“Hello Sparnatural”

How-to adapt the “hello Sparnatural” page to your own Knowledge Graph

Version : 1.0

Last modified on : june 2023

Authors : thomas.francart@sparna.fr, marie.muller@sparna.fr

Introduction

Objectives

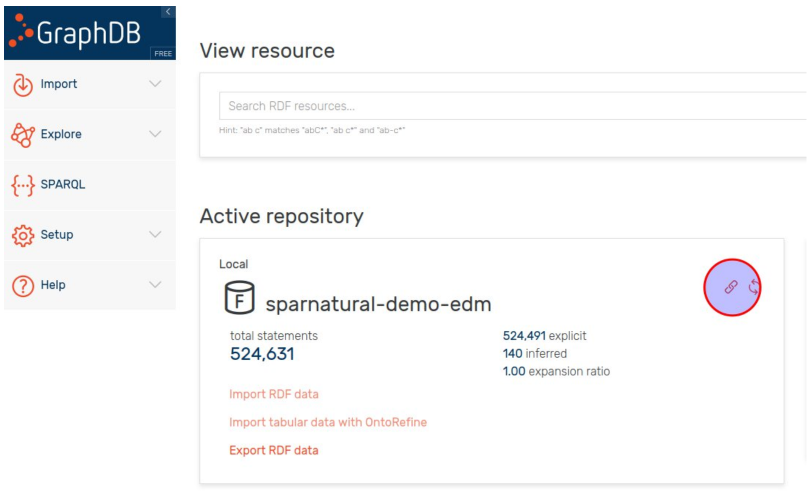
Welcome to this guide on how to setup the *"Hello Sparnatural*" page ! *"Hello Sparnatural*" is the tutorial page released with Sparnatural in each of the releases. This documentation will guide you on the necessary steps to work with this tutorial page, and start adapting it to your own knowledge graph. Once customized you can keep the tutorial page in your local machine or publish it online, provided that your SPARQL endpoint is also publicly available.

At the end of this tutorial, you will have a working HTML page demonstrating the feasibility of plugin Sparnatural on your own data. You can then explore further possibilities of configuring Sparnatural in the other guide "How to configure Sparnatural", and adapt the look-and-feel of the page to match your website.

Prerequisites

In order to follow this guide, you need the following prerequisites:

1. You need to have your own SPARQL-accessible dataset. If you don't have any SPARQL-accessible dataset, you can use the DBPedia dataset, accessible at https://dbpedia.org/sparql. GraphDB users can find their active repository SPARQL URL in the home page, in the "link" icon in the "Active repository" section:



. .

2. You need to have a basic understanding of the structure of your dataset. More specifically you need to know the URI identifiers of 2 classes in your ontology, and one property linking these two classes. If you don't know this information, try to find the documentation of your dataset, in the form of UML diagrams or documentation tables.

3. You need to have the Protégé OWL editor installed on your machine. To install Protégé, go to https://protege.stanford.edu/.
4. You need to have a basic understanding of HTML, and a basic text editor to edit an HTML page.

Structure of this guide

This guide will explain:

1. How to setup your local work environment
2. How to adjust the tutorial page to use your configuration file and your own SPARQL endpoint URL
3. How to setup a minimal configuration demonstrating how Sparnatural can work on your data
4. What are the next steps once you completed this tutorial

Setup the tutorial page

Get the tutorial page

The *"Hello Sparnatural"* tutorial page is included with each release of Sparnatural starting from 8.4.0. To get this tutorial page:

1. go to the latest release of Sparnatural
2. download the "hello-sparnatural.zip" file from the assets list
3. unzip it in a local folder on your computer

Content of the tutorial folder

The tutorial folder is based on a very simple Bootstrap HTML template, in which Sparnatural is inserted and integrated with the YasGUI SPARQL editor and query result viewer. While using YasGUI is an obvious choice, YasGUI is *not* a requirement of Sparnatural. You can choose to present query results differently than in YasGUI, and you may not want to display the raw SPARQL query to the user.

The tutorial folder is composed of the following files:

Files you will edit in this tutorial:

- **index.html**: the main HTML page that you can open in a browser and that includes the Sparnatural component.
- **config.ttl**: a Sparnatural configuration file. In this tutorial you will create your own new config file.
- **example-queries.js**: sample queries that you can change after your configuration has been adapted.

Files you can further customize after this tutorial if you want to enhance your demo:

- **main.js**: the Javascript code executed by index.html, containing Sparnatural event listeners as well as YasGUI initialization code.
- **styles.css**: the specific CSS rules for the index.html page.

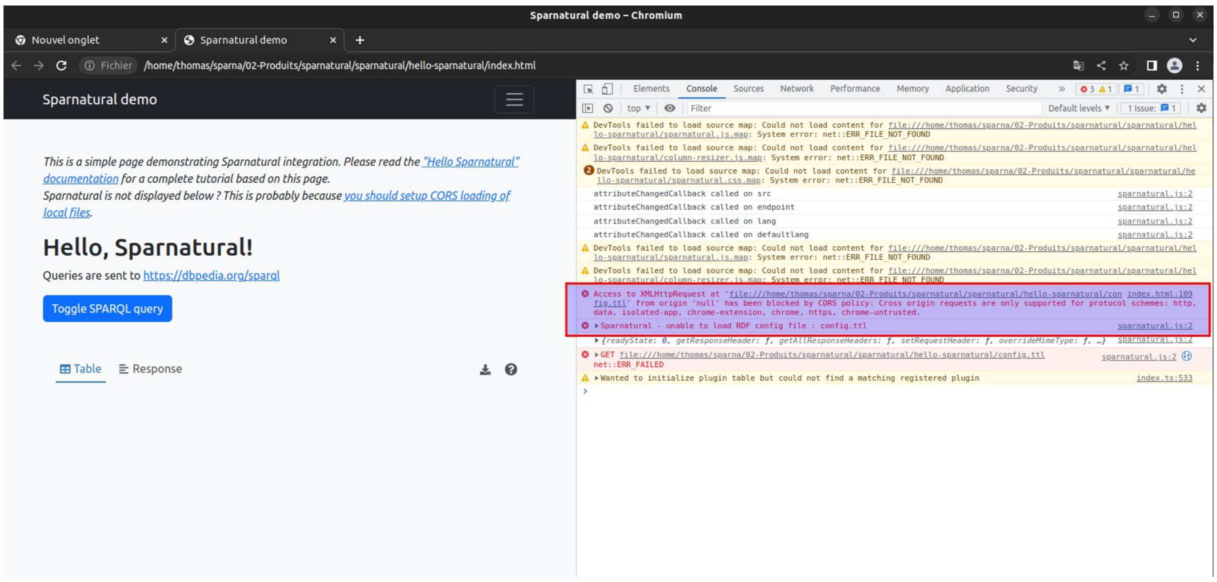
Files you will not have to edit:

- **sparnatural.js** and **sparnatural.css**: the code of the Sparnatural component, with its CSS rules
- **sparnatural-yasgui-plugins.js**: the code of the result table display plugin[1]
- **fa** subfolder: folder containing the Fontawesome free icon set.

Setup your local working environment

Allow loading of local files in your browser

Open the file index.html page from the tutorial folder in your browser. You will see a blank page with Sparnatural not loading properly, and and error in your browser console[2]:



This is a simple page demonstrating Sparnatural integration. Please read the "Hello Sparnatural" documentation for a complete tutorial based on this page.
Sparnatural is not displayed below ? This is probably because you should setup CORS loading of local files.

Hello, Sparnatural!

Queries are sent to https://dbpedia.org/sparql

```
DevTools failed to load source map: Could not load content for file:///home/thomas/sparna/02-Produits/sparnatural/sparnatural/hel
lo-sparnatural/sparnatural.js.map: System error: net::ERR_FILE_NOT_FOUND
DevTools failed to load source map: Could not load content for file:///home/thomas/sparna/02-Produits/sparnatural/sparnatural/hel
lo-sparnatural/column-resizer.js.map: System error: net::ERR_FILE_NOT_FOUND
2 DevTools failed to load source map: Could not load content for file:///home/thomas/sparna/02-Produits/sparnatural/sparnatural/he
llo-sparnatural/sparnatural.css.map: System error: net::ERR_FILE_NOT_FOUND
attributeChangedCallback called on src                                                              sparnatural.js:2
attributeChangedCallback called on endpoint                                                         sparnatural.js:2
attributeChangedCallback called on lang                                                             sparnatural.js:2
attributeChangedCallback called on defaultlang                                                      sparnatural.js:2
DevTools failed to load source map: Could not load content for file:///home/thomas/sparna/02-Produits/sparnatural/sparnatural/hel
lo-sparnatural/sparnatural.js.map: System error: net::ERR_FILE_NOT_FOUND
DevTools failed to load source map: Could not load content for file:///home/thomas/sparna/02-Produits/sparnatural/sparnatural/hel
lo-sparnatural/column-resizer.js.map: System error: net::ERR_FILE_NOT_FOUND
Access to XMLHttpRequest at 'file:///home/thomas/sparna/02-Produits/sparnatural/sparnatural/hello-sparnatural/con index.html:109
fig.ttl' from origin 'null' has been blocked by CORS policy: Cross origin requests are only supported for protocol schemes: http,
data, isolated-app, chrome-extension, chrome, https, chrome-untrusted.
▸Sparnatural - unable to load RDF config file : config.ttl                                          sparnatural.js:2
▸{readyState: 0, getResponseHeader: f, getAllResponseHeaders: f, setRequestHeader: f, overrideMimeType: f, …}  sparnatural.js:2
▸GET file:///home/thomas/sparna/02-Produits/sparnatural/sparnatural/hello-sparnatural/config.ttl     sparnatural.js:2
net::ERR_FAILED
▸Wanted to initialize plugin table but could not find a matching registered plugin                   index.ts:533
```

This is because browsers, for security reasons, disable by default the dynamic loading of other files from your local directory - in our case the Sparnatural configuration file. In order for this to work, we need to instruct the browser that it is safe to dynamically load local files. This is called "enabling CORS for local files".

/!\ Don't worry, changing these settings will not affect the other security restrictions in your browser, in particular related to CORS.

/!\ Of course, this is only in order to work with local files. Once deployed on a web server, this restriction is not applicable anymore, and normal users of your page will not have to set the same settings.

The procedure to enable CORS for local files depends on the browser:

Firefox

To make Firefox CORS-enabled for local files:

1. Open Firefox

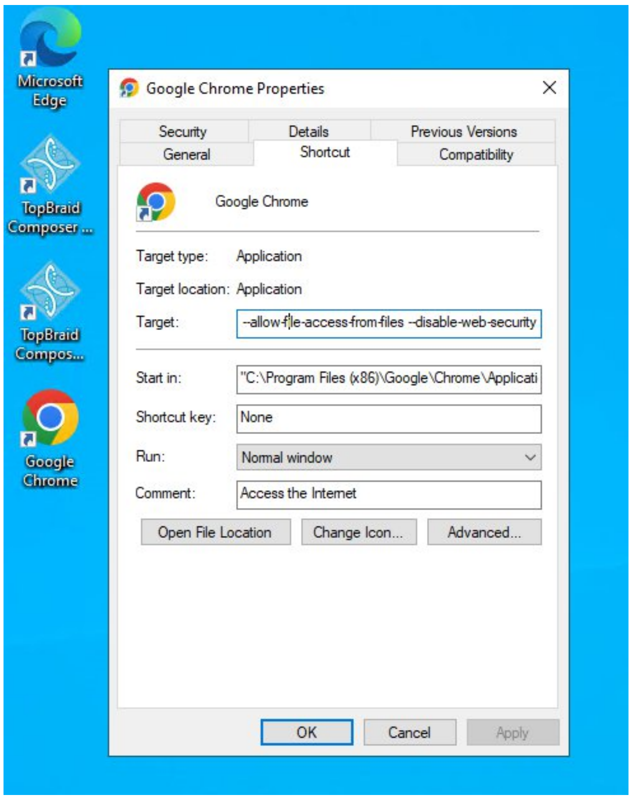
[1] This plugin is part of another project at https://github.com/sparna-git/sparnatural-yasgui-plugins
[2] The browser console can be opened with the F12 key

2. Type “about:config” in the address bar
3. Accept security warning
4. Search for the config **security.fileuri.strict_origin_policy**
5. Set this config to “false”
6. Restart your browser to make sure this is taken into account

Chrome, Chromium or Edge

To make Chrome, Chromium or Edge CORS-enabled for local files :

1. Close Chrome
2. Open a command-line or a terminal
3. Re-run chrome with the flag “**--allow-file-access-from-files**”, e.g. on Ubuntu Linux “chromium --allow-file-access-from-files”
4. You can create a shortcut with this flag set, for example in Windows, create a shortcut to Chrome and modify the command being run in the shortcut properties:

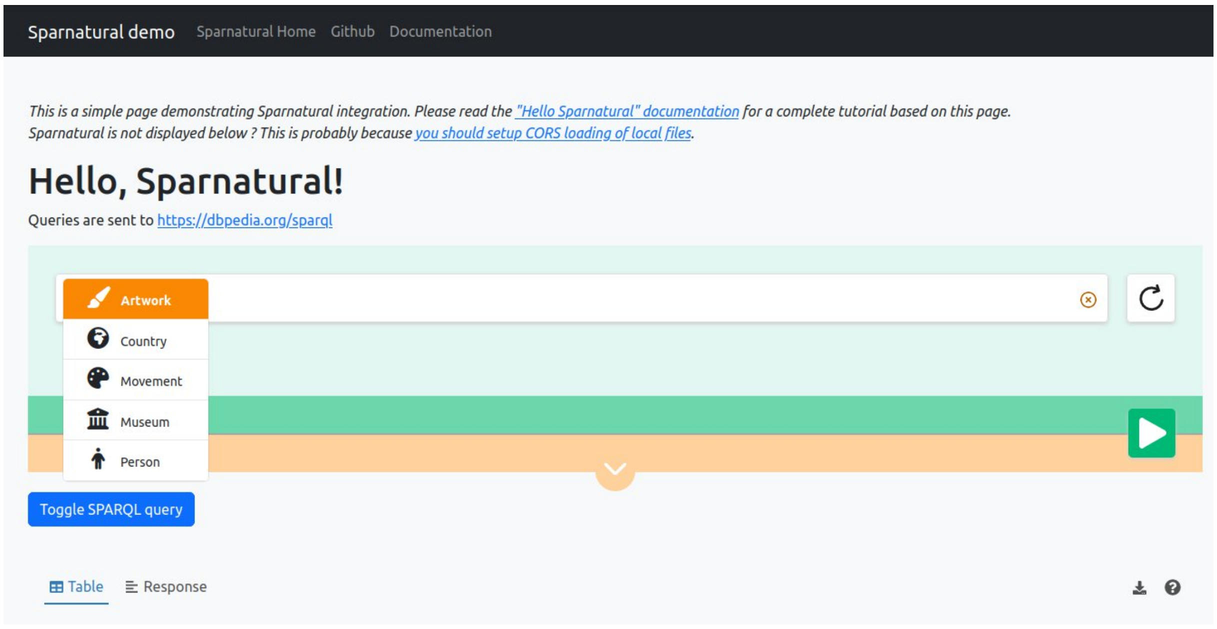


Google Chrome

Target type: Application
Target location: Application
Target: --allow-file-access-from-files --disable-web-security

Start in: "C:\Program Files (x86)\Google\Chrome\Applicati
Shortcut key: None
Run: Normal window
Comment: Access the Internet

Once you have configured this setting, try reopening the index.html page. You should see Sparnatural loaded correctly:



This is a simple page demonstrating Sparnatural integration. Please read the "Hello Sparnatural" documentation for a complete tutorial based on this page.
Sparnatural is not displayed below ? This is probably because you should setup CORS loading of local files.

Hello, Sparnatural!

Queries are sent to https://dbpedia.org/sparql

Ensure your SPARQL endpoint is CORS-enabled, or use a proxy

Why do we need CORS ?

CORS stands for Cross-Origin Resource Sharing. It is a mechanism by which a web client can retrieve resources from a different server from the one which it was originally loaded[3]. With CORS, servers can indicate that they allow clients loaded from another domain (or from certain other domains) to send them requests. Note that this restriction applies to client-server interactions, not to server-server interactions.

In the case of Sparnatural, it is the HTML page that sends the SPARQL query directly to a SPARQL service. This SPARQL service thus needs to allow clients/webpages to send queries to it even if they are loaded from another domain, otherwise the query will simply fail with a security warning, and you can see security errors in the requests of your browser console.

You thus need to make sure that the SPARQL endpoint you want to query with Sparnatural is CORS-enabled, unless in your target architecture the Sparnatural page will be served from the same domain as the SPARQL endpoint.

Check CORS

In case the SPARQL endpoint you want to query is public, the website https://www.test-cors.org/ can allow you to check if CORS is enabled. Most of the large SPARQL services already allow CORS, e.g. DBpedia, Wikidata, etc.

Allow CORS on your triplestore

The procedure to allow CORS from your triplestore varies depending on the tool.

[3] For more information we suggest to check https://enable-cors.org/, and the wikipedia page on Same Origin Policy at https://en.wikipedia.org/wiki/Same-origin_policy

GraphDB

To enable CORS on GraphDB, you need to set some specific runtime properties. Please refer to the GraphDB documentation at https://graphdb.ontotext.com/documentation/10.2/directories-and-config-properties.html?highlight=cors#workbench-properties

Virtuoso

To enable CORS on Virtuoso, follow the documentation at https://vos.openlinksw.com/owiki/wiki/VOS/VirtTipsAndTricksCORsEnableSPARQLURLs

…or use Sparnatural SPARQL proxy

Sometimes you can't modify the triplestore parameters, or your security policy would not allow it. In that case we provide a SPARQL proxy that is CORS-enabled and that will forward the request to a target SPARQL endpoint, just acting as a bridge between Sparnatural and the target SPARQL endpoint. This SPARQL proxy is deployed at https://proxy.sparnatural.eu and documented in the Sparnatural documentation.

/!\ Of course this is only a temporary workaround, and **you must not use the online proxy server in a production environment**. You should deploy your own SPARQL proxy in that case. The code of the SPARQL proxy server is open-source in its own Github repository, just in case.

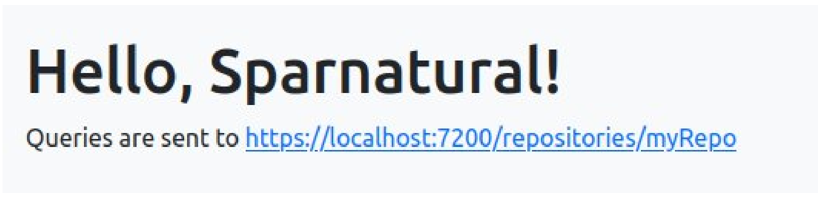
Point the tutorial page to your SPARQL service

Now that you have 1. adjusted the security settings of your browser and 2. enabled CORS on your triplestore (or used a proxy), it is time to point the tutorial page to your endpoint !

To do this, edit the index.html file, and search for the "<spar-natural" HTML element name (around line 68) and set the "endpoint" configuration attribute to your SPARQL endpoint URL (or to the proxy URL):

```
	<spar-natural
			src="config.ttl"
			endpoint="https://localhost:7200/repositories/myRepo"
			lang="en"
			defaultLang="en"
			distinct="true"
			limit="1000"
			debug="true"
	></spar-natural>
```

Now the tutorial page is configured to point to your endpoint. Reload the page in your browser and you should see the endpoint URL displayed:


Hello, Sparnatural!

Queries are sent to https://localhost:7200/repositories/myRepo

Create your first Sparnatural configuration

Setup your configuration ontology in Protégé

Now that the Sparnatural tutorial page points to your SPARQL endpoint, it is time to configure the query builder following your data structure. All this can be done via the configuration ontology you need to construct in Protégé.

Open Protégé, and start a new config ontology of yours (keep the provided config.ttl from the tutorial folder apart).

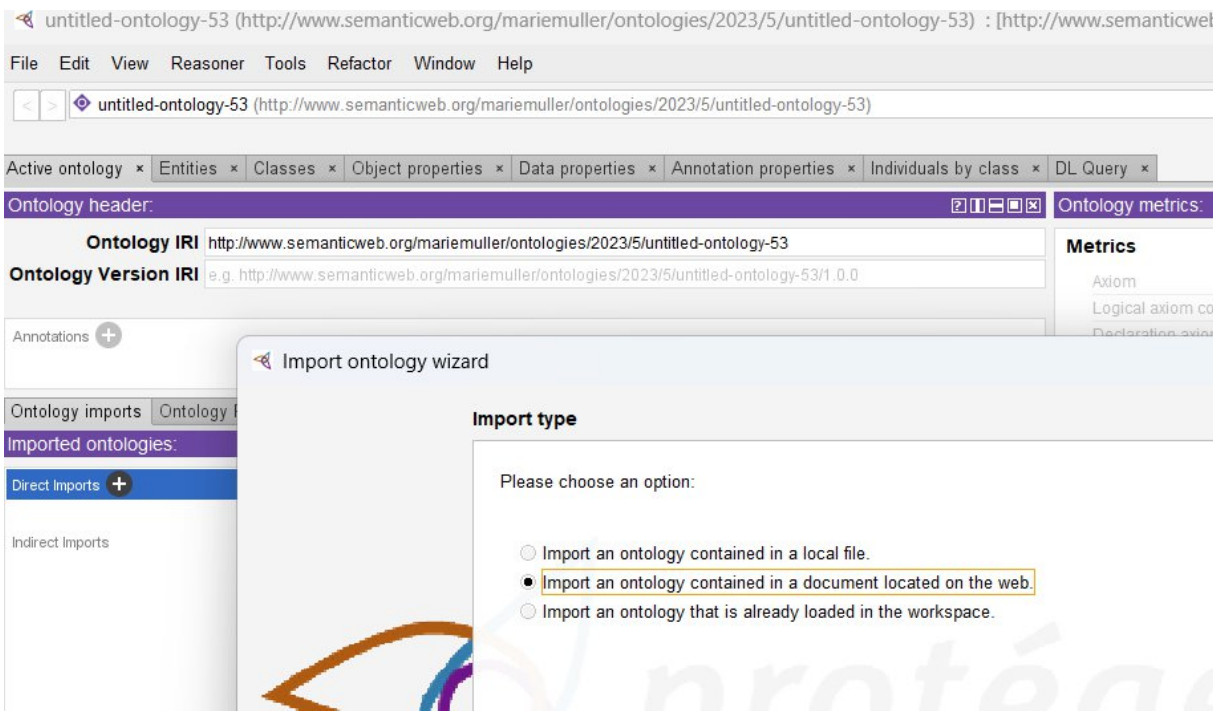
Import Sparnatural ontologies

Sparnatural comes with 2 ontologies that need to be imported (through owl:imports) in your own configuration ontology :

- A core configuration ontology at http://data.sparna.fr/ontologies/sparnatural-config-core
- A datasource configuration ontology at http://data.sparna.fr/ontologies/sparnatural-config-datasources

Import the 2 Sparnatural configuration ontologies in your ontology ;

To import these ontologies in Protégé use section under “Imported Ontologies > Direct imports” and then in the dialog select “Import an ontology contained in a document located on the web”, and then enter the URI of an ontology above. Repeat the process for the other one.


untitled-ontology-53 (http://www.semanticweb.org/mariemuller/ontologies/2023/5/untitled-ontology-53) : [http://www.semanticwe

File Edit View Reasoner Tools Refactor Window Help

untitled-ontology-53 (http://www.semanticweb.org/mariemuller/ontologies/2023/5/untitled-ontology-53)

Active ontology × | Entities × | Classes × | Object properties × | Data properties × | Annotation properties × | Individuals by class × | DL Query ×

Ontology header:

Ontology IRI: http://www.semanticweb.org/mariemuller/ontologies/2023/5/untitled-ontology-53

Ontology Version IRI: e.g. http://www.semanticweb.org/mariemuller/ontologies/2023/5/untitled-ontology-53/1.0.0

Annotations

Ontology metrics:

Metrics

Axiom

Logical axiom co

Ontology imports | Ontology

Imported ontologies:

Direct Imports

Indirect Imports

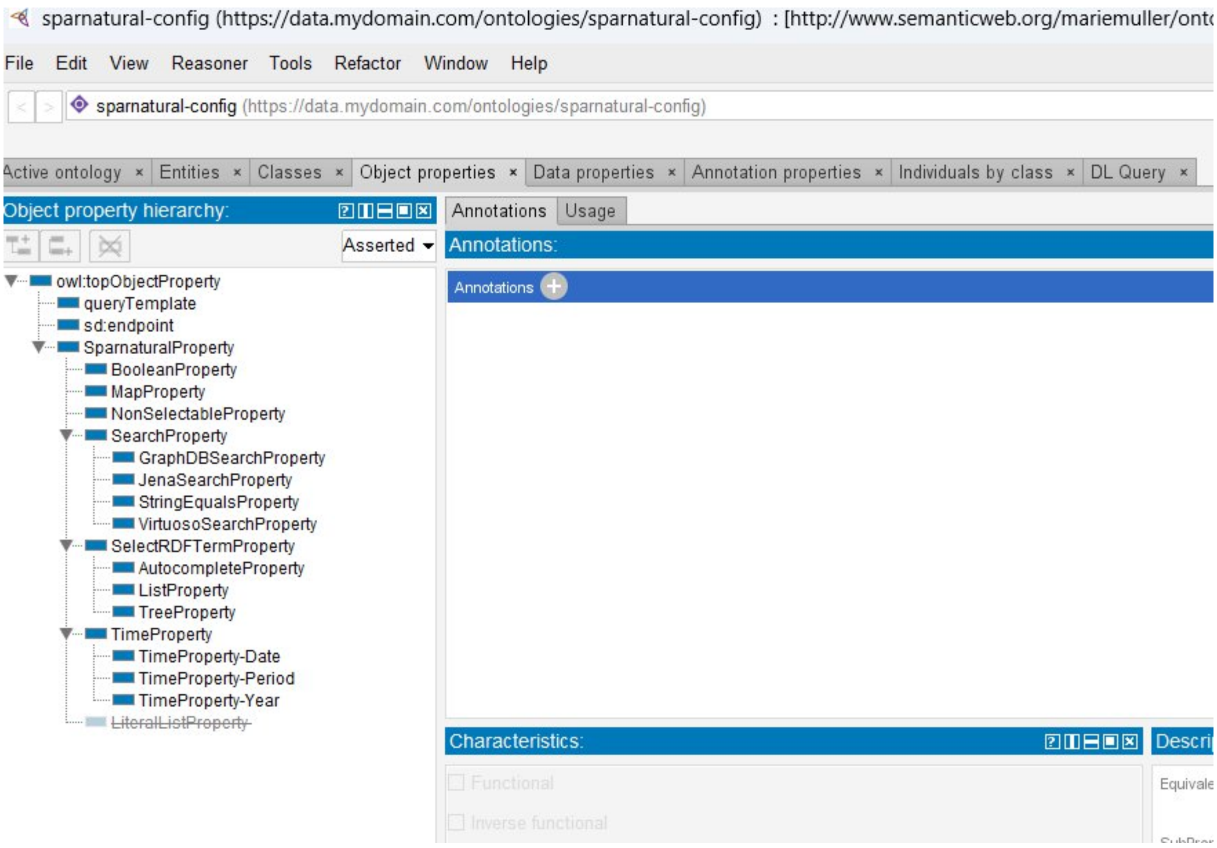
Import ontology wizard

Import type

Please choose an option:

- Import an ontology contained in a local file.
- [✓ Import an ontology contained in a document located on the web.]
- Import an ontology that is already loaded in the workspace.

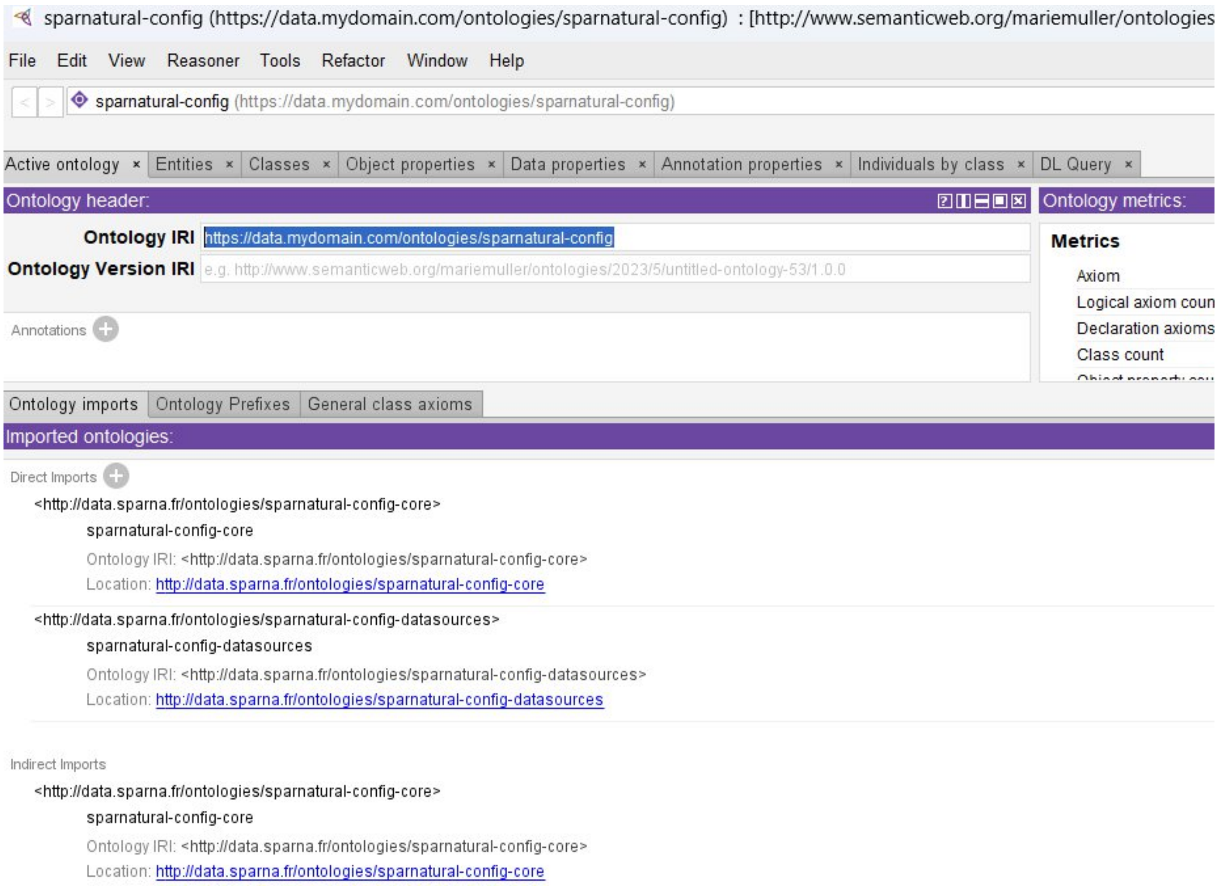
Once both ontologies are loaded in Protégé, check in the Classes and Object Properties tab of Protégé that you see now you can see imported classes and properties from Sparnatural base ontologies.



```
owl:topObjectProperty
    queryTemplate
    sd:endpoint
    SparnaturalProperty
        BooleanProperty
        MapProperty
        NonSelectableProperty
        SearchProperty
            GraphDBSearchProperty
            JenaSearchProperty
            StringEqualsProperty
            VirtuosoSearchProperty
        SelectRDFTermProperty
            AutocompleteProperty
            ListProperty
            TreeProperty
        TimeProperty
            TimeProperty-Date
            TimeProperty-Period
            TimeProperty-Year
        ~~LiteralListProperty~~
```

Specify your configuration ontology URI

Back in the “Active Ontology” tab, in the Ontology IRI field above the screen, write the https address where your configuration ontology is theoretically to be published (ex : https://data.mydomain.com/ontologies/sparnatural-config)


sparnatural-config (https://data.mydomain.com/ontologies/sparnatural-config) : [http://www.semanticweb.org/mariemuller/ontologies

File Edit View Reasoner Tools Refactor Window Help

sparnatural-config (https://data.mydomain.com/ontologies/sparnatural-config)

Active ontology × Entities × Classes × Object properties × Data properties × Annotation properties × Individuals by class × DL Query ×

Ontology header:

Ontology IRI https://data.mydomain.com/ontologies/sparnatural-config

Ontology Version IRI e.g. http://www.semanticweb.org/mariemuller/ontologies/2023/5/untitled-ontology-53/1.0.0

Annotations

Ontology metrics:

Metrics

Axiom

Logical axiom coun

Declaration axioms

Class count

Ontology imports | Ontology Prefixes | General class axioms

Imported ontologies:

Direct Imports

<http://data.sparna.fr/ontologies/sparnatural-config-core>

sparnatural-config-core

Ontology IRI: <http://data.sparna.fr/ontologies/sparnatural-config-core>

Location: http://data.sparna.fr/ontologies/sparnatural-config-core

<http://data.sparna.fr/ontologies/sparnatural-config-datasources>

sparnatural-config-datasources

Ontology IRI: <http://data.sparna.fr/ontologies/sparnatural-config-datasources>

Location: http://data.sparna.fr/ontologies/sparnatural-config-datasources

Indirect Imports

<http://data.sparna.fr/ontologies/sparnatural-config-core>

sparnatural-config-core

Ontology IRI: <http://data.sparna.fr/ontologies/sparnatural-config-core>

Location: http://data.sparna.fr/ontologies/sparnatural-config-core

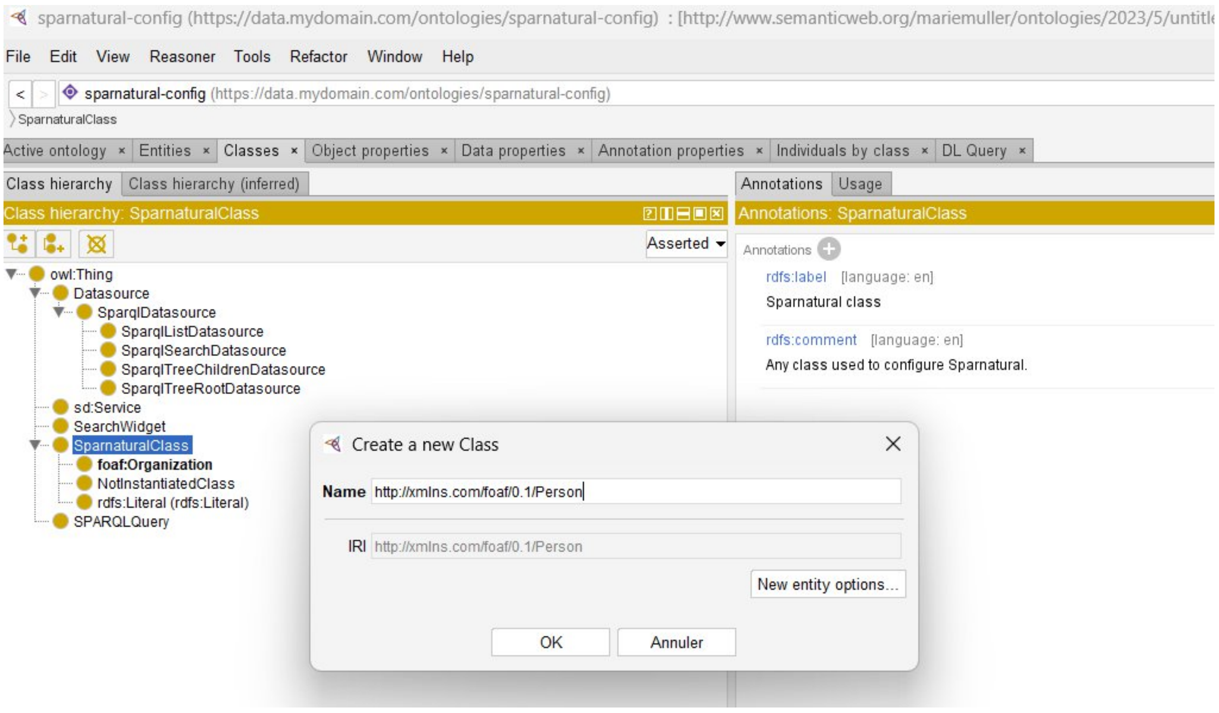
You are now ready to populate the existing structure with the entities (« Classes ») and the relations (« Object properties ») of your knowledge graph.

Create 2 classes

We will suppose in this tutorial that the knowledge graph data model consists of FOAF Persons and Organizations, with respective URI foaf:Person and foaf:Organization, linked with a property foaf:member. You will need to adapt the identifiers with your own URI identifiers.

First you need to create a class for each kind of entity : Organization and Person. Switch to the “Classes” tab for this.

Selec the “SparnaturalClass” in the class tree, and click the button to add a subclass under it. In the dialog you can directly type in the « Name » field of the dialog the real URI of the FOAF class you want to add; we will start with foaf:Person http://xmlns.com/foaf/0.1/Person .


sparnatural-config (https://data.mydomain.com/ontologies/sparnatural-config) : [http://www.semanticweb.org/mariemuller/ontologies/2023/5/untitle
File Edit View Reasoner Tools Refactor Window Help
Active ontology Entities Classes Object properties Data properties Annotation properties Individuals by class DL Query
Class hierarchy: SparnaturalClass
owl:Thing
Datasource
SparqlDatasource
SparqlListDatasource
SparqlSearchDatasource
SparqlTreeChildrenDatasource
SparqlTreeRootDatasource
sd:Service
SearchWidget
SparnaturalClass
foaf:Organization
NotInstantiatedClass
rdfs:Literal (rdfs:Literal)
SPARQLQuery
Annotations: SparnaturalClass
rdfs:label [language: en]
Sparnatural class
rdfs:comment [language: en]
Any class used to configure Sparnatural.
Create a new Class
OK
Annuler

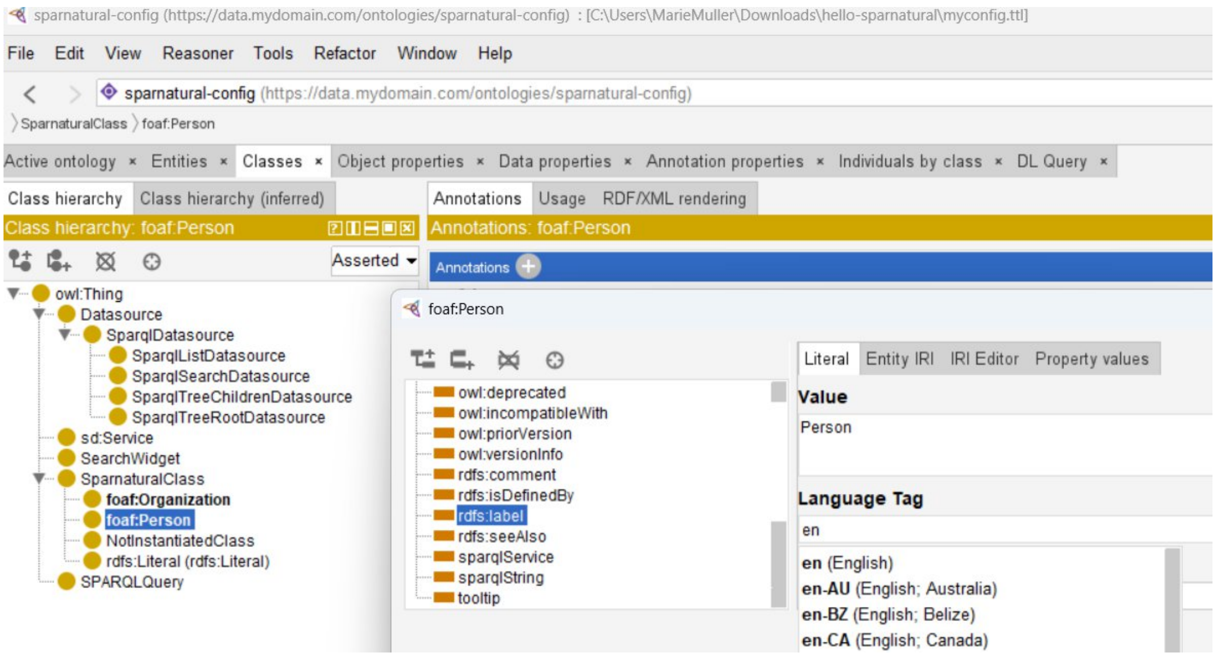
Name: http://xmlns.com/foaf/0.1/Person

IRI: http://xmlns.com/foaf/0.1/Person

New entity options...

Once created, edit the attributes of the new class by clicking the “+” button next to the « Annotations » label:

1. First add an « rdfs:label » annotation, in language “en”, with value “Person”.
    a. Note that you can create labels in as many languages as needed as the Sparnatural component can switch languages, but for the moment let’s stick to English.



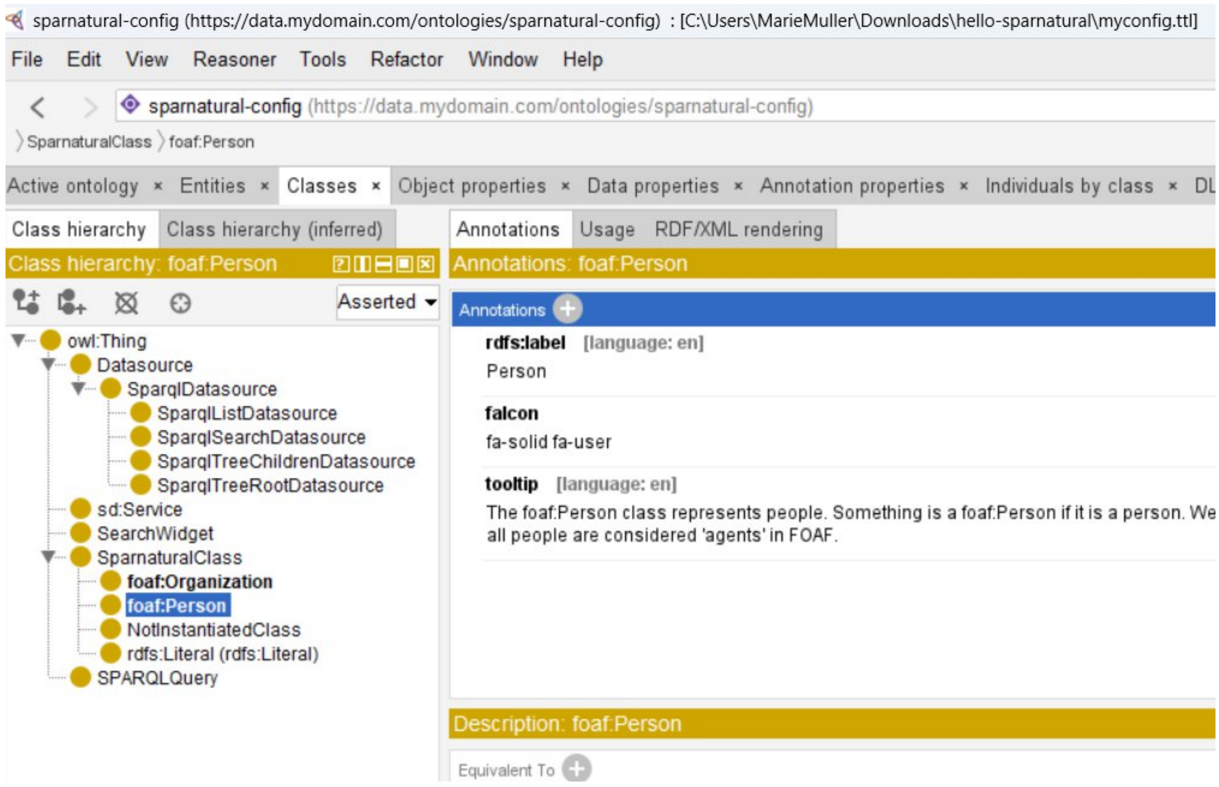
Value: Person

Language Tag: en

2. Second, add another annotation of type “tooltip” and enter a description that shall be displayed as a tooltip when the user hovers the corresponding entry in the query builder. You can enter the definition of the foaf:Person class “*The foaf:Person class represents people.*

Something is a foaf:Person if it is a person. We don't nitpic about whether they're alive, dead, real, or imaginary."

3. Add a third annotation to set the icon. Select the annotation type "faIcon" that is under "icon", and enter the value "fa-solid fa-user".
    a. In general, to search for an icon code, go on https://fontawesome.com/ website, and search for a free License icon in the search bar, note the icon code (typically something like "fa-solid fa-user"), and enter this as a value.


sparnatural-config (https://data.mydomain.com/ontologies/sparnatural-config) : [C:\Users\MarieMuller\Downloads\hello-sparnatural\myconfig.ttl]

sparnatural-config (https://data.mydomain.com/ontologies/sparnatural-config)

SparnaturalClass 〉 foaf:Person

Class hierarchy: foaf:Person

- owl:Thing
  - Datasource
    - SparqlDatasource
      - SparqlListDatasource
      - SparqlSearchDatasource
      - SparqlTreeChildrenDatasource
      - SparqlTreeRootDatasource
  - sd:Service
  - SearchWidget
  - SparnaturalClass
    - foaf:Organization
    - foaf:Person
    - NotInstantiatedClass
    - rdfs:Literal (rdfs:Literal)
  - SPARQLQuery

Annotations: foaf:Person

rdfs:label [language: en]
Person

faIcon
fa-solid fa-user

tooltip [language: en]
The foaf:Person class represents people. Something is a foaf:Person if it is a person. We
all people are considered 'agents' in FOAF.

Description: foaf:Person

Equivalent To

Repeat the same process to create http://xmlns.com/foaf/0.1/Organization, with label "Organization" in language "en", tooltip *"An organisation represents a kind of Agent corresponding to social instititutions such as companies, societies etc. "* , and icon code "fa-solid fa-building".

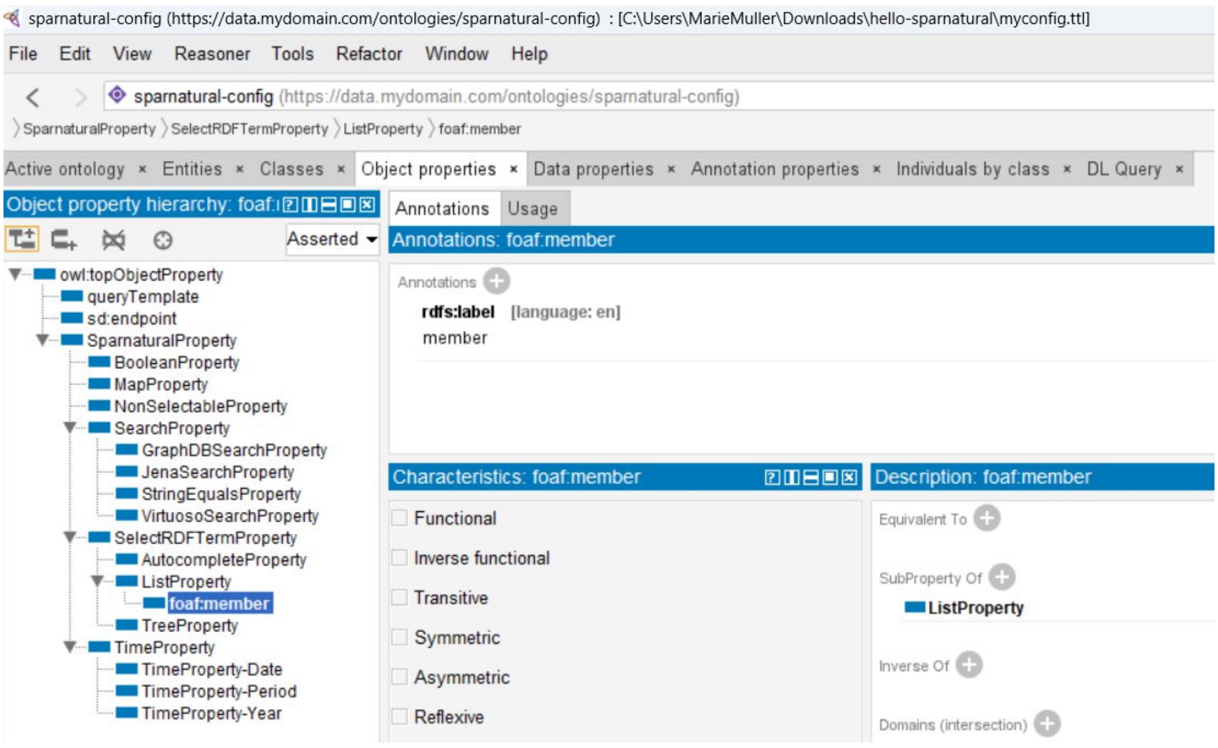
Create a property

We will now create the relationship "member" between an Organization and a Person, and we will indicate we want it to be created as a dropdown list.

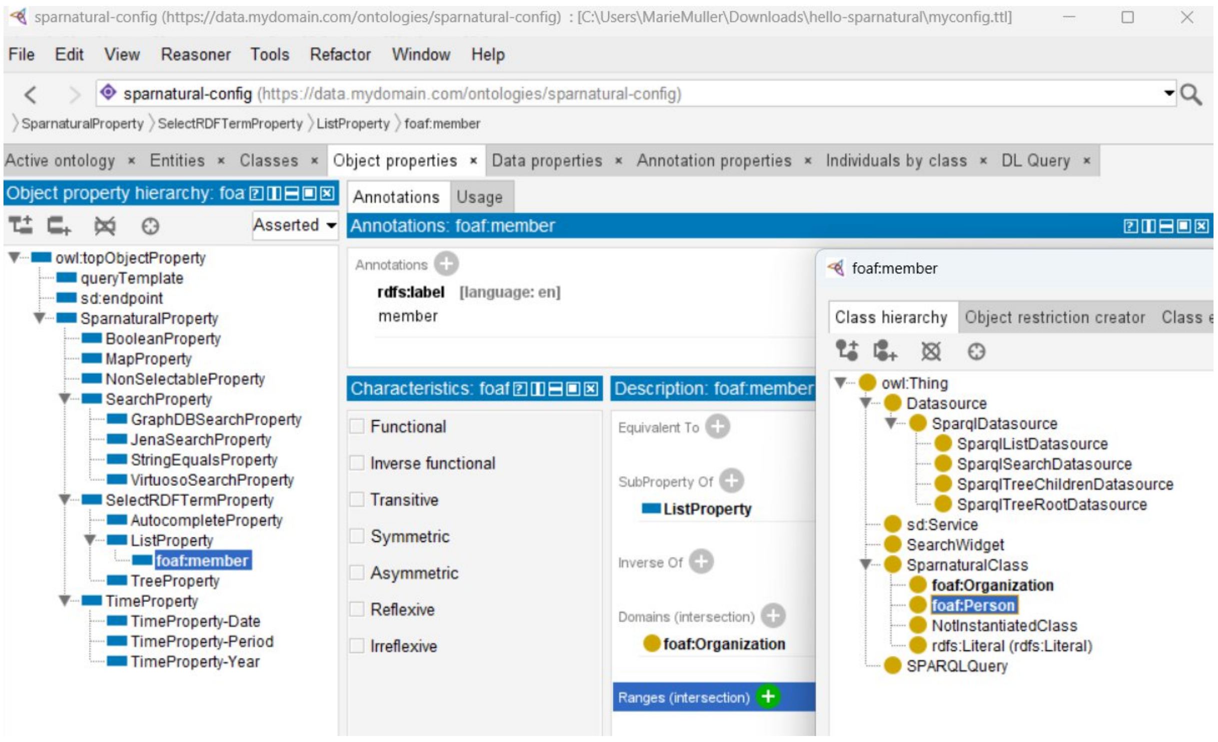
Switch into the « Object properties » tab. This is where the different kind of relations between entities can be set up as List properties, Autocomplete properties, Tree, Search fields, Map properties, Time properties and so on.

Unfold the object properties tree, select ListProperty entry, and click on the button to add a subproperty under it. Use http://xmlns.com/foaf/0.1/member as its URI.

Do as for the classes to give it an rdfs:label annotation in English.



We will relate foaf:member property to foaf:Organization class as a Domain and foaf:Person class as a Range, this you can edit in the Description area of your Protégé :

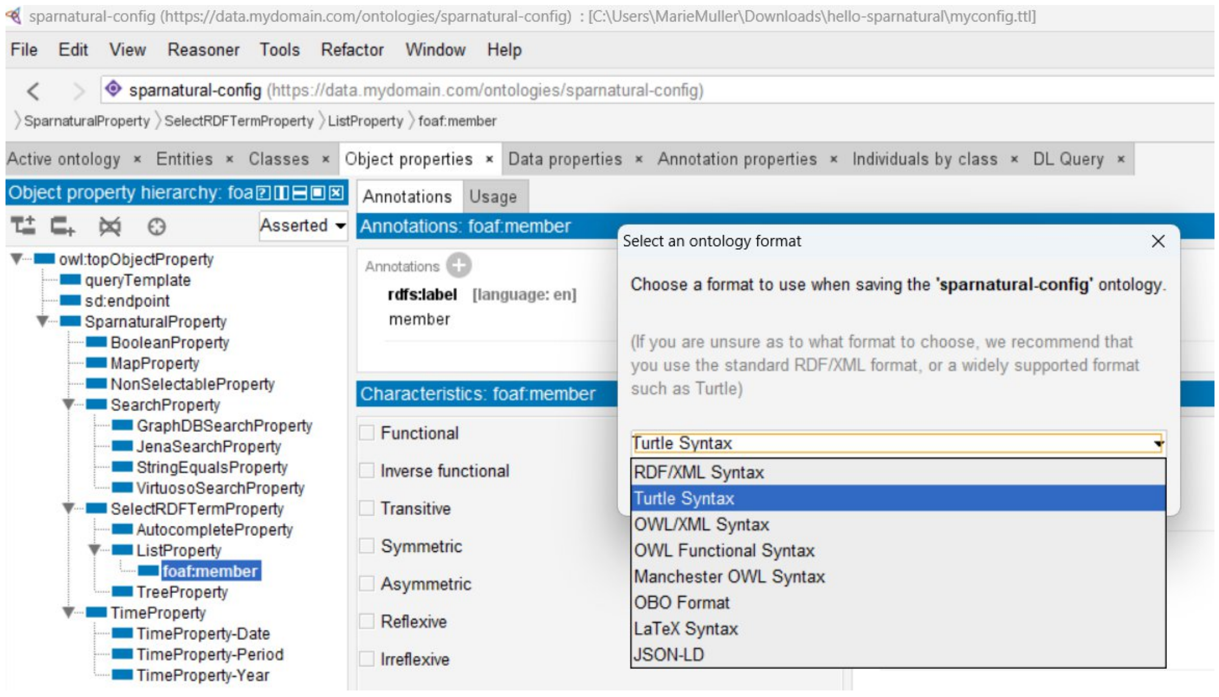


Select “foaf:Organization” as a value in the “Domains” section, and “foaf:Person” as a value in the “Ranges” section.

2 classes, one property, domain and range defined, just enough to see what your ontology looks like in Sparnatural’s interface !

Save your ontology

Save your ontology in the “hello-sparnatural” tutorial folder with a name of your choice. Save it preferably in Turtle, or in RDF/XML, but NOT in an OWL-specific serialization (such as OWL/XML). Use the file name of your choice, like “myconfig.ttl”.


sparnatural-config (https://data.mydomain.com/ontologies/sparnatural-config) : [C:\Users\MarieMuller\Downloads\hello-sparnatural\myconfig.ttl]

Select an ontology format

Choose a format to use when saving the **'sparnatural-config'** ontology.

(If you are unsure as to what format to choose, we recommend that you use the standard RDF/XML format, or a widely supported format such as Turtle)

Turtle Syntax

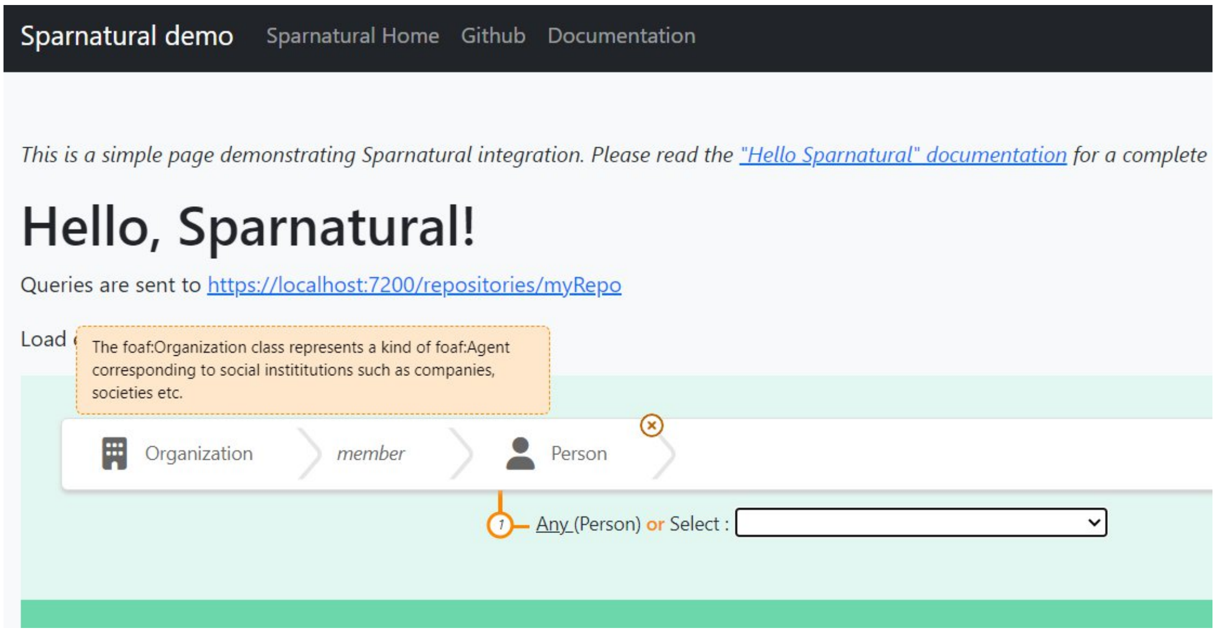
Point demo page to your config file

Edit the index.html page, look for the “<spar-natural>” tag and set the “src” attribute to point to your configuration file name, like “myconfig.ttl”, and save the file:

```
	<spar-natural
			src="myconfig.ttl"
			endpoint="https://localhost:7200/repositories/myRepo"
			lang="en"
			defaultLang="en"
			distinct="true"
			limit="1000"
			debug="true"
	></spar-natural>
```

Test and enjoy

Open a CORS-enabled browser to load the index.html file in it : you should see your 2 classes linked with your properties, populated by the data from your SPARQL endpoint:



This is a simple page demonstrating Sparnatural integration. Please read the "Hello Sparnatural" documentation for a complete

Hello, Sparnatural!

Queries are sent to https://localhost:7200/repositories/myRepo

Load

The foaf:Organization class represents a kind of foaf:Agent corresponding to social instititutions such as companies, societies etc.

If everything is OK, you should see a list of Person URIs (or another class from your knowledge graph) listed in the dropdown. You can try to build a query and verify that it works !

If it does not work:

1. check that the URIs you used in your configuration file for the 2 classes and the property do correspond to actual URI from your knowledge graph.
2. debug the SPARQL query by opening your web browser console (F12 key) and monitor the SPARQL query that is sent to populate the dropdown box.

Congratulations, you have successfully completed this “Hello Sparnatural” tutorial !

Next steps

Look at the provided configuration ontology

The tutorial folder comes with an included ontology in the “config.ttl” file. You can try loading this ontology in Protégé and have a look at the various configuration features it contains, as a source of inspiration.

Read the configuration how-to

Sparnatural offers many nice features to explore your knowledge graph :

- Autocomplete widgets, trees, maps, calendars and more.
- Ability to map the configuration ontology to the underlying knowledge graph structure, if you want to display things with a different structure.

- Ability to enable optional, negative or multilingual properties.
- Ability to customize Sparnatural through a CSS theme file.
- Ability to configure Sparnatural using a (Google or Excel) spreadsheet

All of these you can learn in the « How-to configure Sparnatural » document. The Sparnatural documentation website offers comprehensive documentation on all the features. The Github repository of Sparnatural is where you can ask questions, report bugs, and reach us.

Tune the HTML and upload the demo page online

You can further tune the content of the index.html webpage, and customize it anyway you like. Once done, you can simply upload the entire folder on a (your) web server so that it is publicly accessible !

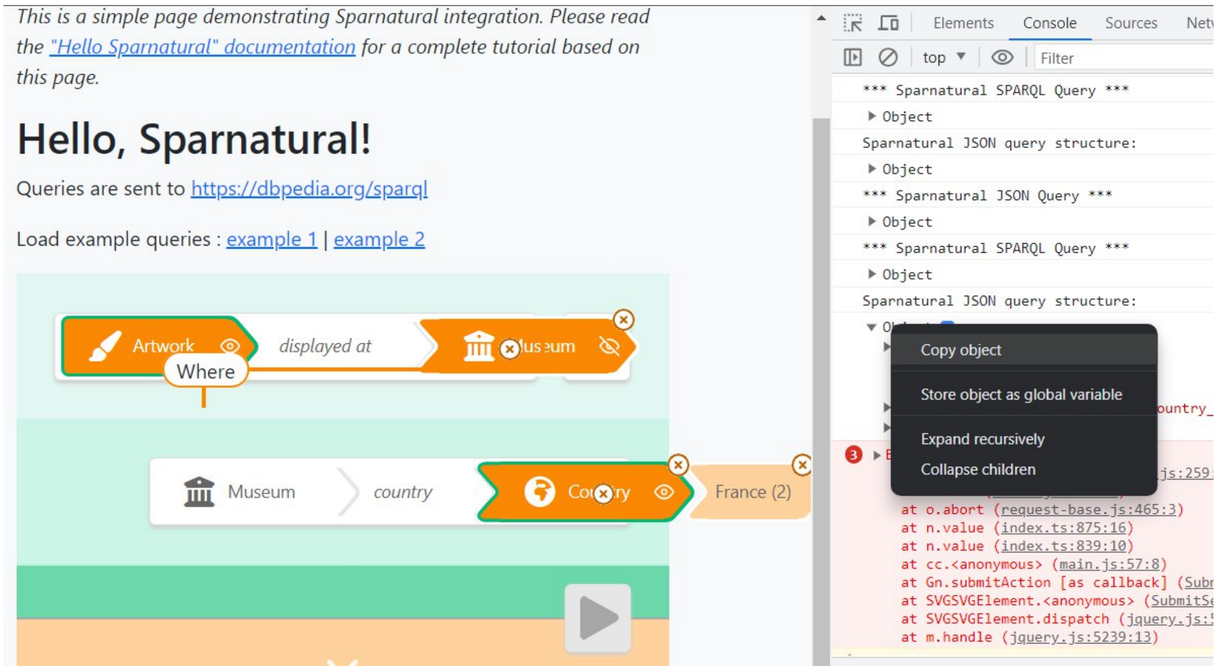
Annex : adjust the example queries

To give the users a few examples of queries they could write in the interface, you can save sample queries to display in the HTML page.

Save the sample queries only after your Sparnatural configuration ontology is stable. Otherwise there is a risk that you will need to rewrite them.

Start creating a query via the in-place query builder and open your browser console at the same time. Here you can see that each time you add a new parameter to your query, a new « Sparnatural JSON query structure » message appears.

Once your query written, then find the last « Object » in the Sparnatural JSON query structures listed here :


This is a simple page demonstrating Sparnatural integration. Please read the "Hello Sparnatural" documentation for a complete tutorial based on this page.

Hello, Sparnatural!

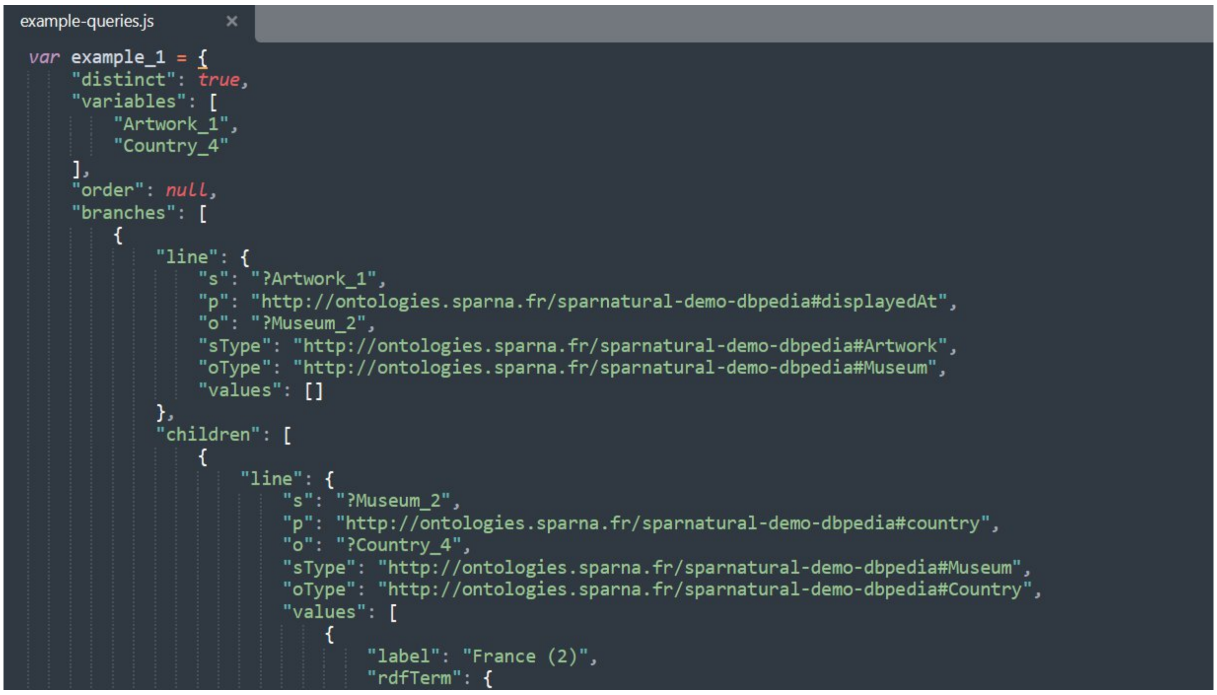
Queries are sent to https://dbpedia.org/sparql

Load example queries : example 1 | example 2

```
*** Sparnatural SPARQL Query ***
▶ Object
Sparnatural JSON query structure:
▶ Object
*** Sparnatural JSON Query ***
▶ Object
*** Sparnatural SPARQL Query ***
▶ Object
Sparnatural JSON query structure:
▼ Object
        at o.abort (request-base.js:465:3)
        at n.value (index.ts:875:16)
        at n.value (index.ts:839:10)
        at cc.<anonymous> (main.js:57:8)
        at Gn.submitAction [as callback] (Subr
        at SVGSVGElement.<anonymous> (SubmitSe
        at SVGSVGElement.dispatch (jquery.js:5
        at m.handle (jquery.js:5239:13)
```

then right click on it and select « Copy object ».

Edit the example-querues.js javascript file :


example-queries.js

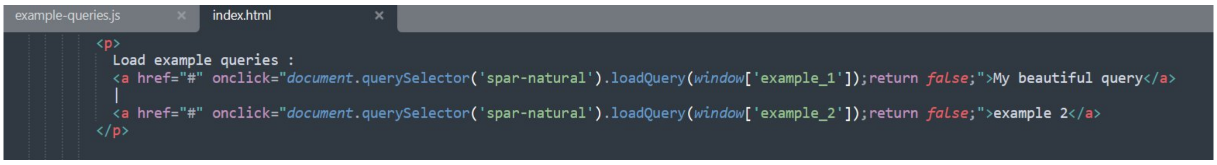
```
var example_1 = {
    "distinct": true,
    "variables": [
        "Artwork_1",
        "Country_4"
    ],
    "order": null,
    "branches": [
        {
            "line": {
                "s": "?Artwork_1",
                "p": "http://ontologies.sparna.fr/sparnatural-demo-dbpedia#displayedAt",
                "o": "?Museum_2",
                "sType": "http://ontologies.sparna.fr/sparnatural-demo-dbpedia#Artwork",
                "oType": "http://ontologies.sparna.fr/sparnatural-demo-dbpedia#Museum",
                "values": []
            },
            "children": [
                {
                    "line": {
                        "s": "?Museum_2",
                        "p": "http://ontologies.sparna.fr/sparnatural-demo-dbpedia#country",
                        "o": "?Country_4",
                        "sType": "http://ontologies.sparna.fr/sparnatural-demo-dbpedia#Museum",
                        "oType": "http://ontologies.sparna.fr/sparnatural-demo-dbpedia#Country",
                        "values": [
                            {
                                "label": "France (2)",
                                "rdfTerm": {
```

1. Select the value of "example_1" variable, from the first "{" character to the last "}", and Paste what you have copied from the console.
2. Repeat the same process to save a second query in the "example_2" variable.
3. Add more variables if needed.
4. Save the file

Edit the index.html file and uncomment the example queries section around line 57 to 64, above the “<spar-natural>” tag:

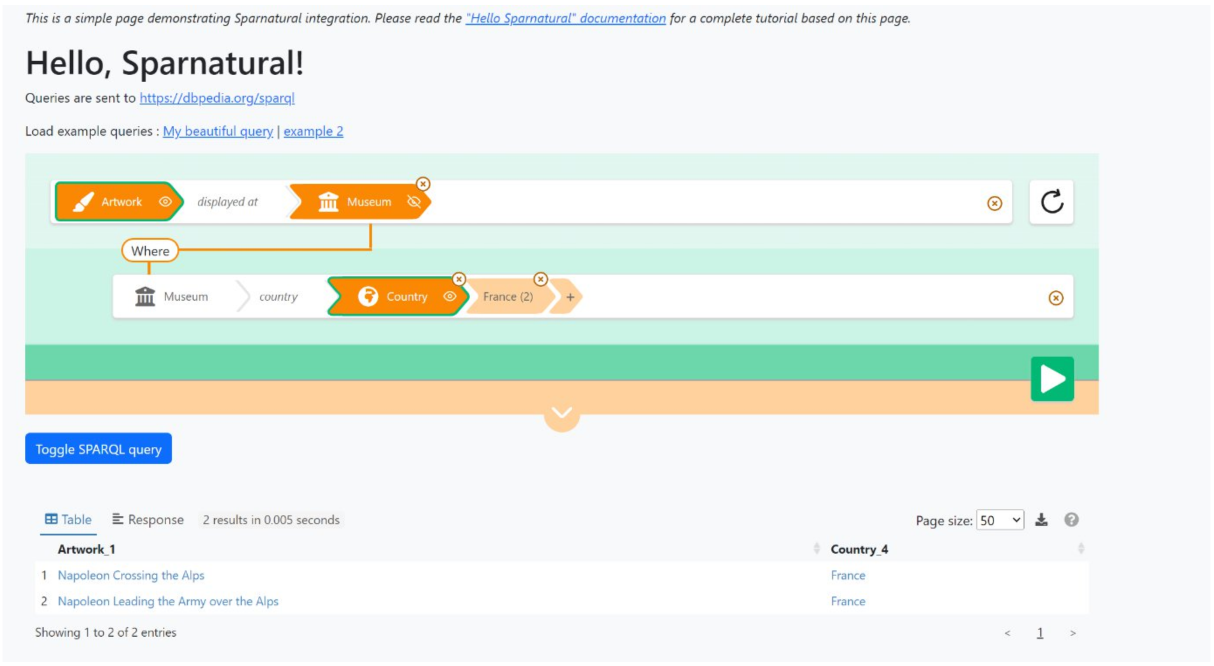
```
	<!-- uncomment to enable links to load sample queries
	<p>
	Load example queries :
	<a href="#" onclick="document.querySelector('spar-
natural').loadQuery(window['example_1']);return false;">example 1</a>
	|
	<a href="#" onclick="document.querySelector('spar-
natural').loadQuery(window['example_2']);return false;">example 2</a>
	</p>
	-->
```

Adjust the title of the query accordingly (e.g. replace “example 1” with “My beautiful query”) :



```
example-queries.js    index.html
  <p>
    Load example queries :
    <a href="#" onclick="document.querySelector('spar-natural').loadQuery(window['example_1']);return false;">My beautiful query</a>
    |
    <a href="#" onclick="document.querySelector('spar-natural').loadQuery(window['example_2']);return false;">example 2</a>
  </p>
```

Your query is saved and can be loaded in the query builder in one click !



This is a simple page demonstrating Sparnatural integration. Please read the "Hello Sparnatural" documentation for a complete tutorial based on this page.

Hello, Sparnatural!

Queries are sent to https://dbpedia.org/sparql

Load example queries : My beautiful query | example 2

2 results in 0.005 seconds

Page size: 50

	Artwork_1	Country_4
1	Napoleon Crossing the Alps	France
2	Napoleon Leading the Army over the Alps	France

Showing 1 to 2 of 2 entries