphelion)
} .
```

stOTTR Example instances

```
@prefix ottr: <http://ns.ottr.xyz/0.4/> .
@prefix ex: <http://example.org/> .

ex:PlanetTemplate(ex:Earth, ("Jorda"@no, "Earth"@en), "1.01671") .
ex:PlanetTemplate(ex:Mars, ("Mars"@en), "1.6660") .
ex:PlanetTemplate(ex:Jupiter, ("Jupiter"@en), ottr:none) .
```

Library of Reasonable Ontology Templates (lOTTR)

http://tpl.ottr.xyz/

tabOTTR

tabOTTR is designed to be simple to use and simple to parse. The development is driven by use cases. There are therefore constructs or types of values that may not be possible to represent in tabOTTR.

tabOTTR Prefix

#OTTR	prefix
ex	http://example.org/
unit	http://qudt.org/vocab/unit/
#OTTR	end

The following are implicitly declared for all files:

```
rdf     http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs    http://www.w3.org/2000/01/rdf-schema#
owl     http://www.w3.org/2002/07/owl#
xsd     http://www.w3.org/2001/XMLSchema#
dc      http://purl.org/dc/elements/1.1/
ottr    http://ns.ottr.xyz/templates#
```

tabOTTR Template instruction

#OTTR	template	ex:PlanetTemplate	
1	2	3	0
iri	text+	unit:AU	ignored
Id	Label	Aphelion	
ex:Mars	Mars@@en | Mars@@no	1.666	
ex:Jupiter	Júpiter@@is | Jupiter@@no	5.4588	
ex:Saturn	Saturn@@de	10.1238	
#OTTR	end		

tabOTTR Data types

`iri`	IRI
`blank`	RDF blank node
`text`	unttyped RDF literal
IRI, e.g. XSD	typed RDF literal
`auto`	individually determined by value
`X+`	an RDF list where the type of items are determined by `X`

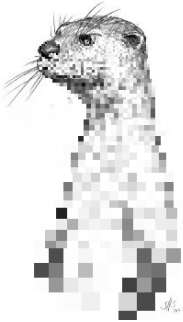
tabOTTR `auto`

Each value is individually typed according to following rules.

`iri`	string is aboslute URL or a QName.
`blank`	string starts with `_:`, fresh node `*`.
typed literal	`xsd:boolean`, `xsd:integer`, `xsd:decimal`
untyped literal	default
none value	`none`

Lutra

Lutra is an open source (LGPL) reference implementation of OTTR.



Running the Lutra CLI

1. Download latest stable `.jar` from `https://gitlab.com/ottr/lutra/lutra/-/releases`.
2. Open a terminal of choice.

Running the Lutra CLI

```
java -jar lutra.jar -l ottrlib.ttl -L stottr -I tabottr -f data.xlsx -o outputfile.ttl
```

`ottrlib.ttl`	Location of stOTTR templates.
`data.xlsx`	Instance data stored in Excel with tabOTTR flavouring.
`outputfile`	Serialized RDF.

Running the Lutra CLI

```
java -jar lutra.jar -l ottrlib.ttl -L stottr -I tabottr -f data.xlsx -o outputfile.ttl
```

`-l, -library`	Location of OTTR library.
`-L, -libraryFormat`	Input format of library, `wottr` or `stottr`.
`-I, -inputFormat`	Input format of instances, `wottr`, `stottr`, `tabottr` or `bottr`. Default: `wottr`.
`-f, -fetchMissing`	Fetch missing template dependencies. Assume that definitions are accessible via IRI. Default: false.
`-o, -output`	Path for writing output.

More flags: `https://www.ottr.xyz/#Lutra`

Using Lutra in Java

```java
import org.eclipse.rdf4j.model.Model;
import org.eclipse.rdf4j.rio.RDFFormat;
import org.eclipse.rdf4j.rio.Rio;
import xyz.ottr.lutra.cli.CLI;

public static Model runLutra() throws IOException {
  String cmd = "..." ;

  ByteArrayOutputStream outStream = new ByteArrayOutputStream();
  PrintStream out = new PrintStream(outStream, true, "UTF-8");
  CLI cli = new CLI(out, System.out);
  cli.run(cmd.split(" "));

  String result = outStream.toString();

  InputStream inputStream = new ByteArrayInputStream(result.getBytes(StandardCharsets.UTF_8));
  return Rio.parse(inputStream, inputStream.toString(), RDFFormat.TURTLE);
}
```

Demo!

https://github.com/veleda/ottr-masterclass

```
ottr-masterclass
 |- data
 | |- 2023
 |- slides
```

OTTR Events

OTTR User Forum

- 1st
- 2nd
- 3rd

Check out `https://ottr.xyz/#Latest_news` for more!

References

Specification	`https://www.ottr.xyz/`
Masterclass	`https://github.com/veleda/ottr-masterclass`
Java example	Implementation at the Norwegian Maritime Authority

#KGC2023

Join the Conversation

@KGConference @veronikaheim

linkedin.com/company/the-knowldge-graph-conference/

youtube.com/playlist?list=PLAiy7NYe9U2Gjg-600CTV1HGypiF95d_D