

Protégé + Fuseki + Jena-OWL + YASGUI

Installation and execution example

Semantic Web course 2013, Stefan Schlobach and Ronald Siebes

About

A manual to set up Fuseki with Jena-owl reasoning over an ontology created in Protégé and querying it via YASGUI.

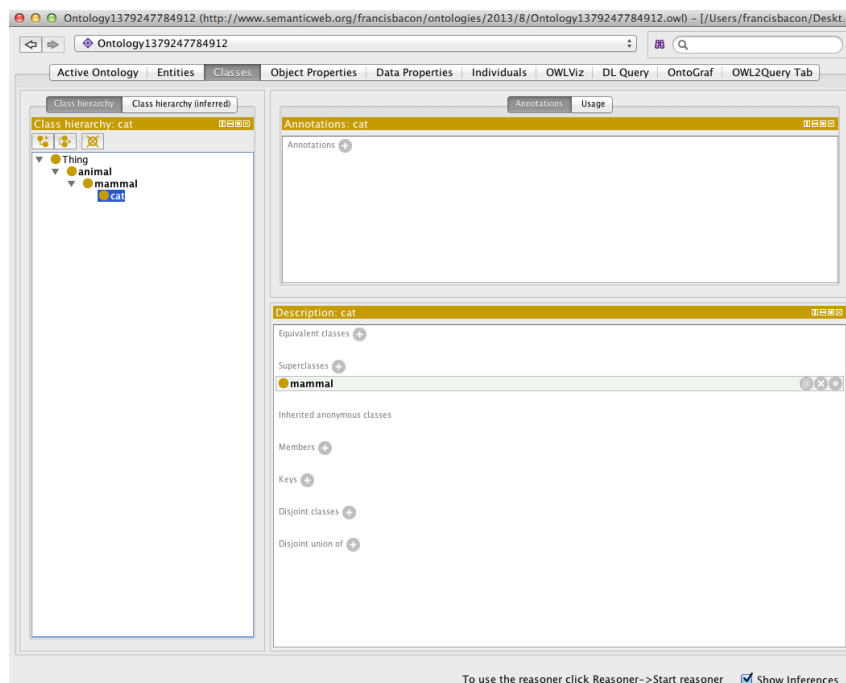
Software used in this example

Protégé Desktop 4.3: <http://protege.stanford.edu/download/protege/4.3/zip/?C=M;O=D>

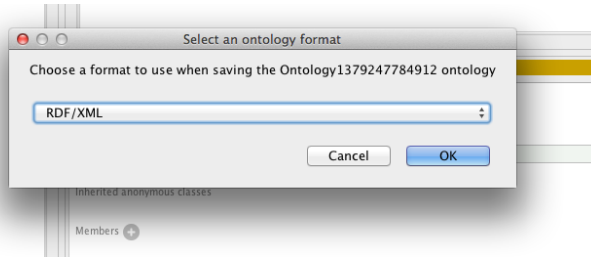
Jena-Fuseki-0.2.7: <http://www.apache.org/dist/jena/binaries/jena-fuseki-0.2.7-distribution.zip>

YASGUI: <http://yasgui.laurensrietveld.nl/>

Step 1: create and save an ontology in Protégé



Create three classes, where mammal is subclass of animal and cat subclass of mammal



Save the ontology in RDF/XML format

Step 2: install, setup and run Fuseki

- download Jena-Fuseki
- unpack it
- go to the installation folder and open config-inf-tdb.ttl with a texteditor
- paste the following:

```
# Licensed under the terms of http://www.apache.org/licenses/LICENSE-2.0

@prefix :      <#> .
@prefix fuseki: <http://jena.apache.org/fuseki#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tdb:   <http://jena.hpl.hp.com/2008/tdb#> .
@prefix ja:    <http://jena.hpl.hp.com/2005/11/Assembler#> .

[ ] rdf:type fuseki:Server ;
    fuseki:services (
        <#service1>
    ) .

# Custom code.
[ ] ja:loadClass "com.hp.hpl.jena.tdb.TDB" .

# TDB
tdb:DatasetTDB rdfs:subClassOf ja:RDFDataset .
tdb:GraphTDB   rdfs:subClassOf ja:Model .

## -----
## Service with only SPARQL query on an inference model.
## Inference model bbase data in TDB.

<#service1> rdf:type fuseki:Service ;
    fuseki:name "inf" ; # http://host/inf
    fuseki:serviceQuery "sparql" ; # SPARQL query service
    fuseki:serviceUpdate "update" ;
    fuseki:serviceUpload "upload" ; # Non-SPARQL upload service
    fuseki:serviceReadWriteGraphStore "data" ; # SPARQL Graph store protocol (read and
write)
    # A separate read-only graph store endpoint:
    fuseki:serviceReadGraphStore "get" ; # SPARQL Graph store protocol (read
only)

    fuseki:dataset <#dataset> ;
    .

<#dataset> rdf:type ja:RDFDataset ;
    ja:defaultGraph <#model_inf> ;
    .

<#model_inf> a ja:InfModel ;
    ja:baseModel <#tdbGraph> ;
    ja:reasoner [
        ja:reasonerURL <http://jena.hpl.hp.com/2003/OWLFBRuleReasoner>
    ] .

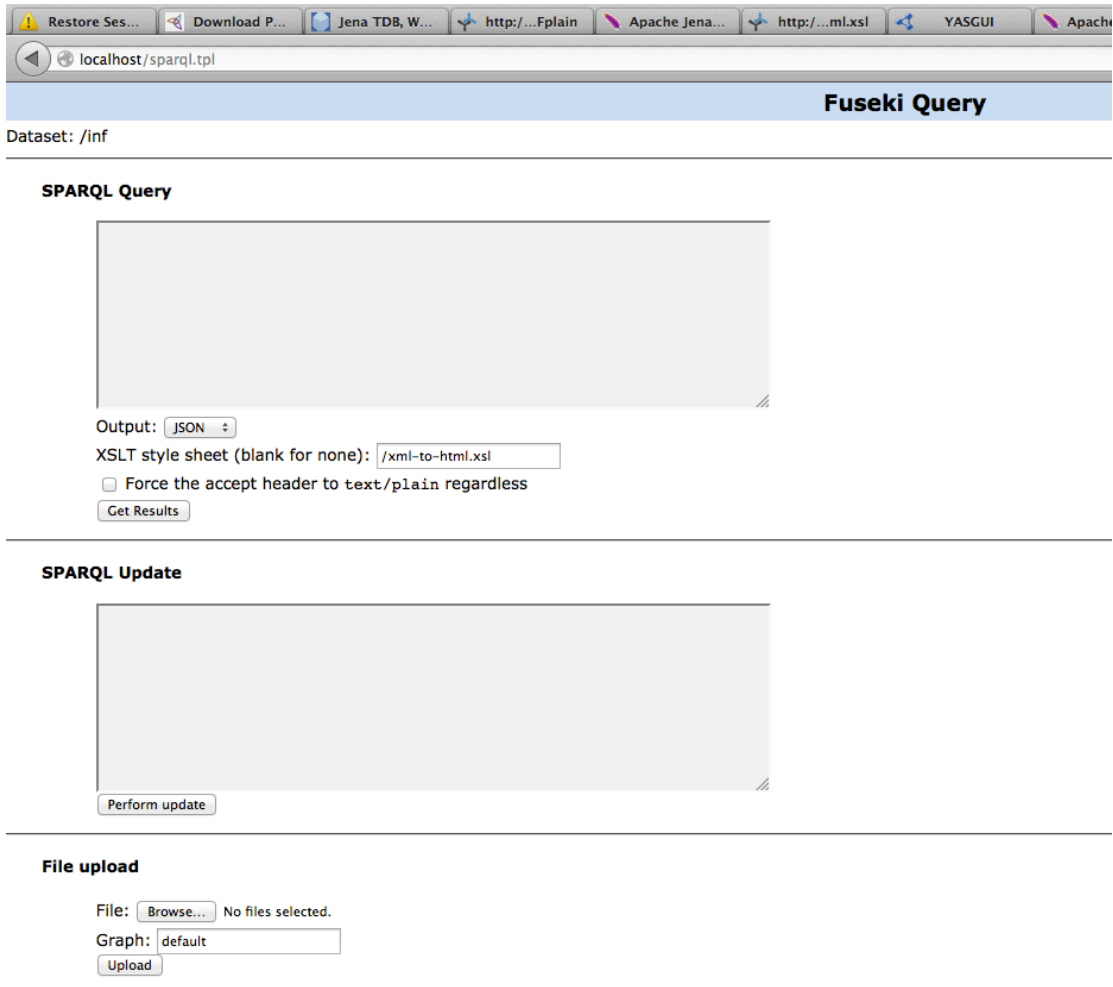
<#tdbDataset> rdf:type tdb:DatasetTDB ;
    tdb:location "DB" ;
    # If the unionDefaultGraph is used, then the "update" service should be removed.
    # The unionDefaultGraph is read only.
    # tdb:unionDefaultGraph true ;
    .

<#tdbGraph> rdf:type tdb:GraphTDB ;
    tdb:dataset <#tdbDataset> .
```

- execute (LINUX/MAC) "sudo ./fuseki-server --port=80 --update --config=config-inf-tdb.ttl" (port 80 is important for it to work with yasgui), or (WINDOWS) "fuseki-server.bat --port=80 --update --config=config-inf-tdb.ttl" (port 80 is important for it to work with yasgui)
- open <http://localhost/control-panel.tpl> in your browser

Step 3: load and query your ontology via SPARQL

- select the /inf dataset



The screenshot shows the Fuseki Query web interface. The browser address bar displays 'localhost/sparql.tpl'. The page title is 'Fuseki Query'. Below the title, the dataset is set to '/inf'. The interface is divided into three main sections:

- SPARQL Query:** A large text area for entering a query. Below it, the output format is set to 'JSON'. The XSLT style sheet is set to '/xml-to-html.xsl'. There is a checkbox for 'Force the accept header to text/plain regardless' which is unchecked. A 'Get Results' button is present.
- SPARQL Update:** A large text area for entering an update query. A 'Perform update' button is located at the bottom.
- File upload:** A section for uploading an ontology file. It includes a 'File:' field with a 'Browse...' button and the text 'No files selected.'. Below it is a 'Graph:' field with the value 'default' and an 'Upload' button.

should look like this

- upload the ontology that you saved
- test it by typing the sparql query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX myOnto:
<http://www.semanticweb.org/francisbacon/ontologies/2013/8/Ontology137
9247784912.owl#>
select * where {myOnto:cat rdfs:subClassOf ?c}
```

(note that your ontology will have a different URI)

The screenshot shows the Fuseki Query interface in a browser. The address bar shows `localhost/sparql.tpl`. The page title is "Fuseki Query". Below the title, it says "Dataset: /inf".

SPARQL Query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX myOnto: <http://www.semanticweb.org/francisbacon/ontologies/2013/8/Ontology1379247784912.owl#>
select * where {myOnto:cat rdfs:subClassOf ?c}
```

Output:
XSLT style sheet (blank for none):
 Force the accept header to text/plain regardless

SPARQL Update

which results into:

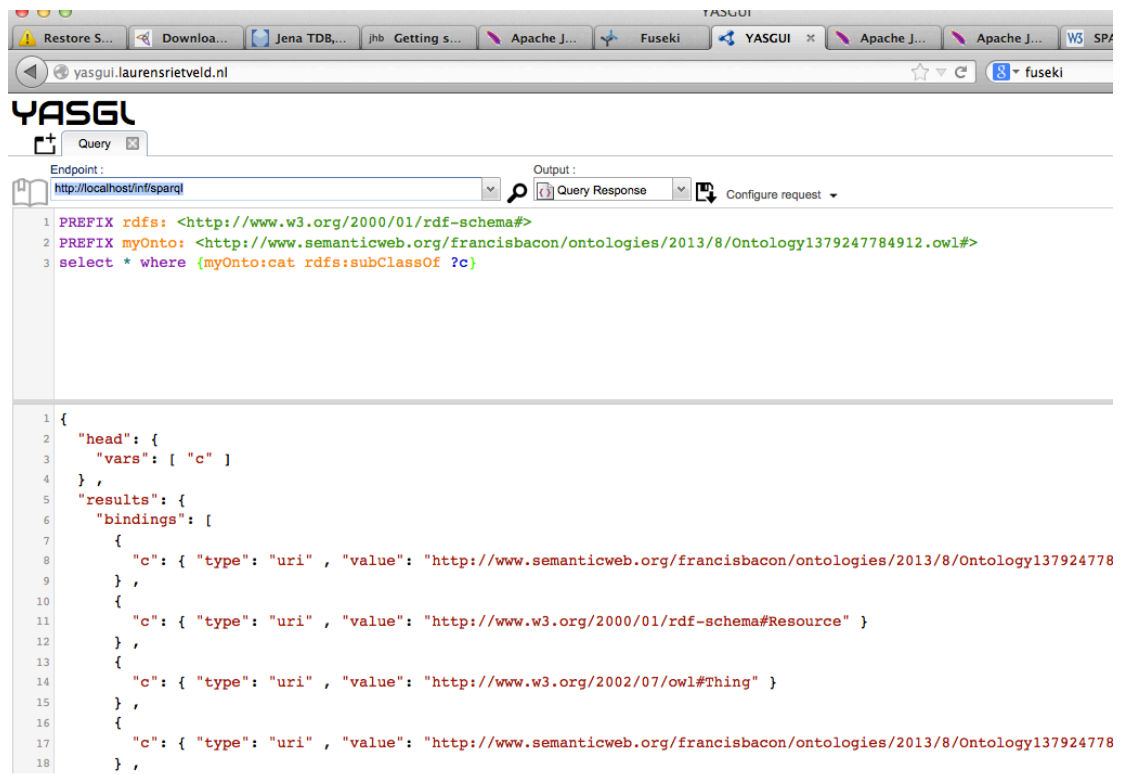
The screenshot shows the result of the SPARQL query. The address bar shows `localhost/inf/sparql?query=PREFIX+rdfs%3A+<http%3A%2F%2Fwww.w3.org%2F2000%2F01%2Frdf-sc`. The result is displayed in a dashed box:

```
| c |
|-----|
| myOnto:cat |
| rdfs:Resource |
| <http://www.w3.org/2002/07/owl#Thing> |
| myOnto:mammal |
| myOnto:animal |
```

(as you can see, the cat is an animal, thus Jena's rdfs reasoner did the job!)

Step 4: Use YASGUI to query your local endpoint

- goto <http://yasgui.laurensrietveld.nl/>
- add your local endpoint : <http://localhost/inf/sparql>
- Type your query (e.g. same as above)



The screenshot shows the YASGUI web interface in a browser. The browser's address bar displays `yasgui.laurensrietveld.nl`. The YASGUI logo is visible at the top left. Below the logo, there is a "Query" tab and an "Endpoint:" field containing `http://localhost/inf/sparql`. To the right of the endpoint field, there is an "Output:" dropdown menu set to "Query Response" and a "Configure request" button. The main area of the interface contains a SPARQL query:

```
1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX myOnto: <http://www.semanticweb.org/francisbacon/ontologies/2013/8/Ontology1379247784912.owl#>
3 select * where {myOnto:cat rdfs:subClassOf ?c}
```

Below the query, the JSON-LD output is displayed:

```
1 {
2   "head": {
3     "vars": [ "c" ]
4   },
5   "results": {
6     "bindings": [
7       {
8         "c": { "type": "uri", "value": "http://www.semanticweb.org/francisbacon/ontologies/2013/8/Ontology137924778"
9       },
10      {
11        "c": { "type": "uri", "value": "http://www.w3.org/2000/01/rdf-schema#Resource" }
12      },
13      {
14        "c": { "type": "uri", "value": "http://www.w3.org/2002/07/owl#Thing" }
15      },
16      {
17        "c": { "type": "uri", "value": "http://www.semanticweb.org/francisbacon/ontologies/2013/8/Ontology137924778"
18      },
19    ]
20  }
```

--

Comments &/or Questions? Email: ronny@cs.vu.nl