

“Hello Sparnatural”

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

Version : 2.1

Last modified on : october 2024

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

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

Introduction	2
Objectives	2
Prerequisites	2
Structure of this guide	3
Setup the tutorial page	3
Get the tutorial page	3
Content of the tutorial folder	3
Setup your local working environment	4
Allow loading of local files in your browser	4
Ensure your SPARQL endpoint is CORS-enabled, or use a proxy	6
Why do we need CORS ?	6
Check CORS	6
Allow CORS on your triplestore	6
...or use Sparnatural SPARQL proxy	7
Point the tutorial page to your SPARQL service	7
Create your first Sparnatural configuration	8
Setup your configuration using the spreadsheet	8
Create 2 entities in the « Entities » tab	8
Create the « foaf:Person » entity	8
Create the « foaf:Organization » entity	10
Create a property in the « Properties » tab	10
Convert the spreadsheet in RDF	12

Point the demo page to your config file	13
Test and enjoy	13
Next steps	14
Annex : adjust the example queries	15

Introduction

Objectives

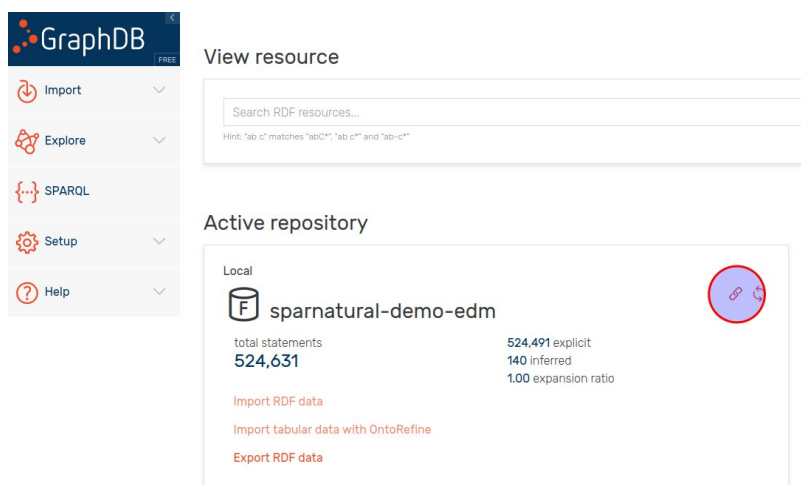
Welcome to this guide on how to setup the “Hello Sparnatural” page ! “Hello Sparnatural” is the tutorial page released with Sparnatural in each of the [releases](#). This documentation will guide you on the necessary steps to work with this tutorial page, and start adapting it to your own knowledge graph. 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 a spreadsheet or online spreadsheet editor (Excel or Google Sheets for example)
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 viewer¹.

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 that you will edit and convert to produce a new version of the « config.ttl » file
- **example-queries.js**: sample queries that you can change after your configuration has been adapted.

¹ 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, or you may not want to display the raw SPARQL query to the user.

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.

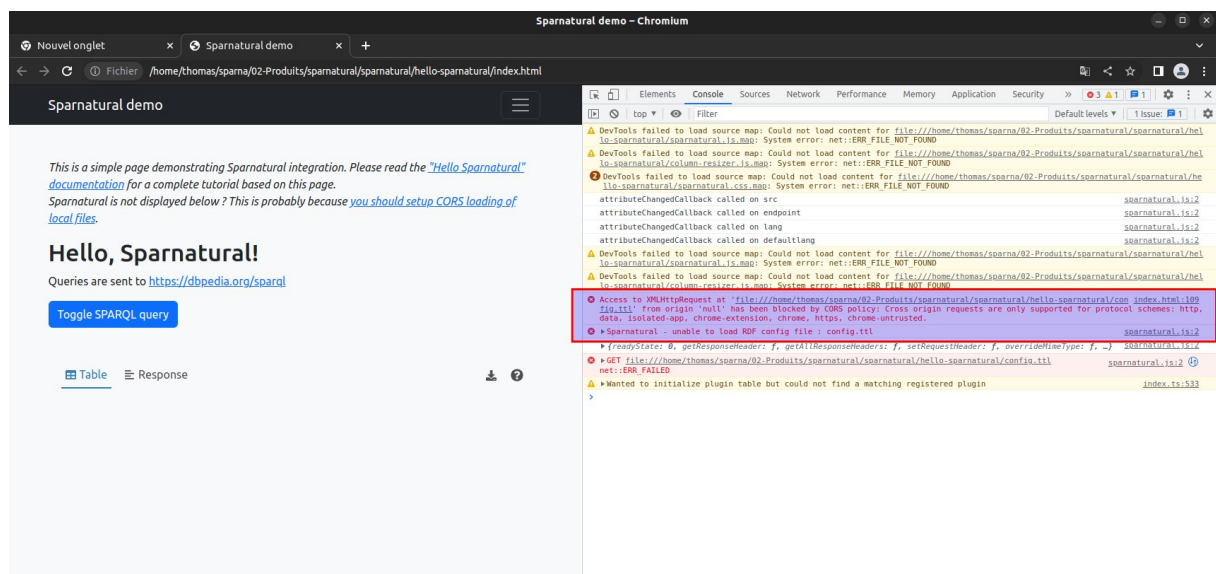
Files you will not have to edit:

- **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 plugin²
- **fa** subfolder: folder containing the [Fontawesome](#) free icon set.
- **config-hello-sparnatural-foaf.xlsx** : a copy of the « config.xlsx » edited to match the content of this tutorial, so that you can compare with what you do

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 an error in your browser console³:



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.

² This plugin is part of another project at <https://github.com/sparna-git/sparnatural-yasgui-plugins>

³ The browser console can be opened with the F12 key

/!\ 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

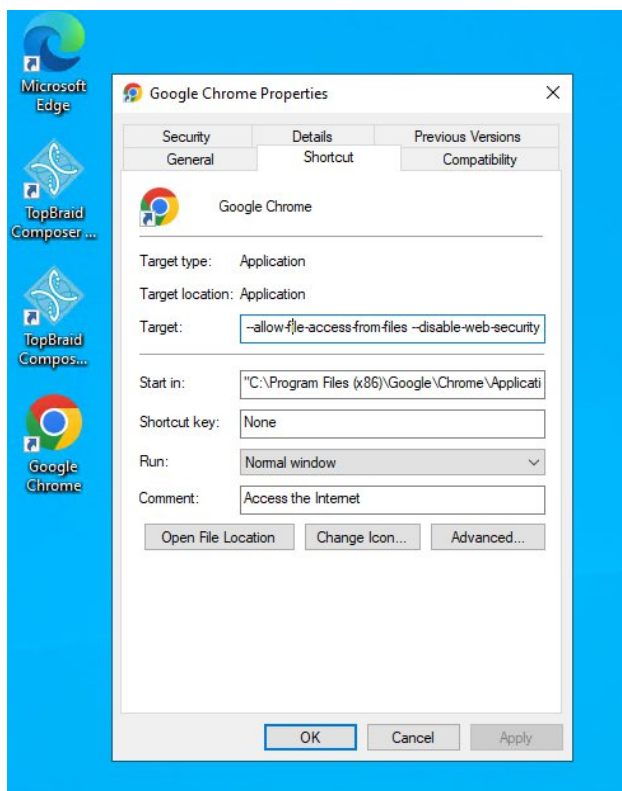
To make Firefox CORS-enabled for local files:

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

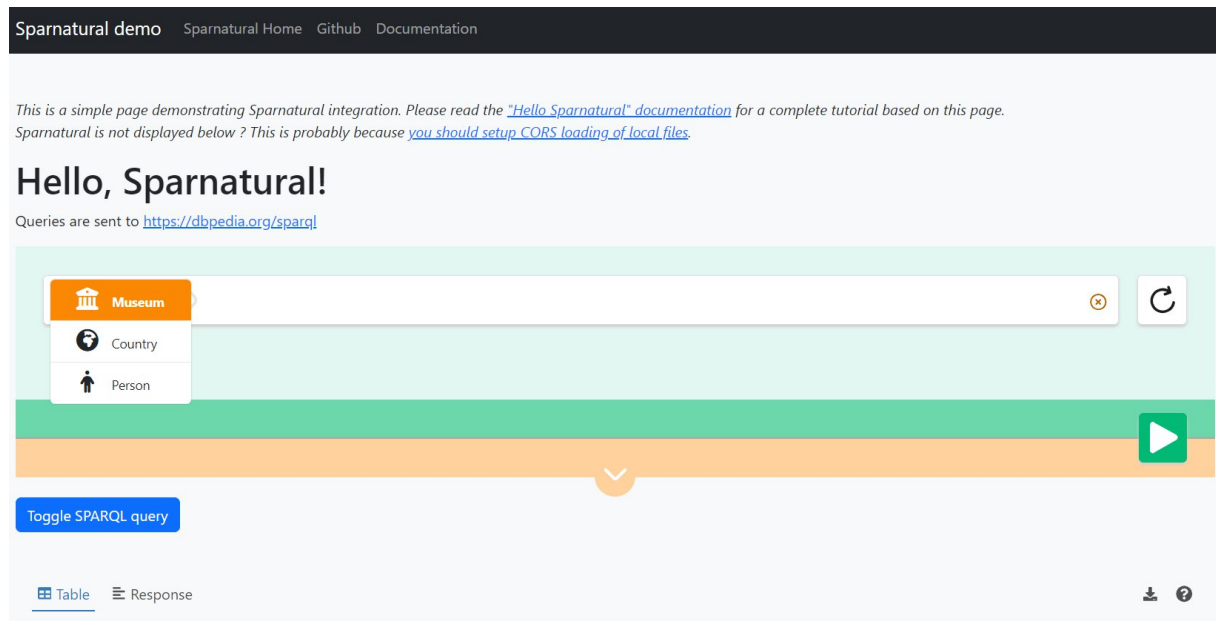
Chrome, Chromium or Edge

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 loaded⁴. 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.

⁴ 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

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](#).

/!\ 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>
```

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:



Hello, Sparnatural!

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

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, which Sparnatural will use as an input.

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

In the « Entities » tab, leave the rows from 1 to 9 unchanged. Start by emptying the example configuration entities line 10 and onwards (from 10 to 14 exactly), so you start with an empty table.

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` ».

2. Sort order : `sh:order^^xsd:integer`

In cell B10, enter the sort order of the entity in the class dropdown list, here « 1 » as we want it to appear first.

3. 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.

4. 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 :

	A	B	C	D
8	<i>URI of the entities. This column will use the "this" prefix declared in the prefixes tab.</i>	<i>The sort order of the entity in the class dropdown list. This is an integer, e.g. "1", "2", etc.</i>	<i>The Fontawesome icon code for the class, e.g. "fa-duotone fa-user". Search for icon codes at https://fontawesome.com/. Fontawesome provides a limited number of icons for free, and you can buy a license to access the full set of icons.</i>	<i>This should **always** be sh:NodeShape.</i>
9	URI	sh:order^^xsd:integer	volipi:iconName	rdf:type(separator=",")
10	this:Person	1	fa-solid fa-user	sh:NodeShape

5. 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 ».

6. 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).

7. 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 !

	A	E	F	H
8	URI of the entities. This column will use the "this" prefix declared in the prefixes tab.	This is the identifier of the class in the OWL ontology to which the entity in the configuration corresponds. This column will use the prefix of your ontology declared in the "prefixes" tab.	English label that will be displayed in Sparnatural.	The English tooltip for the entity.
9	URI	sh:targetClass	rdfs:label@en	sh:description@en
10	this:Person	foaf:Person	Person	The Person class represents people [...] We don't nitpic about whether they're alive, dead, real, or imaginary.

Create the « foaf:Organization » entity

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

1. A11 : write the URI « this:Organization » ;
2. B11 : the order « 2 » ;
3. C11 : Fontawesome icon code « fa-solid fa-building » ;
4. D11 : rdf:type « sh:NodeShape » ;
5. E11 : target Class « foaf:Organization » ;
6. F11 : english label « Organization » ;
7. 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 :

	A	B	C	D	E	F	H
8	URI of the entities. This column will use the "this" prefix declared in the prefixes tab.	The sort order of the entity in the class dropdown list. This is an integer, e.g. "1", "2", etc.	The Fontawesome icon code for the class, e.g. "fa-duotone fa-user". Search for icon codes at https://fontawesome.com/ . Fontawesome provides a limited number of icons for free, and you can buy a license to access the full set of icons.	This should **always** be sh:NodeShape.	This is the identifier of the class in the OWL ontology to which the entity in the configuration corresponds. This column will use the prefix of your ontology declared in the "prefixes" tab.	English label that will be displayed in Sparnatural.	The English tooltip for the entity.
9	URI	sh:order^^xsd:integer	volipi:iconName	rdf:type(separator=",")	sh:targetClass	rdfs:label@en	sh:description@en
10	this:Person	1	fa-solid fa-user	sh:NodeShape	foaf:Person	Person	The Person class represents people [...] We don't nitpic about whether they're alive, dead, real, or imadinary.
11	this:Organization	2	fa-solid fa-building	sh:NodeShape	foaf:Organization	Organization	The Organization class represents a kind of Agent corresponding to social institutions such as companies, societies etc.

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.

Start by emptying all the table from line 8 downwards (from line 8 to line 17 exactly). You can also keep one line as an example or copy-paste it and adapt it.

1. URI of the constraint

In cell A8, enter the URI of the constraint (This is actually the URI of a SHACL property shape) : 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 », as the "member" property will apply to Organizations in our case ;

2. URI of the property

In cell B8, enter the exact URI of the property from the ontology, here it's « foaf:member ».

3. Domain entity

In cell C8, indicate the « domain » entity to which the property applies : « this:Organization » (the URI must be one of the entities in the Entities tab).

4. Display label

In cell E8, enter the display label in English, as it will appear in the UI interface, here « member ».

5. Tooltip

In cell G8, enter the tooltip in English, for example « Specifies the Agent member of this Group ».

This should give you a line similar to this one :

	A	B	C	D	E	G
6	<i>URI of the property shape in the configuration. This column uses the "this" prefix declared in the "Prefixes". Do not confuse with the URI of the property itself. Usually this identifier is the concatenation of the entity URI and the ontology property URI.</i>	<i>URI of the property from the ontology. This uses prefixes declared in the "Prefixes" tab. This can be a more elaborate SHACL property path.</i>	<i>The reference to an entity URI from the first sheet to which this property applies. This column uses the "this" prefix.</i>	<i>The sort order of the property in the property dropdown list. This is an integer, e.g. "1", "2", etc.</i>	<i>English label of the property</i>	<i>The english tooltip for the property.</i>
7	URI	sh:path	^sh:property(separator=";"")	sh:order	sh:name@en	sh:description@en
8	this:Organization_member	foaf:member	this:Organization	1	member	Specifies the Agent member of a Group.

6. Node kind

In cell K8, enter "sh:IRI", which indicates the property points to IRI resources (as opposed to blank nodes or literals).

7. Target entity/class

In cell M8, the range will be the target class of the property, here « foaf:Person ». Note how this is a reference to the class identifier, and **not** to the entity URI using the « this » prefix. It must correspond to one of the value of the “sh:targetClass” column from the “Entities” tab.

	A	K	L	M	N	O
6	URI of the property shape in the configuration. This column uses the "this" prefix declared in the "Prefixes". Do not confuse with the URI of the property itself. Usually this identifier is the concatenation of the entity URI and the ontology property URI.	The expected nature of the value in terms of graph structure. This can be set to sh:Literal for literals, sh:IRI for IRIs, sh:BlankNode for blank nodes.	The expected datatype of the property value, in case the property is a literal.	The expected class of the property value, in case the property is an IRI or a blank node. This is the "range" of the property. This is a reference to the identifier of a class in the ontology, that is itself targeted by an entity shape in the first tab.	The expected entity that is the "range" of the property. Use this column to override the default entity that Sparnatural generates for literal properties ("Text", "Date", etc.)	Indicates the widget type of the property. This can take its value in one of the predefined sparnatural property types.
7	URI	sh:nodeKind	sh:datatype	sh:class	sh:node	dash:searchWidget
8	this:Organization_member	sh:IRI		foaf:Person		core:ListProperty ▼

8. Search widget

In cell O8 enter « core:ListProperty » to display the values as a list of items. Other values are available to control how the user can search for values of this property.

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

Convert the spreadsheet in RDF

Before Sparnatural can read the configuration, it needs to be converted to RDF. More details on this “xls2rdf” converter are given in the “how to configure” documentation, we will simply describe here the most simple way to do it:

1. Go to SKOS Play website, tab Convert : <https://skos-play.sparna.fr/play/convert>
2. Submit the Excel file in the “In a local file on my computer” field
3. Check the “Ignore SKOS post-processings on the data” box at the bottom of the form
4. Click on « Convert » button
5. Save the result in a new « myconfig.ttl » file in the hello-sparnatural folder

Where is the Excel file you want to convert ?

☐


In one of the included example

Example 1 (simple example, in english) ▼

Download example : [Example 1 \(simple example, in english\)](#)

☒

In a local file on my computer

 **Sparnatural SHACL configuration template.xlsx**

Change

Remove

(Supported extensions : .xls or .xlsx - OpenOffice is not supported !)

☐

On the web

A link to an excel file available online. You can convert a public [Google Sheet](#) :

1. Share the Sheet with everyone ("Everyone with the link" in Sharing options.) It is not possible to convert private Google Sheets.
2. Build the Excel download URL of the sheet, which is <https://docs.google.com/spreadsheets/d/{ID of your spreadsheet}/export?format=xlsx>
3. Pass this URL to the converter in the above field.

For example: <https://docs.google.com/spreadsheets/d/1MpN4tzd7S7m7Dnr7IFOz43YoWcSYqUG1/export?format=xlsx> (from this spreadsheet)

Note that, each time you modify the Excel configuration file, you need to convert it again.

Point the 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 institutions such as companies, societies etc.



Organization

> member >



Person



Any_(Person) or Select :

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](#) 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 :

The screenshot shows the Sparnatural demo interface and a browser console. The interface has a header with links: Sparnatural demo, Sparnatural Home, Github, and Documentation. Below the header, there is a text block: "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](\"#\")."

The main content area displays "Hello, Sparnatural!" and "Queries are sent to [https://dbpedia.org/sparql](\"https://dbpedia.org/sparql\")". Below this is a query builder interface with a bar containing "Organization", "member", "Person", and "Any". A "Toggle SPARQL query" button is present. The results section shows "Table", "Response", and "0 results in 0.036 seconds". The table has two columns: "Organization_1" and "Person_2", with the message "No data available in table".

The browser console on the right shows a SPARQL query and its JSON structure. The query is:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?Organization_1 ?Person_2
WHERE {
  ?Organization_1 rdf:type <http://xmlns.com/foaf/0.1/Organization>
  ?Person_2 rdf:type <http://xmlns.com/foaf/0.1/Person>
  ?Organization_1 foaf:member ?Person_2
}
```

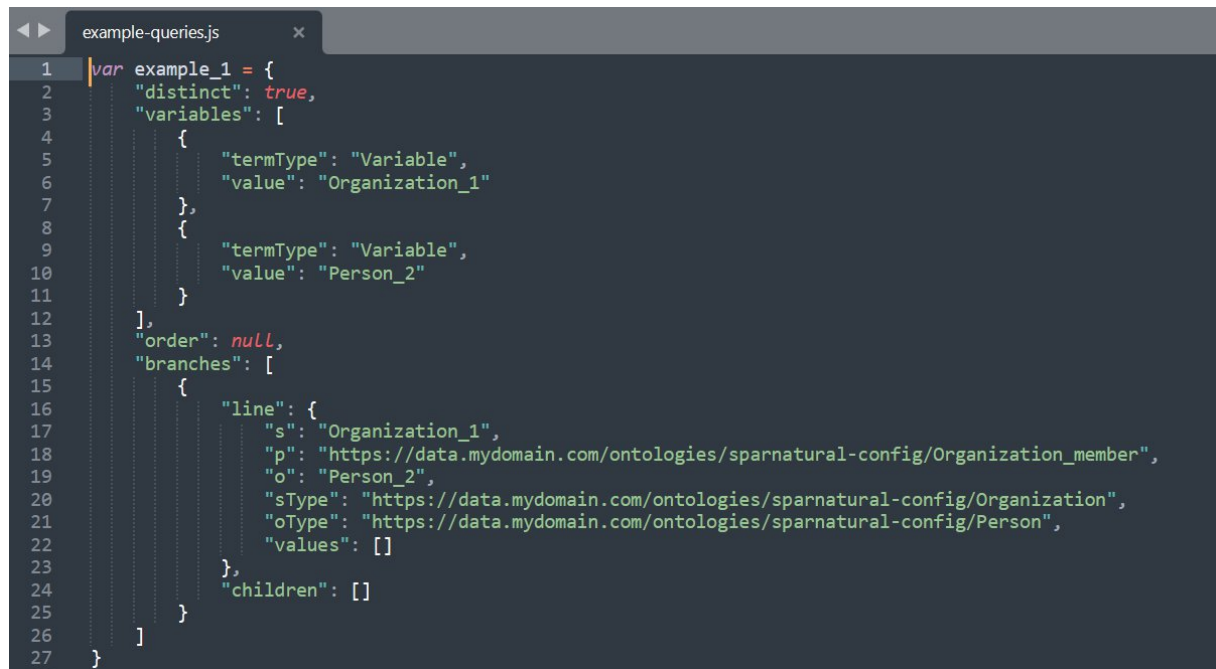
 The JSON structure is:

```
{
  "PREFIX": "rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>",
  "SELECT": "DISTINCT ?Organization_1 ?Person_2",
  "WHERE": [
    {
      "type": "triple",
      "subject": "?Organization_1",
      "predicate": "rdf:type",
      "object": "<http://xmlns.com/foaf/0.1/Organization>"
    },
    {
      "type": "triple",
      "subject": "?Person_2",
      "predicate": "rdf:type",
      "object": "<http://xmlns.com/foaf/0.1/Person>"
    },
    {
      "type": "triple",
      "subject": "?Organization_1",
      "predicate": "foaf:member",
      "object": "?Person_2"
    }
  ]
}
```

 A context menu is open over the JSON structure, showing options: "Copy object", "Store as global variable", "Expand recursively", and "Collapse children".

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

Edit the example-queries.js javascript file :



```
1 var example_1 = {
2   "distinct": true,
3   "variables": [
4     {
5       "termType": "Variable",
6       "value": "Organization_1"
7     },
8     {
9       "termType": "Variable",
10      "value": "Person_2"
11    }
12  ],
13  "order": null,
14  "branches": [
15    {
16      "line": {
17        "s": "Organization_1",
18        "p": "https://data.mydomain.com/ontologies/sparnatural-config/Organization_member",
19        "o": "Person_2",
20        "sType": "https://data.mydomain.com/ontologies/sparnatural-config/Organization",
21        "oType": "https://data.mydomain.com/ontologies/sparnatural-config/Person",
22        "values": []
23      },
24      "children": []
25    }
26  ]
27 }
```

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”) :


```
54
55
56
57
58
59
60
61
62
63
64

<p>Queries are sent to <a href="#" id="displayEndpoint">...</a></p>

<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 !

Sparnatural demo Sparnatural Home Github Documentation

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>

Load example queries : [My beautiful query](#) | [example 2](#)

Organization

member

Person

Any

Toggle SPARQL query