



# Practical Knowledge Graph Example



Protege,  
Stardog and  
Peeps

# Today's exercise

1. Look at a simple ontology for information about people and their relations in Protégé
2. Look at some instance data in Protégé
3. Run the DL and rule reasoner in Protégé
4. Load the ontology and data into Stardog
5. Browse and query the resulting knowledge graph in Stardog

# Preliminaries

- On your own computer (Windows, Mac, Linux)
  - Download and install [Protégé](#)
  - Download, install and configure the community edition of [Stardog 5](#)
  - Clone the 691 [peeps](#) repository

# Peeps files

- The peeps repo has five files
  - **README.md**
  - **catalog-v001.xml** – protégé config file
  - **load\_peeps.sh** – bash script to load peeps into stardog
  - **mypeeps.ttl** – data encoded using peeps ontology
  - **peeps.ttl** – the peeps ontology
  - **prefixes.ttl** – list of prefixes, used by stardog's query component

# Separate ontology and data?

- An ontology is a knowledge graph schema
  - `peeps:Man owl:disjointWith peeps:Woman .`
- We talk about populating it with instance data
  - `:janeDoe a peeps:Woman; foaf:givenName "Jane" .`
- Good practice for real applications is to keep the ontology and data separate
  - i.e., in different files
- Hence, `peeps.ttl` and `mypeeps.ttl`

# Why separate ontology and data?

- It really depends on the usecase
- Some facts are part of an ontology if they're important, unchanging knowledge
- Maybe the ontology is a one-off, and will never be used with any other data
- Maybe you added data while developing the ontology for testing and debugging
- But many ontologies are intended for reuse or to represent datasets that change frequently

# Namespaces

- Promoting reuse also entails giving the ontology and a knowledge graph that uses it with data different namespaces
- Namespace = uri = unique identifier
- Example
  - <http://dbpedia.org/resource/>
  - <http://dbpedia.org/ontology/>
- BTW, lookup prefixes at <http://prefix.cc>
- Ideally, the uris are ones you control and no one else will use

# Namespace best practice

- Ideally, the namespace should resolve to a file containing the ontology or data
  - Maybe not the data if it's big or proprietary
- Enables other ontologies to **import and use** yours just from its URI
- If you don't control a long-lived URI ...
  - You might use a file on github
  - You might use [purl](#) to create a “permanent url” that redirects to the current location

# Peeps.ttl in Protégé

peeps.ttl (https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/peeps.ttl) : [Users/finin/Desкто...

peeps.ttl Search...

Data Properties x Annotation Properties x Individuals by class x DL Query x SWRLTab x

Active Ontology x Entities x Object Properties x

Annotations Selected entailments Rules Ontology prefixes

**Ontology header:**

**Ontology IRI** https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/peeps.ttl

**Ontology Version IRI** e.g. https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/peeps.ttl

Annotations +

rdfs:label

An example ontology for people created in Protege OWL 5.5"

OWL/XML rendering OWL functional syntax rendering

Ontology imports General axioms RDF/XML rendering

**Imported ontologies:**

Direct Imports +

Indirect Imports

**Rules:**

Rules +

hasParent(?p1, ?p2), Woman(?p2) -> hasMother(?p1, ?p2)

hasParent(?p1, ?p2) -> youngerThan(?p1, ?p2)

hasAge(?p1, ?a1), hasAge(?p2, ?a2), lessThan(?a1, ?a2) -> youngerThan(?p1, ?p2)

Git: master

To use the reasoner click Reasoner > Start reasoner  Show Inferences

# Mypeeps.ttl

The screenshot shows a web browser window with the following elements:

- Address Bar:** `mypeeps.ttl (https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl)`
- Navigation:** Back, Forward, and Search buttons.
- Active Ontology:** `mypeeps.ttl (https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl)`
- Menu:** Active Ontology x, Entities x, Individuals by class x, DL Query x
- Annotations:** Annotations, Selected entailments, Rules, Ontology prefixes
- Ontology header:**
  - Ontology IRI:** `https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl`
  - Ontology Version IRI:** e.g. `https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl/1.0.0`
- Annotations:** Annotations +
- Ontology imports:** General axioms, RDF/XML rendering, OWL/XML rendering, OWL functional syntax rendering
- Imported ontologies:** Imported ontologies +
- Direct Imports:**
  - `<https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/peeps.ttl>`
  - `peeps.ttl`
  - Ontology IRI:** `<https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/peeps.ttl>`
  - Location:** <https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/peeps.ttl>
- Indirect Imports:** Indirect Imports
- Footer:** Git: master, To use the reasoner click Reasoner > Start reasoner,  Show Inferences

# When to import an ontology

- In Protégé, we import an ontology if we want a reasoner to understand its vocabulary
- It does not add the ontology to the file that will be saved
- Plus: the knowledge may be important or essential in testing
- Minus: big ontologies may add a lot of useless data
- Here `mypeeps.ttl` imports `peeps`, but not `foaf` or `schema`



# Stardog Graph Platform

- Stardog is easy to install and use, but rich in features
- It has a Web interface, good command-line tools and a Java API
- We'll look at how to
  - Load the peeps example files
  - Browse the results
  - Query the graph via the Web console

# Start Stardog

- This command will start Stardog listening to its default port (5820) and disable security

**stardog-admin server start --disable-security**

- Enter the URL <http://localhost:5820> to access the Web console

Use admin for both the user and password

Stardog Admin Web Console x +

localhost:5820

Stardog Admin Databases Security Query Management admin

### Server

Stardog Home: /Users/finin/stardog Stardog Version: 5.3.5

### Databases

Name	Status
------	--------

### Users

User name	Roles
admin	No roles assigned
anonymous	reader
root	No roles assigned

# Stardog script

- load\_peeps.sh is a bash script for loading the peeps data and ontology
- Use variations for other systems or shells
- Once loaded go to <http://localhost:5820/> to use Stardog's web interface

# Stardog's web interface

Stardog Admin Web Console

localhost:5820

Stardog Admin **Databases** Security Query Management admin

Server

Stardog Home: /Users/finin/stardog Stardog Version: 5.3.5

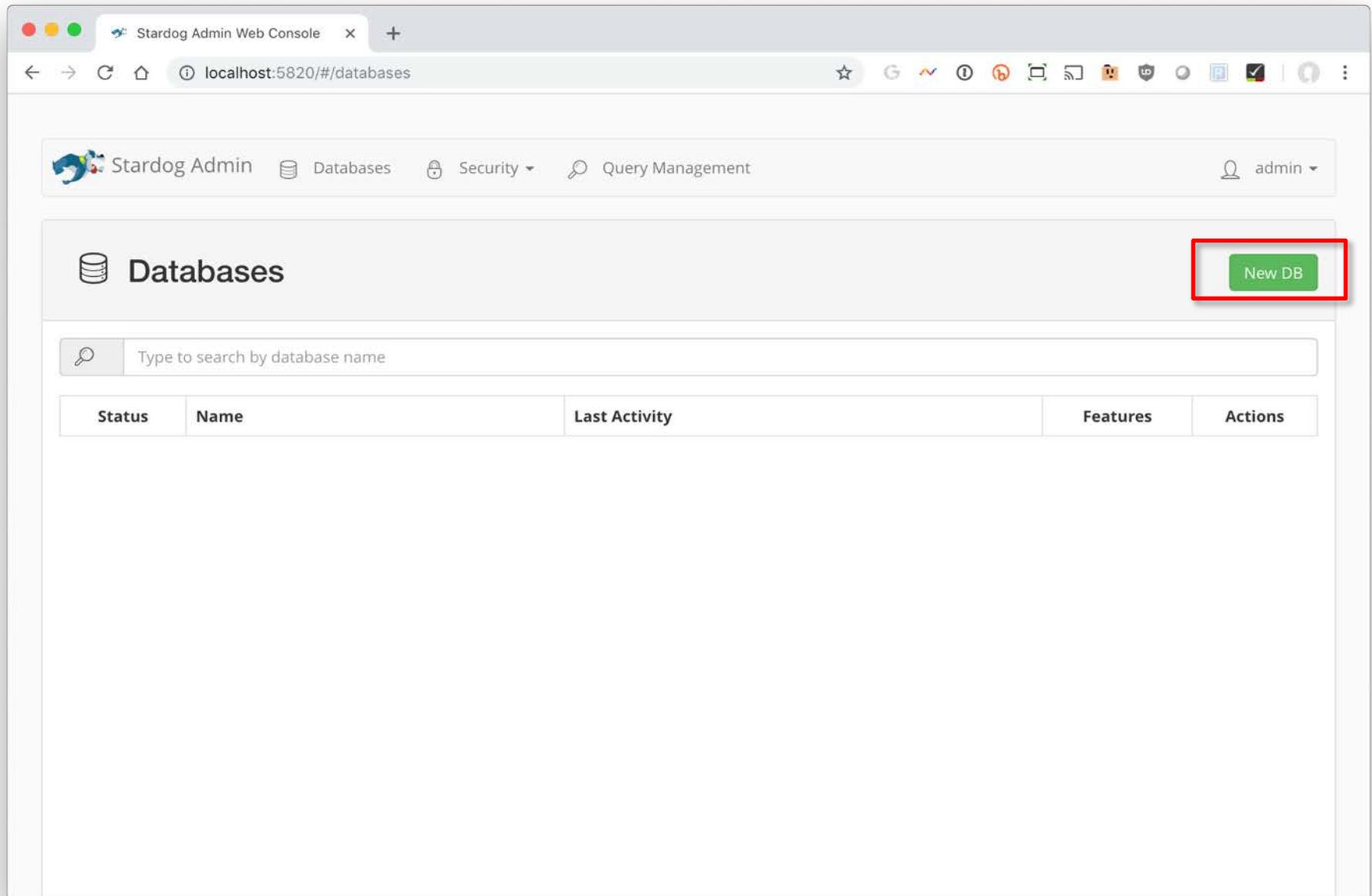
Databases

Name	Status
------	--------

Users

User name	Roles
admin	No roles assigned
anonymous	<b>reader</b>
root	No roles assigned

# Create a database



The screenshot shows the Stardog Admin Web Console interface. The browser address bar indicates the URL is localhost:5820/#/databases. The page header includes the Stardog Admin logo, navigation links for Databases, Security, and Query Management, and a user profile for 'admin'. The main content area is titled 'Databases' and features a search input field with the placeholder text 'Type to search by database name'. Below the search field is a table with the following columns: Status, Name, Last Activity, Features, and Actions. A green 'New DB' button is highlighted with a red border in the top right corner of the 'Databases' section.

Stardog Admin Web Console

localhost:5820/#/databases

Stardog Admin Databases Security Query Management admin

Databases

New DB

Type to search by database name

Status	Name	Last Activity	Features	Actions
--------	------	---------------	----------	---------

# Name it mypeeps and accept the defaults

The screenshot shows the Stardog Admin Web Console interface for creating a new database. The browser address bar shows `localhost:5820/#/databases/new`. The page title is "New database".

This wizard will help you create a new Stardog database. It will go through all the options available for setting up a new DB. All the options are filled up with the default values. If all you need are the default options, just go ahead and click Finish, otherwise click Next.

**Database name**

mypeeps

**Database archetypes**

None selected

**Database online**

ON

**Strict Parsing**

ON

**Preserve BNode identifiers**

ON

**Database namespaces**

- rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
- rdfs=http://www.w3.org/2000/01/rdf-schema#
- xsd=http://www.w3.org/2001/XMLSchema#
- owl=http://www.w3.org/2002/07/owl#
- stardog=tag:stardog:api:

Add namespace

Finish

Next

**Database created!**  
Database **mypeeps** was created, go to **mypeeps console** to add data

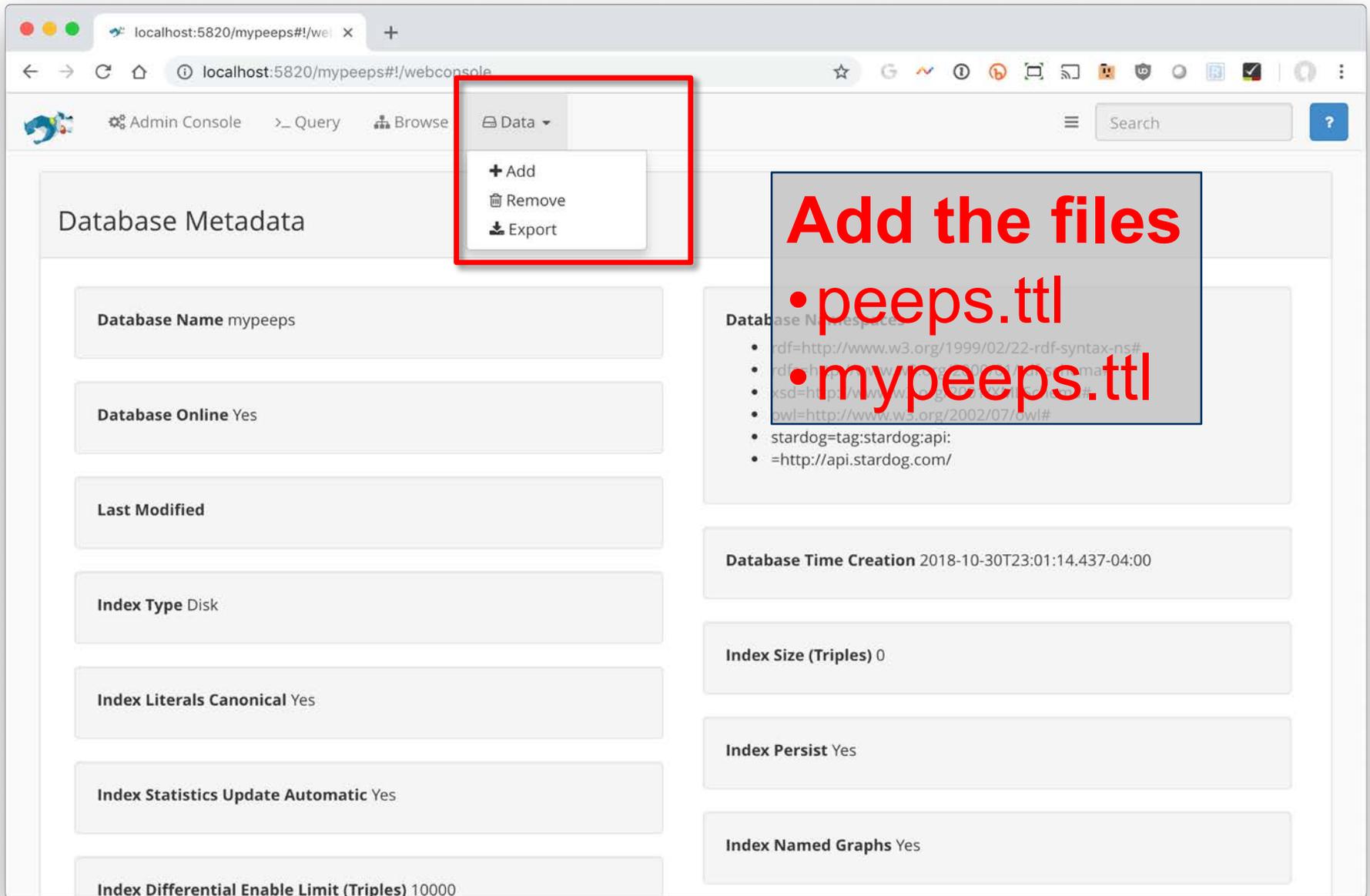
> Query | Browse | Edit | Optimize | Drop | ON

## mypeeps

### Database

Database archetypes	
Database name	mypeeps
Database namespaces	rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# rdfs=http://www.w3.org/2000/01/rdf-schema# xsd=http://www.w3.org/2001/XMLSchema# owl=http://www.w3.org/2002/07/owl# stardog=tag:stardog:api: =http://api.stardog.com/
Database creation time	Tuesday, October 30th 2018, 10:48:07 pm -04:00
database modification time	Tuesday, October 30th 2018, 10:48:08 pm -04:00

# Click on *data* and select *+Add*

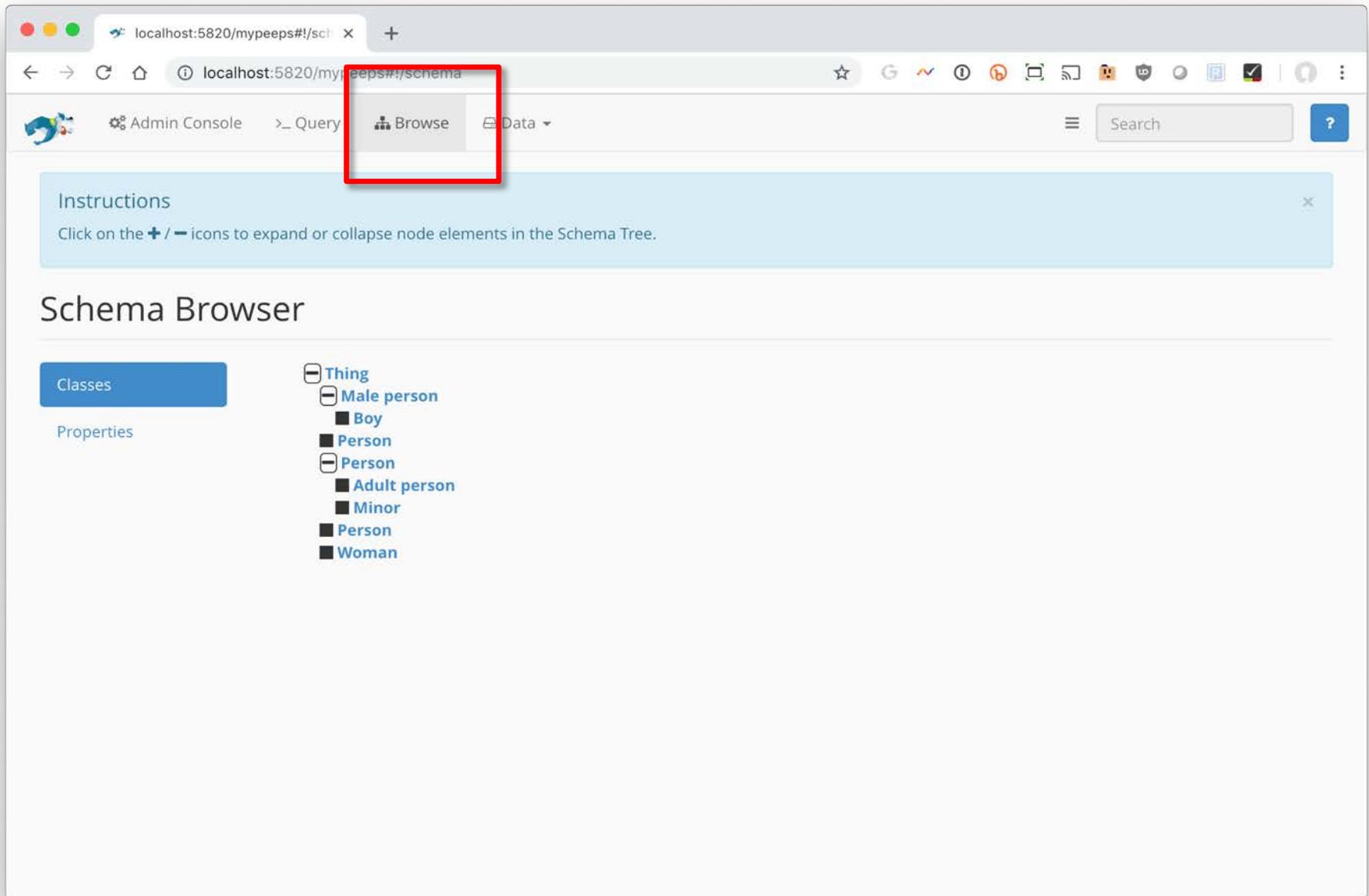


The screenshot shows a web browser window with the URL `localhost:5820/mypeeps#!/webconsole`. The interface includes a navigation bar with 'Admin Console', 'Query', 'Browse', and 'Data' (highlighted with a red box). A dropdown menu for 'Data' is open, showing options: '+ Add', 'Remove', and 'Export'. The main content area is titled 'Database Metadata' and displays various database properties. A blue box highlights a list of files to be added:

- `peeps.ttl`
- `mypeeps.ttl`

Other visible metadata includes: Database Name `mypeeps`, Database Online `Yes`, Last Modified, Index Type `Disk`, Index Literals Canonical `Yes`, Index Statistics Update Automatic `Yes`, Index Differential Enable Limit (Triples) `10000`, Database Time Creation `2018-10-30T23:01:14.437-04:00`, Index Size (Triples) `0`, Index Persist `Yes`, and Index Named Graphs `Yes`.

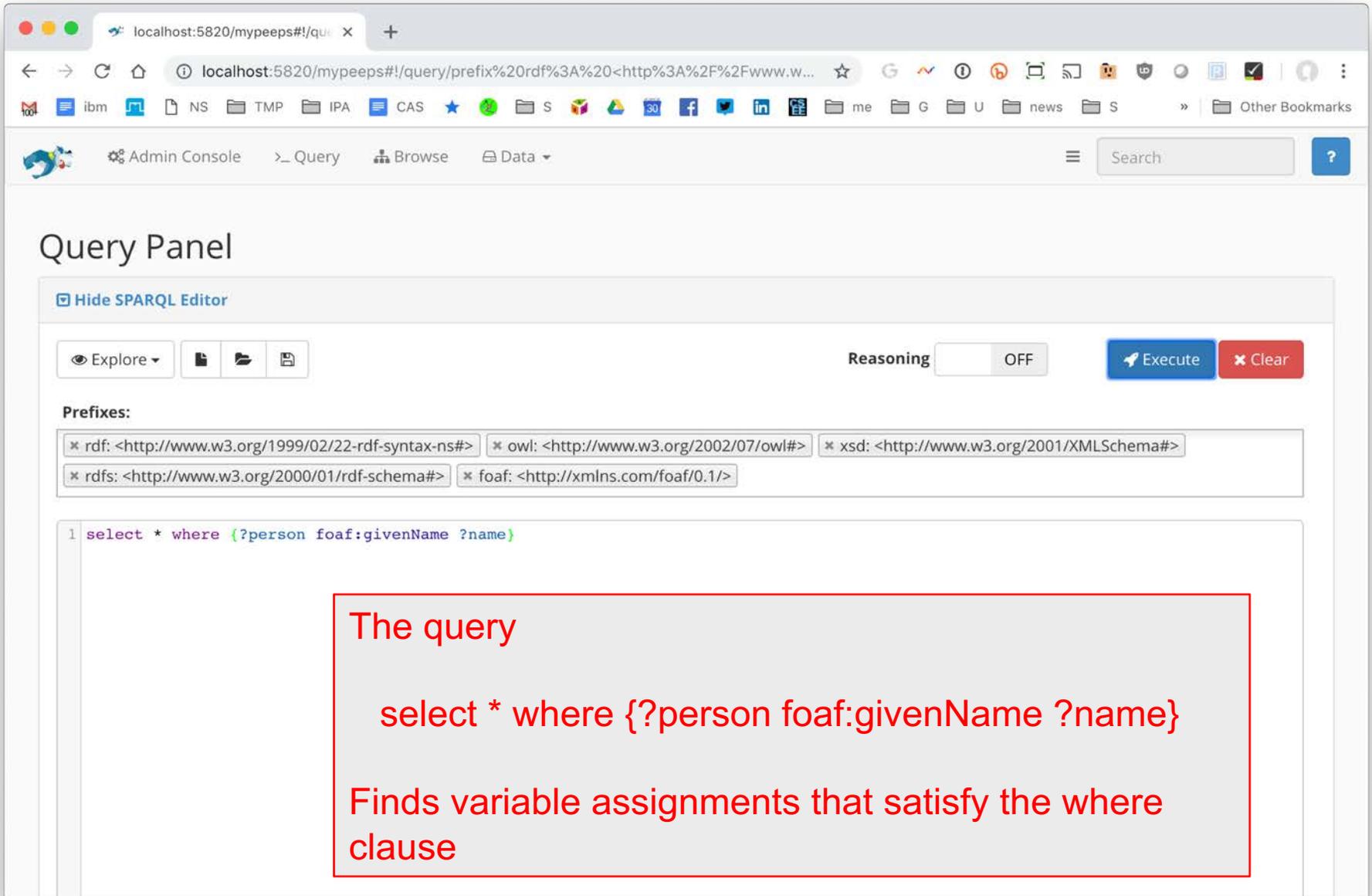
# Go to Browse to explore the graph



The screenshot shows a web browser window with the URL `localhost:5820/mypeeps#!/schema`. The browser's navigation bar includes a search box and a menu with options: `Admin Console`, `>_ Query`, `Browse` (highlighted with a red box), and `Data`. Below the navigation bar, there is a light blue instruction box that reads: "Instructions: Click on the + / - icons to expand or collapse node elements in the Schema Tree." The main content area is titled "Schema Browser" and features a sidebar with "Classes" and "Properties" sections. The "Classes" section is active, displaying a tree view of the schema:

- [-] Thing
  - [-] Male person
    - Boy
    - Person
  - [-] Person
    - Adult person
    - Minor
    - Person
    - Woman

# Go to Query to enter a SPARQL query



The screenshot shows a web browser window with the URL `localhost:5820/mypeeps#!/query/prefix%20rdf%3A%20<http%3A%2F%2Fwww.w...`. The page title is "Query Panel". Below the title, there is a "Hide SPARQL Editor" button. The interface includes a navigation menu with "Admin Console", ">\_ Query", "Browse", and "Data". A search bar is located in the top right. The main content area features a toolbar with an "Explore" dropdown, file management icons, and a "Reasoning" toggle set to "OFF". There are "Execute" and "Clear" buttons. Below the toolbar, a "Prefixes:" section contains several input fields for URIs: `* rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>`, `* owl: <http://www.w3.org/2002/07/owl#>`, `* xsd: <http://www.w3.org/2001/XMLSchema#>`, `* rdfs: <http://www.w3.org/2000/01/rdf-schema#>`, and `* foaf: <http://xmlns.com/foaf/0.1/>`. The SPARQL query editor contains the following query:

```
1 select * where {?person foaf:givenName ?name}
```

The query is highlighted in red in the original image. Below the query editor, a red-bordered box contains the following text:

The query

```
select * where {?person foaf:givenName ?name}
```

Finds variable assignments that satisfy the where clause

# Go to Query to enter a SPARQL query

localhost:5820/mypeeps#!/query/prefix%20rdf%3A%20<http%3A%2F%2Fwww.w...

Admin Console > Query Browse Data

SPARQL Results (returned in 18 ms)

person	name
<a href="https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#alan">https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#alan</a>	Alan
<a href="https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#bob">https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#bob</a>	Robert
<a href="https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#carol">https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#carol</a>	Carol
<a href="https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#diana">https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl#diana</a>	Diana

SPARQL Results RDF/XML  
SPARQL Results JSON  
TSV  
CSV

Export as...

Page 1

It found four solutions. The data can be exported to your computer as a file in any of several formats (e.g., rdf, json, csv, tsv)

localhost:5820/mypeeps#!/query/prefix%20rdf%3A%20<http%3A%2F%2Fwww.w...

Admin Console >\_ Query Browse Data Search ?

Error!  
Unknown prefix: peeps

### Query Panel

Hide SPARQL Editor

Explore Reasoning OFF Execute Clear

Prefixes:

- \* rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- \* owl: <http://www.w3.org/2002/07/owl#>
- \* xsd: <http://www.w3.org/2001/XMLSchema#>
- \* rdfs: <http://www.w3.org/2000/01/rdf-schema#>
- \* foaf: <http://xmlns.com/foaf/0.1/>

```
1 select * where {?person a peeps:Man}
```

The query systems needs to know (independently) about any namespace prefixes you want to use (other than the common ones). Enter these when you create the database.

# Command line commands

Running a simple bash [script](#) will create or refresh the peeps knowledge graph example

```
#!/bin/bash
# loads peeps.ttl, mypeeps.ttl and associated namespaces into a Stardog database.
PORT="5820"
SERVER="http://localhost:$PORT"
DBNAME="mypeeps"
DBURL="$SERVER/$DBNAME"
# stop server in case one is already running
stardog-admin --server $SERVER server stop
# start server
stardog-admin server start --port $PORT --disable-security
# drop database $DBNAME in case it exists already
stardog-admin --server $SERVER db drop -n $DBNAME
# create database $DBNAME with reasoning and search enabled
stardog-admin --server $SERVER db create -o reasoning.sameas=FULL -o search.enabled=true -n $DBNAME
# load ontology and data
stardog data add $DBURL peeps.ttl mypeeps.ttl
# add namespace prefixes for the query system to use
stardog namespace import --verbose $DBURL prefixes.ttl
```

# Query from Python

- Stardog serves as a endpoint for SPARQL queries
- Use this URL to send queries to the mypeeps database  
<http://localhost:5820/mypeeps/query/>
- There are packages that help do this in many languages, including Python
- See [query.py](#) in the peeps repository