

# Ontology Engine

---

- [Ontology Engine](#)
  - [Follow up](#)
  - [Rules implementation remarks](#)
    - [Date Vs DateFrom](#)
    - [as-IS function imply lowercase](#)
    - [gamingtest-rules-structure inconsistent with gamingtest-rules](#)
    - [No way to know the correct language for this rule](#)
  - [Description](#)
  - [Environment](#)
    - [Libraries](#)
      - [Test framework](#)
      - [Yaml](#)
      - [RDFS](#)
        - [Online Course : Cambridge Semantics](#)
        - [Other Sources](#)
  - [Mindmatcher sources](#)
  - [References](#)

## Follow up

Date	description	author
13/05/2007	Add rules implementation remarks to be discuss	Y. Le Razer
08/05/2007	create the present document and the directory ontology_engine	Y. Le Razer

## Rules implementation remarks

### Date Vs DateFrom

```
"id": "mmr:rule-3",  
"sourcePath": "Date", --> should be dateFrom (or defaulted)  
"targetClass": "soo:Experience",  
"targetProperty": "dateFrom",  
"targetFunction": "fno:date-to-xsd"
```

### as-IS function imply lowercase

```
if rule.targetFunction == "fno:as-is":  
    currentInstance[rule.targetProperty] = str(document[rule.sourcePath]).lower()  
    continue
```

## gamingtest-rules-structure inconsistent with gamingtest-rules

Rules in the gamingtest-rules-structure are not coherent with the one of gamingtest-rules. (can't remember why)

I used the gamingtest-rules.

No way to know the correct language for this rule

```
"id": "mmr:rule-4",
"sourcePath": "Associated Soft Skill Block",
"targetClass": "soo:Skill",
"generateId": "true",
"targetFunction": "fno:search-for-mapping-with-source",
"relationTo": "soo:Experience",
"relationName": "soo:resultFromExperience",
"relationNameInverse": "soo:hasSkill"
```

```
if rule.targetFunction == "fno:search-for-mapping-with-source":
    currentInstance['prefLabel'] = {}
    currentInstance['prefLabel']['@value'] = document[rule.sourcePath]
    currentInstance['prefLabel']['@language'] = 'en'
```

## Description

The primary function of this software engine is to generate a [RDF](#) file following the model.yaml (an simplified description of an ontology), the rules of transformation and a json file with the data to be included.

This conversion involves interpreting the YAML data according to predefined transformation rules that dictate how to map YAML structures to RDF triples.

## Environment

We use [poetry](#) as dependency management and packaging in Python. This is a [cheat sheet](#) for basic usage.

### Libraries

#### Test framework

We use the [pytest](#) library : `pip install pytest`. This is [article that explain python testing with PyTest](#).

#### Yaml

We use the [pyyaml](#) library : `pip install pyyaml`. This is [an example of CRUD operations on yaml](#).

#### RDFS

Cambridge Semantics presents a RDF 101 Course.

- RDF is a graph data model.
- RDF data are directed, labeled graphs.
- A single edge in an RDF graph is a 3-tuple that is called either a statement or triple.
- Triples are organized into named graphs, forming 4-tuples, or quads.
- RDF resources (nodes), predicates (edges), and named graphs are labeled by URIs.
- Although preferable to reuse URIs when possible, Semantic Web technologies, including OWL and SPARQL, make it easy to resolve URI conflicts, as we'll see in future lessons.

#### Other Sources

- <https://www.easyrdf.org/docs/rdf-formats-json>

## Mindmatcher sources

07/05/2024 18h44 - Florent provide in [Slack](#) a [Definition files in RDFS](#).

## References

- [Python Naming Convention](#)

Rules