“Hello Sparnatural”

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

Version : 2

Last modified on : july 2024

Authors : [thomas.francart@sparna.fr](mailto:thomas.francart@sparna.fr), [marie.muller@sparna.fr](mailto:marie.muller@sparna.fr)

License : this document is placed under the LGPL 3.0 license, like Sparnatural

[**Introduction 2**](#_heading=h.h1whbxuxcexm)

[Objectives 2](#_heading=h.k8jwkshduose)

[**Prerequisites 2**](#_heading=h.uvf25hqig8x7)

[Structure of this guide 3](#_heading=h.ig8ijvbgovwr)

[**Setup the tutorial page 3**](#_heading=h.cohs47ulcwsl)

[Get the tutorial page 3](#_heading=h.joc4yigunf6b)

[Content of the tutorial folder 3](#_heading=h.h8hofl47d2lx)

[**Setup your local working environment 4**](#_heading=h.77tfaspfvghw)

[Allow loading of local files in your browser 4](#_heading=h.q2fa78r1eu67)

[Ensure your SPARQL endpoint is CORS-enabled, or use a proxy 6](#_heading=h.slsprvtpiut3)

[Why do we need CORS ? 6](#_heading=h.4eh7s7osnl86)

[Check CORS 6](#_heading=h.arlzudigbbwy)

[Allow CORS on your triplestore 6](#_heading=h.kuso6sldcfvt)

[…or use Sparnatural SPARQL proxy 7](#_heading=h.tnru0k1ta5c8)

[Point the tutorial page to your SPARQL service 7](#_heading=h.u9mj9hqmx97m)

[**Create your first Sparnatural configuration 8**](#_heading=h.u30wdy77ox8i)

[Setup your configuration ontology in Protégé 8](#_heading=h.nkq1438l5o55)

[Import Sparnatural ontologies 8](#_heading=h.fv8tngi3g6i7)

[Specify your configuration ontology URI 9](#_heading=h.qpmz59axukwl)

[Create 2 classes 10](#_heading=h.j17spo1lvk98)

[Create a property 12](#_heading=h.urpowic81k3v)

[Save your ontology 14](#_heading=h.tkrc2k93oyuq)

[Point demo page to your config file 15](#_heading=h.1qwtlv45059n)

[**Test and enjoy 15**](#_heading=h.edq6ccf31o4w)

[**Next steps 16**](#_heading=h.d6yju7l4w35x)

[**Annex : adjust the example queries 17**](#_heading=h.expcutf8unbu)

# Introduction

## Objectives

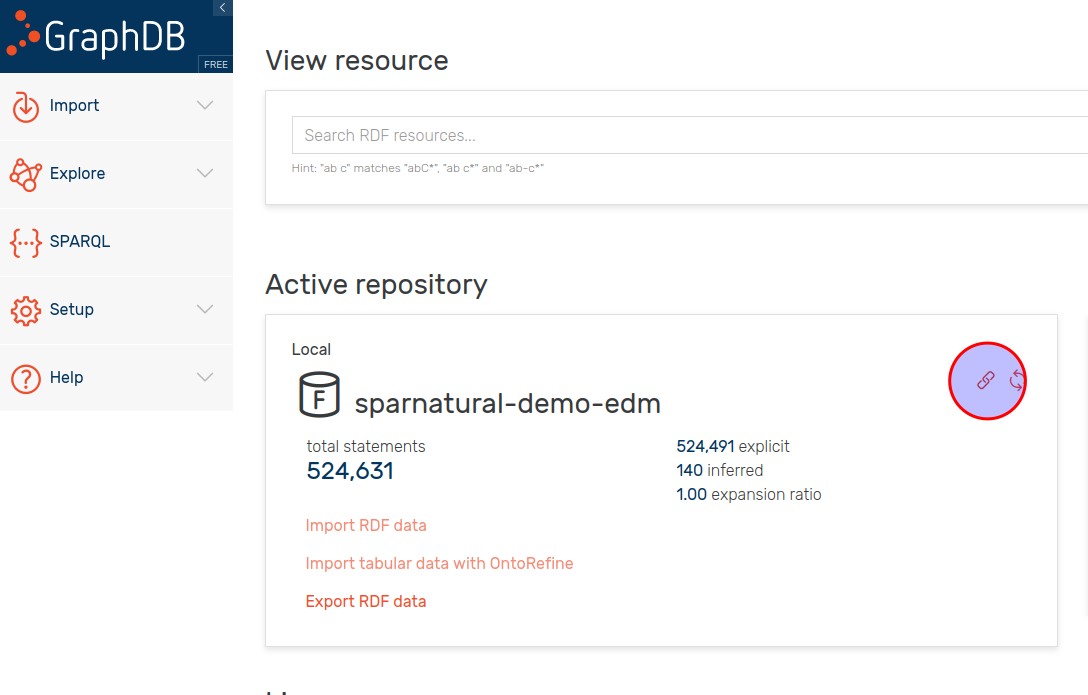
Welcome to this guide on how to setup the *“Hello Sparnatural”* page ! *“Hello Sparnatural”* is the tutorial page released with Sparnatural in each of the [releases](https://github.com/sparna-git/Sparnatural/releases). This documentation will guide you on the necessary steps to work with this tutorial 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:



1. 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.
2. You need to have a spreadsheet or online spreadsheet editor (Excel or Google Sheets for example)
3. 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](https://github.com/sparna-git/Sparnatural/releases/latest)
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](https://getbootstrap.com) HTML template, in which Sparnatural is inserted and integrated with the [YasGUI](https://yasgui.triply.cc/) 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 basic Sparnatural configuration file. In this tutorial you will create your own new config file.
* **config.xlsx** : the SHACL configuration spreadsheet from where the .ttl file will be generated
* **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.

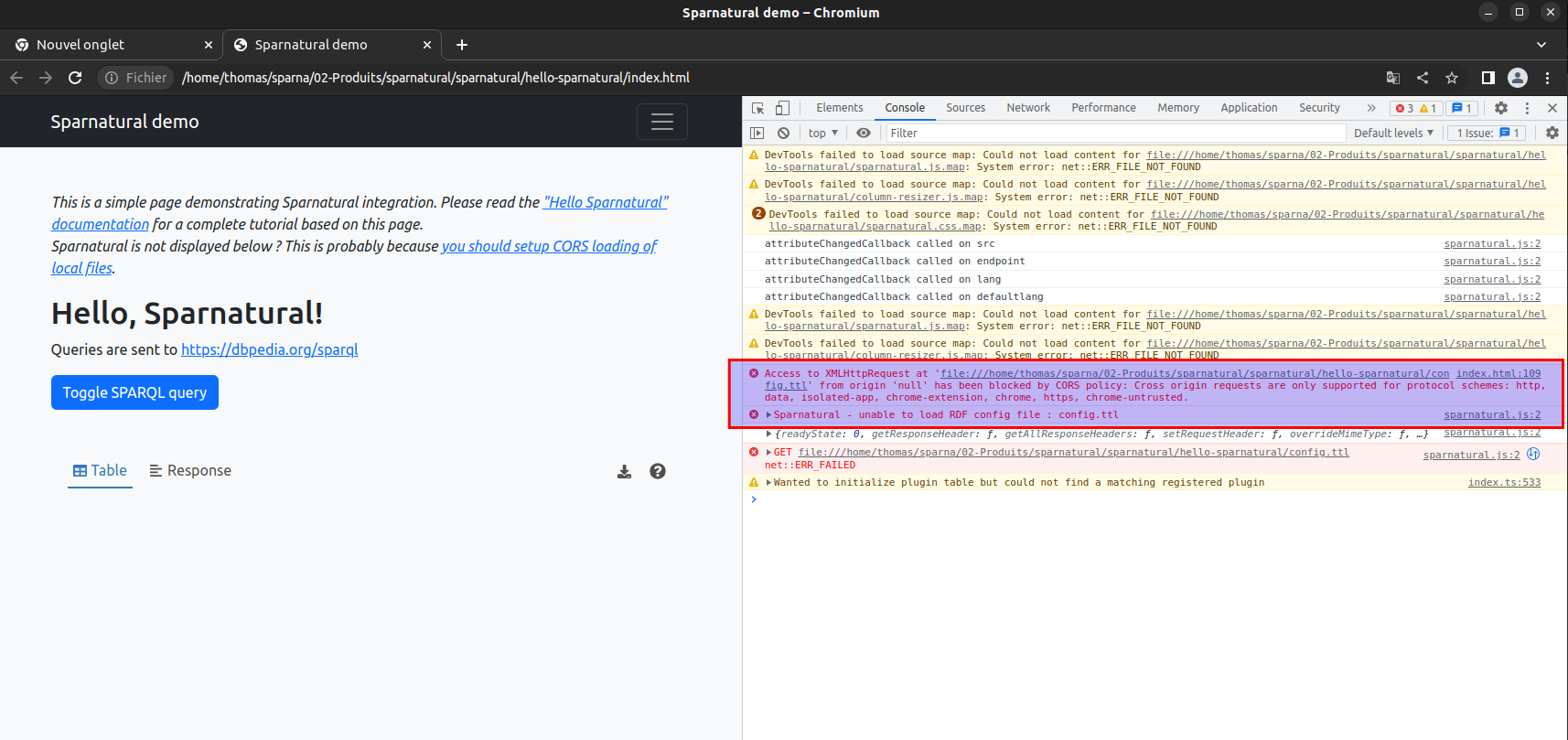
Files you will not have to edit:

* **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]](#footnote-2)
* **fa** subfolder: folder containing the [Fontawesome](https://fontawesome.com/) 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]](#footnote-3):



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**

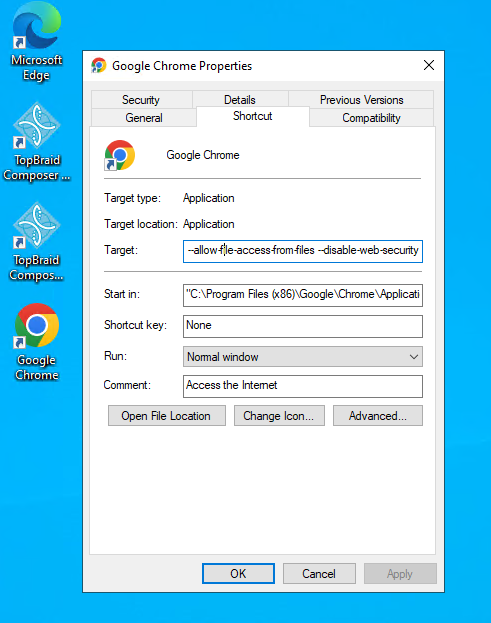
To make Firefox CORS-enabled for local files:

1. Open Firefox
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

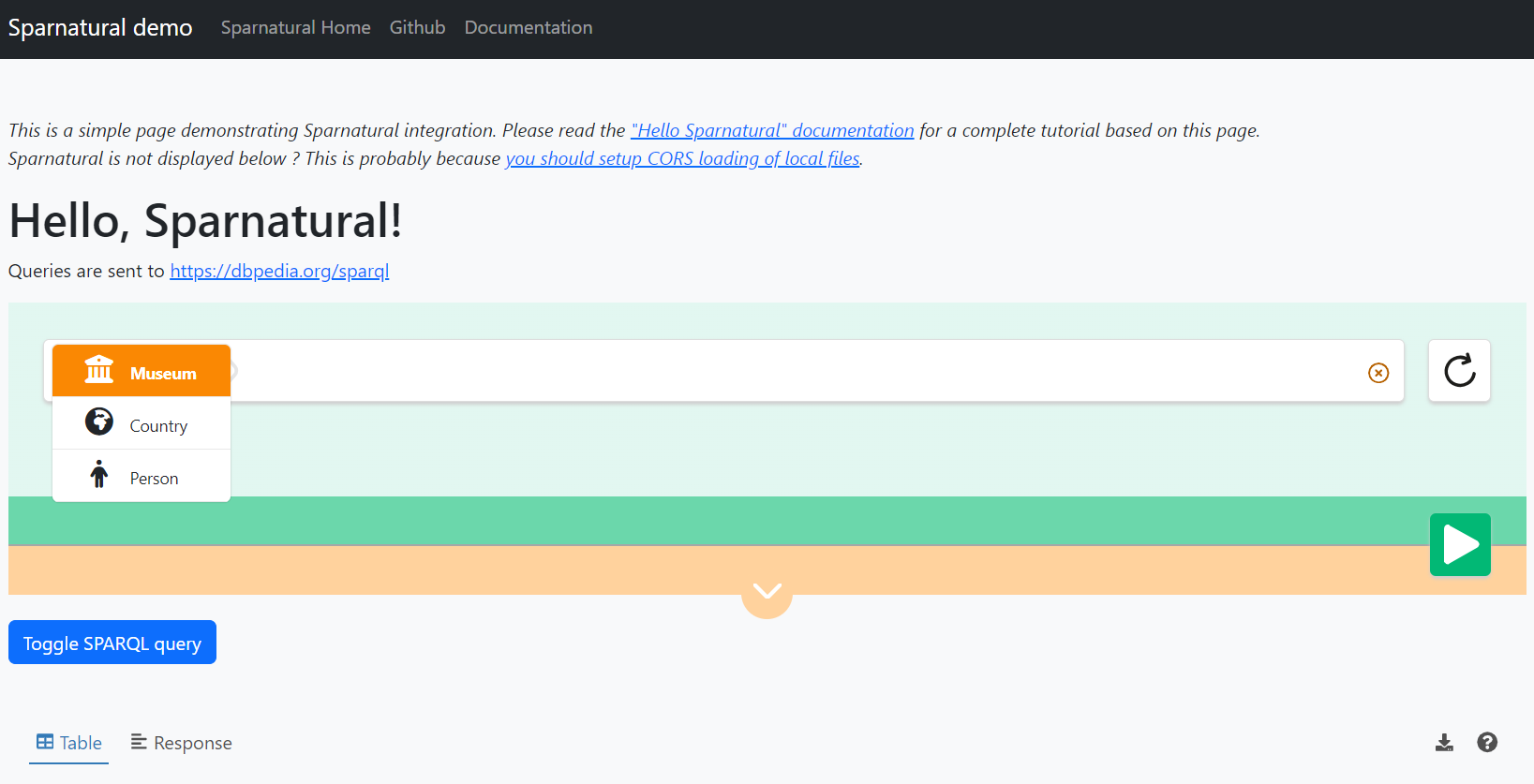
**Chrome, Chromium or Edge**

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:



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



## 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]](#footnote-4). 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.

**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](http://docs.sparnatural.eu/SPARQL-proxy.html).

**/!\** 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](https://github.com/sparna-git/sparql-proxy), 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"

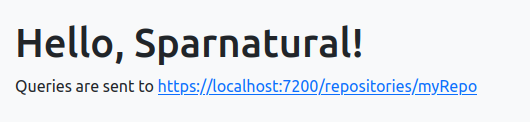
limit="1000"

debug="true"

></spar-natural>

Here you can also adjust the language parameters : the default language is the language in which the knowledge graph is supposed to always have a label for all entities. This is meant to deal with situations where some entities do have a label in the user preferred language, and others don’t, but will have a label in the default language. This is explained in more details in the « how to configure » documentation.

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:



# Create your first Sparnatural configuration

## Setup your configuration using the spreadsheet

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 in the configuration file config.xlsx. This spreadsheet is actually a way to encode a SHACL specification of your data model.

## In the provided config.xlsx file, you will find a small configuration example working on DBPedia. We will guide you on how to adapt it with different classes and properties. The followup « how to configure » documentation contains an in-depth documentation of the structure of the Excel table, which we will not describe in details here.Create 2 entities in the « Entities » tab

Start by emptying the example configuration entities on lines 10 and 11.

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.

### Create the « foaf:Person » entity

The foaf:Person entity will be created on line 10.

1. URI

In cell A10 you write the URI of the configuration entity (actually, the URI of a SHACL NodeShape): this column uses the "this" prefix declared in the header of the configuration sheet, and the same local part of the class URI, here « Person »,, which gives the following URI : « this:Person ».

1. Sort order : **sh:order^^xsd:integer**

In cell B10,enter the sort order of the entity in the class dropdown list, here « 1 ».

1. Fontawesome icon : **volipi:iconName**

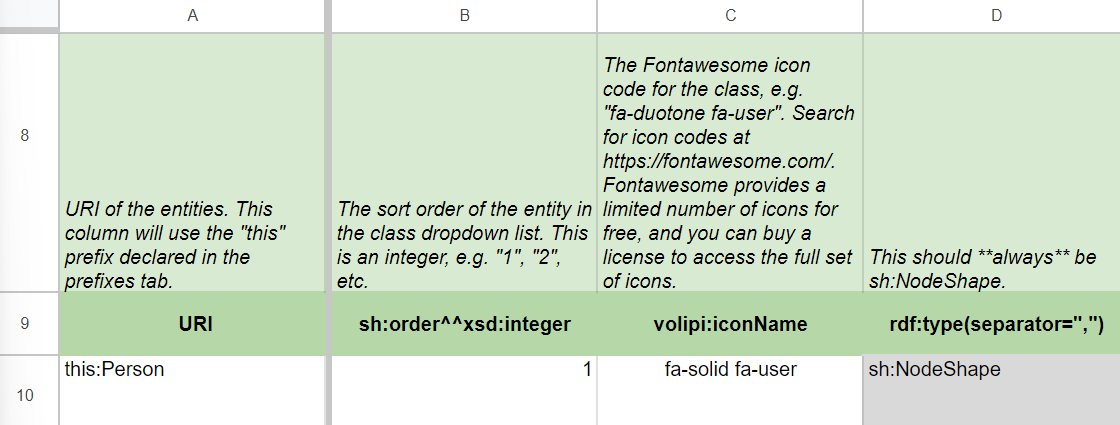
In cell C10, enter the Fontawesome icon code for the entity, here "fa-solid fa-user"

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.

1. Type : **rdf:type(separator=",")**

In cell D10, enter « sh:NodeShape ». The value of the column D needs to always be set to « sh:NodeShape ».

This should give you something like this :



1. Target Class : **sh:targetClass**

In cell E10, write the identifier of the class in the OWL ontology to which the entity in the configuration corresponds : here « foaf:Person ».

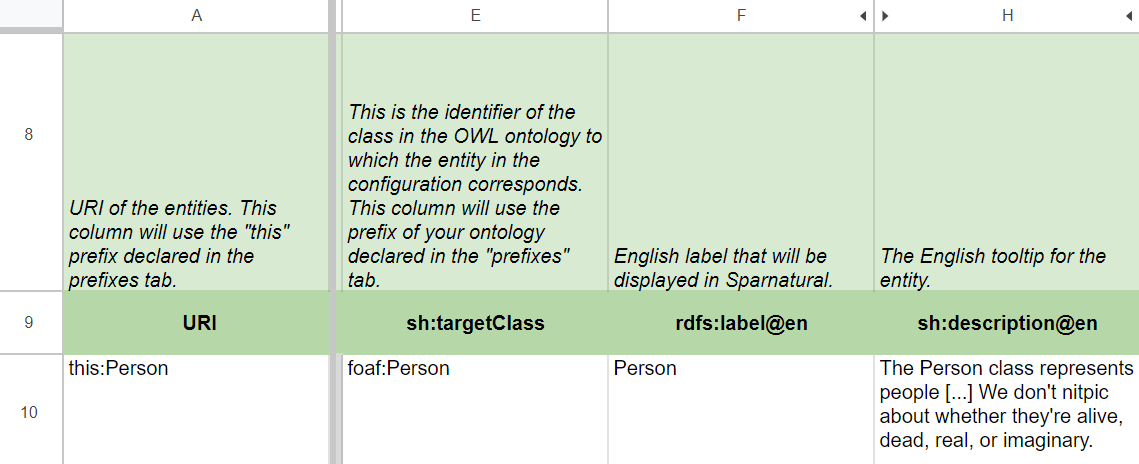
1. Label : **rdfs:label@en**

**In** cell F10, enter the label that will be displayed in Sparnatural for this entity, in english ( you can adjust the language code in the header row line 9 to another language if needed - don’t forget to change the « lang » parameter of the <spar-natural /> HTML element in that case).

1. Tooltip : **sh:description@en**

In cell H10, enter the following tooltip in English (which is the definition of the Person class given in the FOAF ontology) : « The Person class represents people [...] We don't nitpic about whether they're alive, dead, real, or imaginary. »

Now your very first SHACL entity has been created !

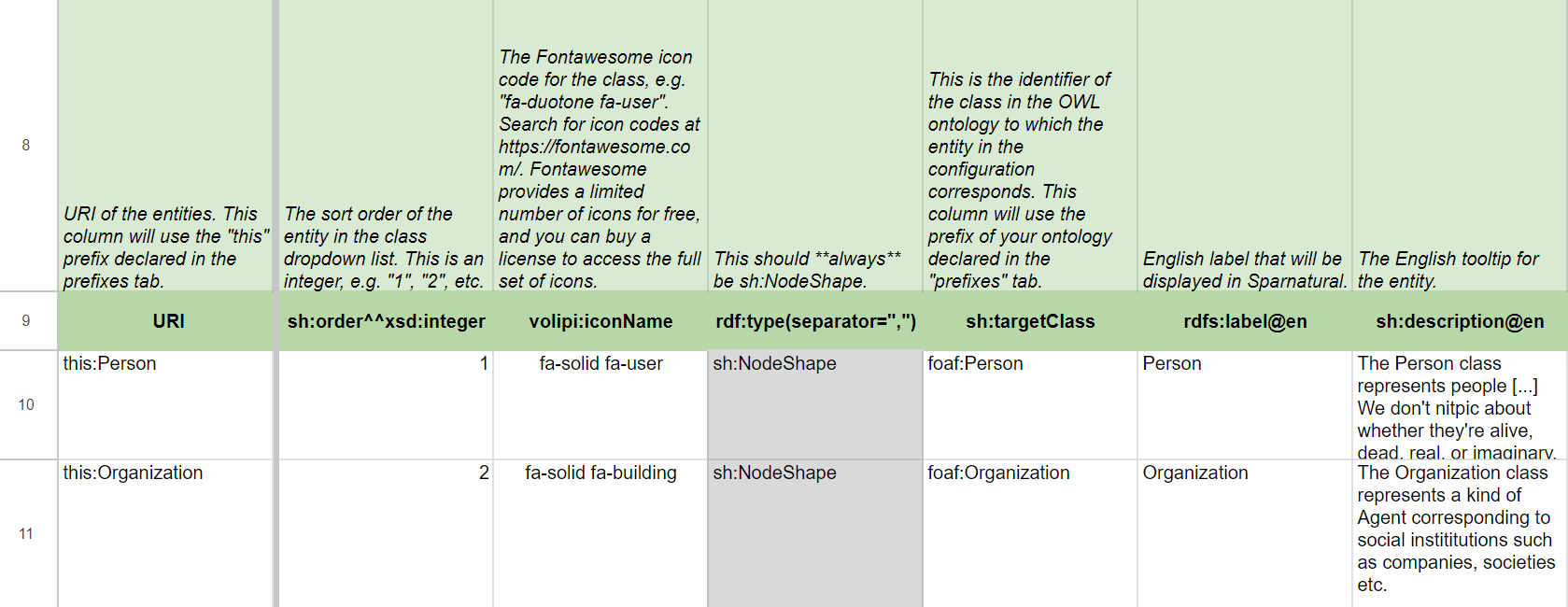


### Create the « foaf:Organization » entity

Repeat the same process to create foaf :Organization line below on line 11, with the following values :

* A11 : write the URI « this:Organization » ;
* B11 : the order « 2 » ;
* C11 : Fontawesome icon code « fa-solid fa-building » ;
* D11 : rdf type « sh:NodeShape » ;
* E11 : target Class « foaf:Organization » ;
* F11 : English label « Organization » ;
* H11 : tooltip « The Organization class represents a kind of Agent corresponding to social institutions such as companies, societies etc. »

This should give you something like this :



All the different options are explained in more details in the « How to configure Sparnatural » manual, chapter « Declaring classes ».

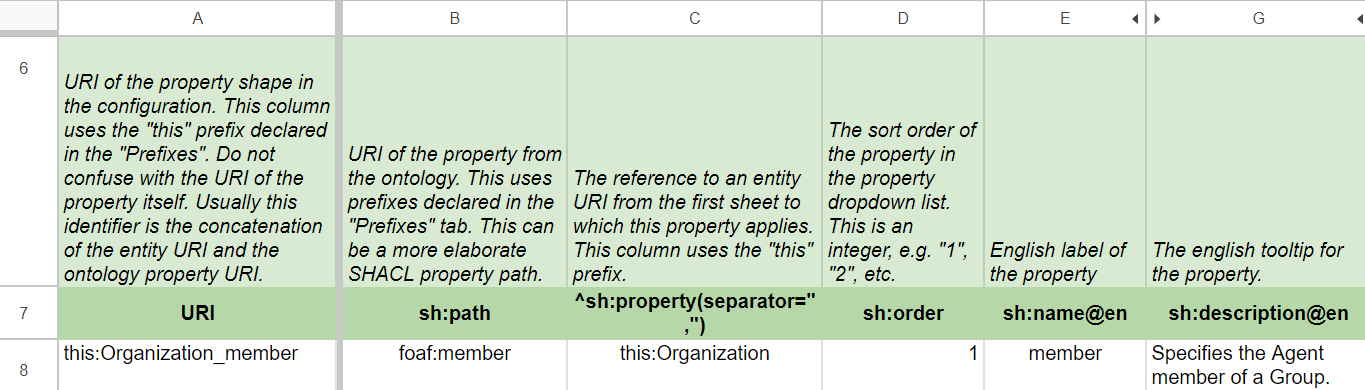
## Create a property in the « Properties » tab

Now let’s add a « foaf:member » property in tab “Properties”. this property will link our 2 classes together, and we will indicate we it as a dropdown list.

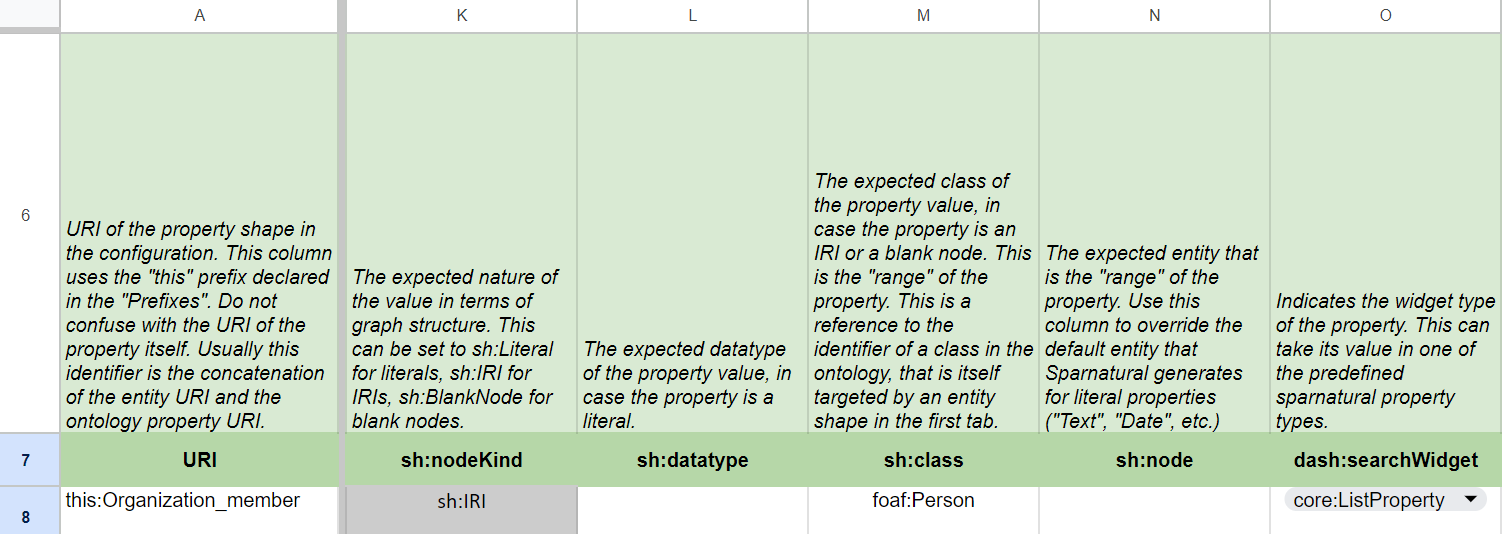
You can copy-paste and edit one existing line from the actual Excel sheetThe logic is quite the same as for the ‘Entities’ tab.

* In cell A8, enter the URI of the constraint : this column uses the "this" prefix. /!\ Do not confuse it with the URI of the property itself. Usually this identifier is the concatenation of the entity URI and the ontology property URI, which in our case gives « this:Organization\_member » ;
* In cell B8, enter the exact URI of the property, here it’s « foaf:member » ;
* In cell C8, the « domain » entity to which the property applies : « this:Organization » (the URI must be one of the entities in the Entities tab) ;
* In cell E8, entre the display label in English, here « member » ;
* In cell G8, the tooltip in English, for example « Specifies the Agent member of this Group » ;

This should give you a line similar to this one :



* In cell K8, we suppose we have to deal with a sh:IRI ;
* In cell M8, the range will be the target Entity of the property, here «foaf:Person » (the URI comes from the corresponding item in the Entities tab) ; Note how this is a reference to the class identifier, and **not** to the entity URI using the « this » prefix.

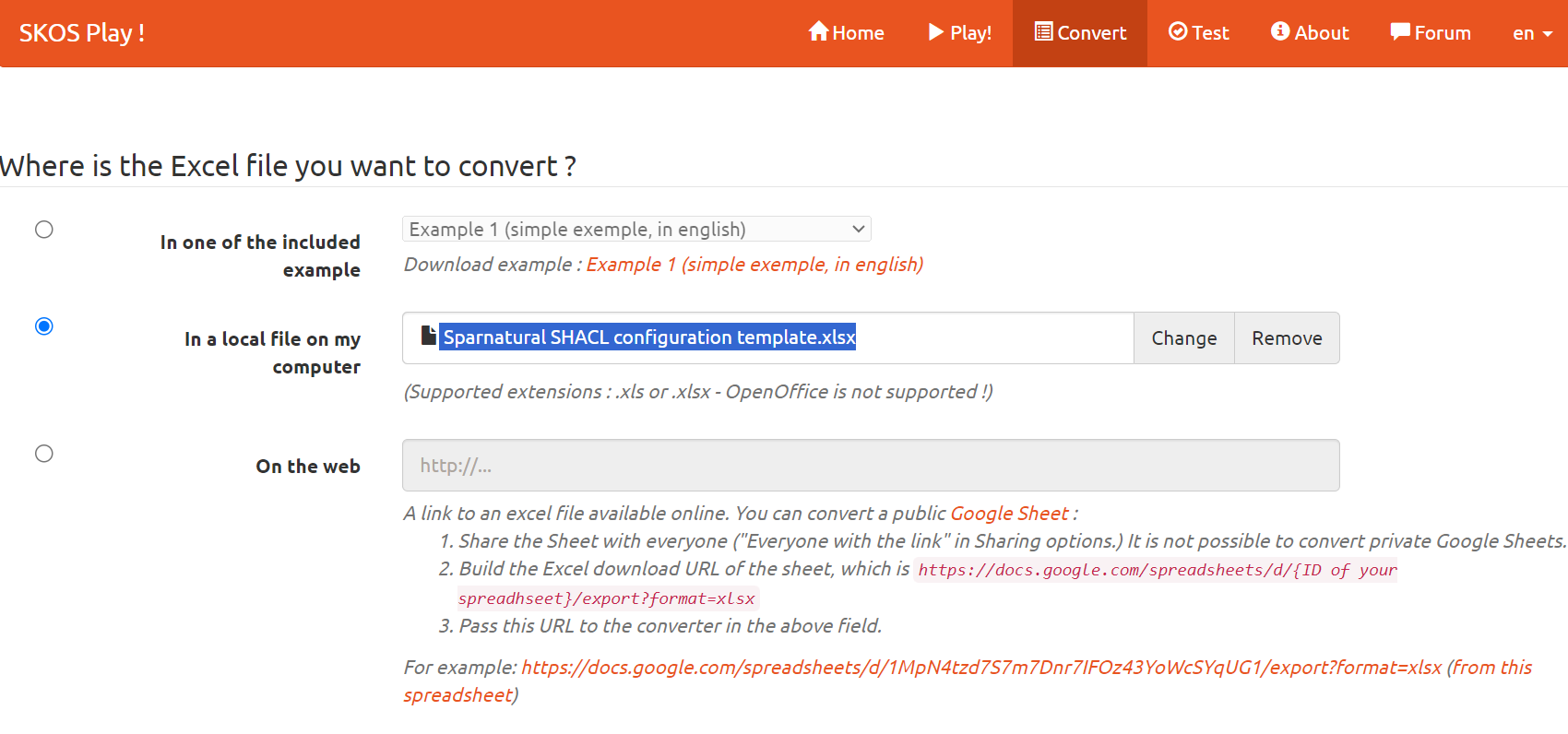


* In cell O8 enter « core:ListProperty » to display your data as a list of items.

For more information about the properties, jump to the the « How to configure Sparnatural » manual, chapter « Declaring properties »

Now you need to convert this configuration file in order to test if it displays correctly into Sparnatural interface :

* + go to SKOS Play website, tab Convert : <https://skos-play.sparna.fr/play/convert?lang=en>
  + submit the Excel file
  + check the “Ignore SKOS post-processings on the data” box
  + click on « Convert » button and save the result in a new « myconfig.ttl » file



## 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"

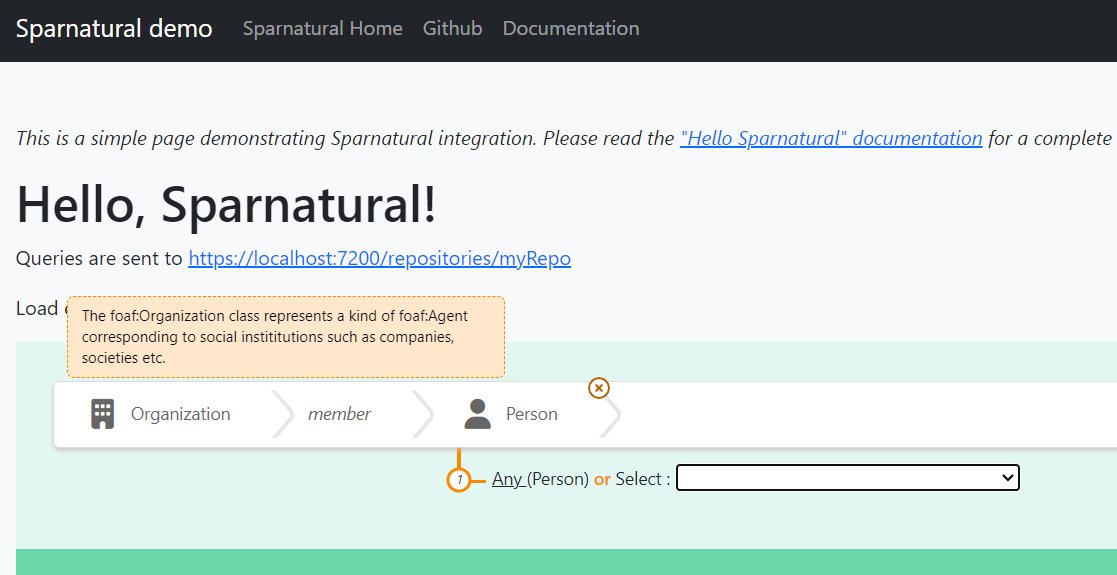
limit="1000"

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:



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

**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 express aggregation queries (like COUNT)

All of these you can learn in the «How-to configure Sparnatural » document. The [Sparnatural documentation website](https://docs.sparnatural.eu) offers comprehensive documentation on all the features. The [Github repository of Sparnatural](https://github.com/sparna-git/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 !

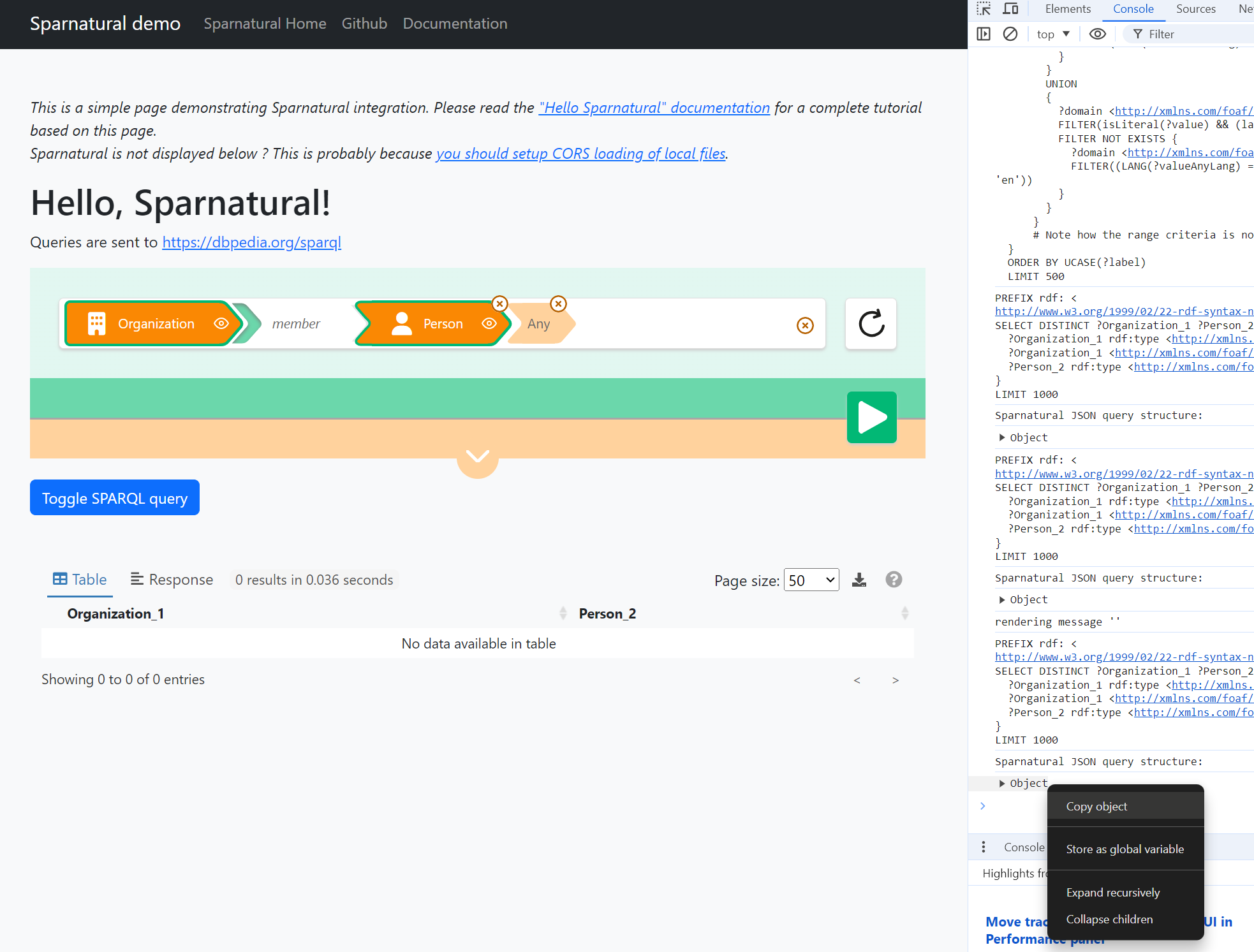
# Annex : adjust the example queries

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.

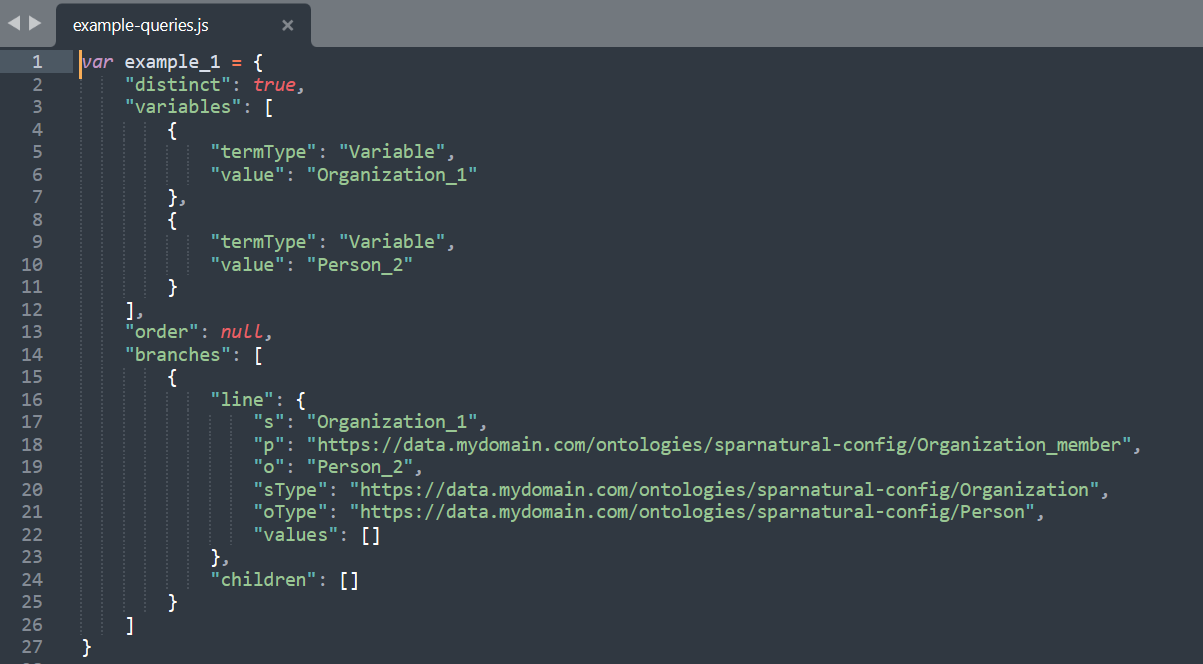
Start creating a query via the in-place query builder and open your browser console at the same time. Here you can see that each time you add a new parameter to your query, a new « Sparnatural JSON query structure » message appears.

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



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

Edit the example-queries.js javascript file :



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.Add more variables if needed.
3. 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>

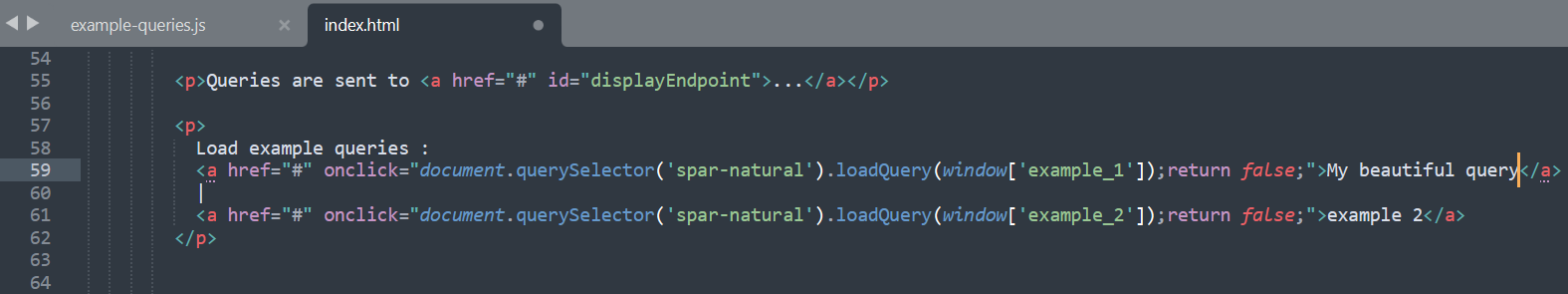
|

<a href="#" onclick="document.querySelector('spar-natural').loadQuery(window['example\_2']);return false;">example 2</a>

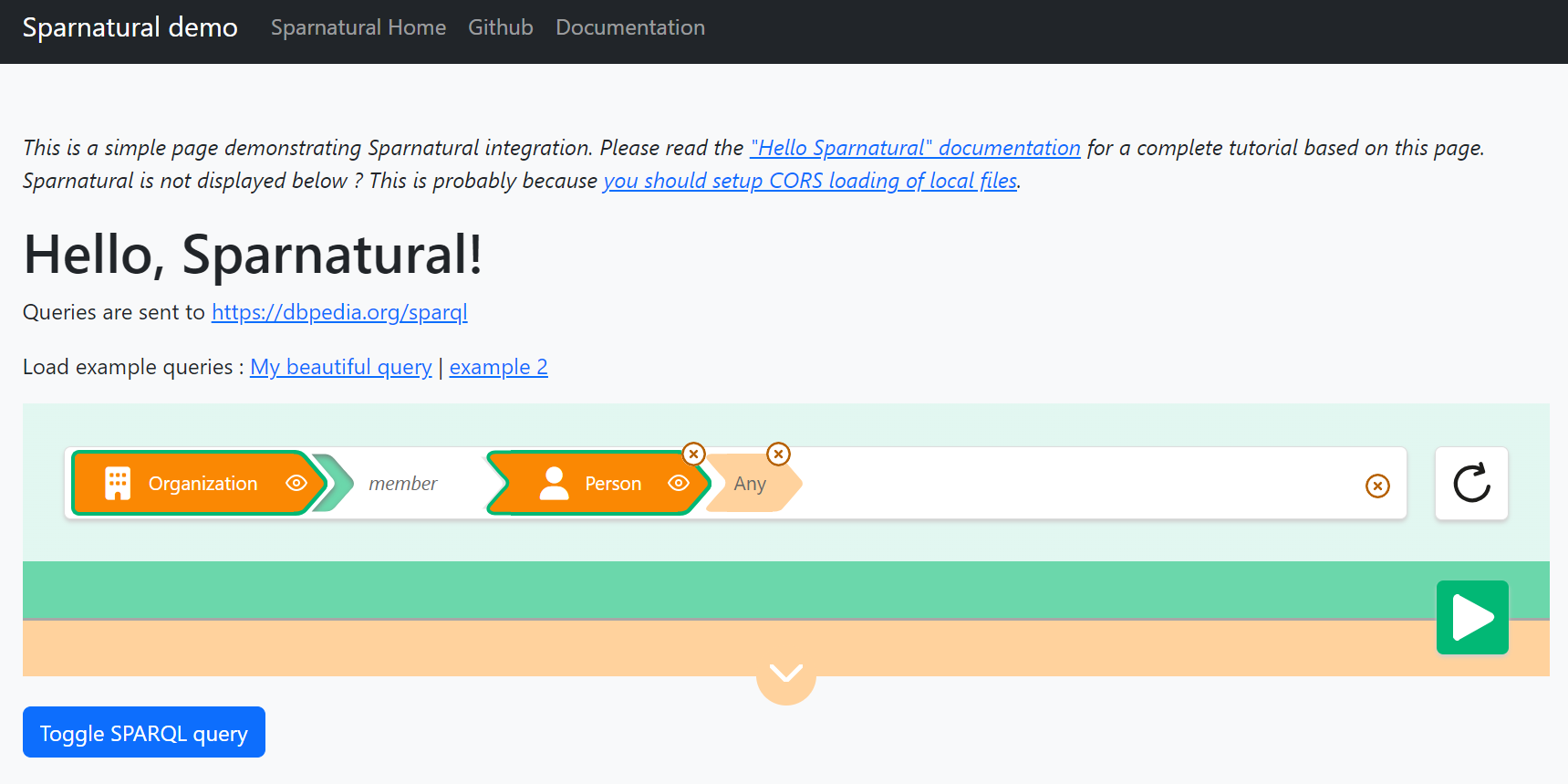
</p>

-->

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



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



1. This plugin is part of another project at <https://github.com/sparna-git/sparnatural-yasgui-plugins> [↑](#footnote-ref-2)
2. The browser console can be opened with the F12 key [↑](#footnote-ref-3)
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> [↑](#footnote-ref-4)