



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 latest edition of [Stardog](#)
 - Clone the 491/691 [peeps](#) repository

Peeps files

- The peeps repo has five files
 - **README.md**
 - **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 use case
- 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

- Promote reuse by giving the ontology and data graphs using it 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 is 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
 - Or use the free [purl](#) service to create a “permanent url” redirecting to a current location, e.g., on GitHub

Peeps.ttl in Protégé

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

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/pe`

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

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) ->`

`voungerThan(?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.
- Page Header:** `mypeeps.ttl (https://raw.githubusercontent.com/UMBC-CMSC-491-691-F18-Knowledge-Graphs/peeps/master/mypeeps.ttl)`
- Navigation Tabs:** Active Ontology, Entities, Individuals by class, DL Query.
- 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:** A section with a plus sign icon, currently empty.
- Ontology imports:** General axioms, RDF/XML rendering, OWL/XML rendering, OWL functional syntax rendering.
- 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:** (Empty)
- Footer:** Git: master, To use the reasoner click Reasoner > Start reasoner, Show Inferences

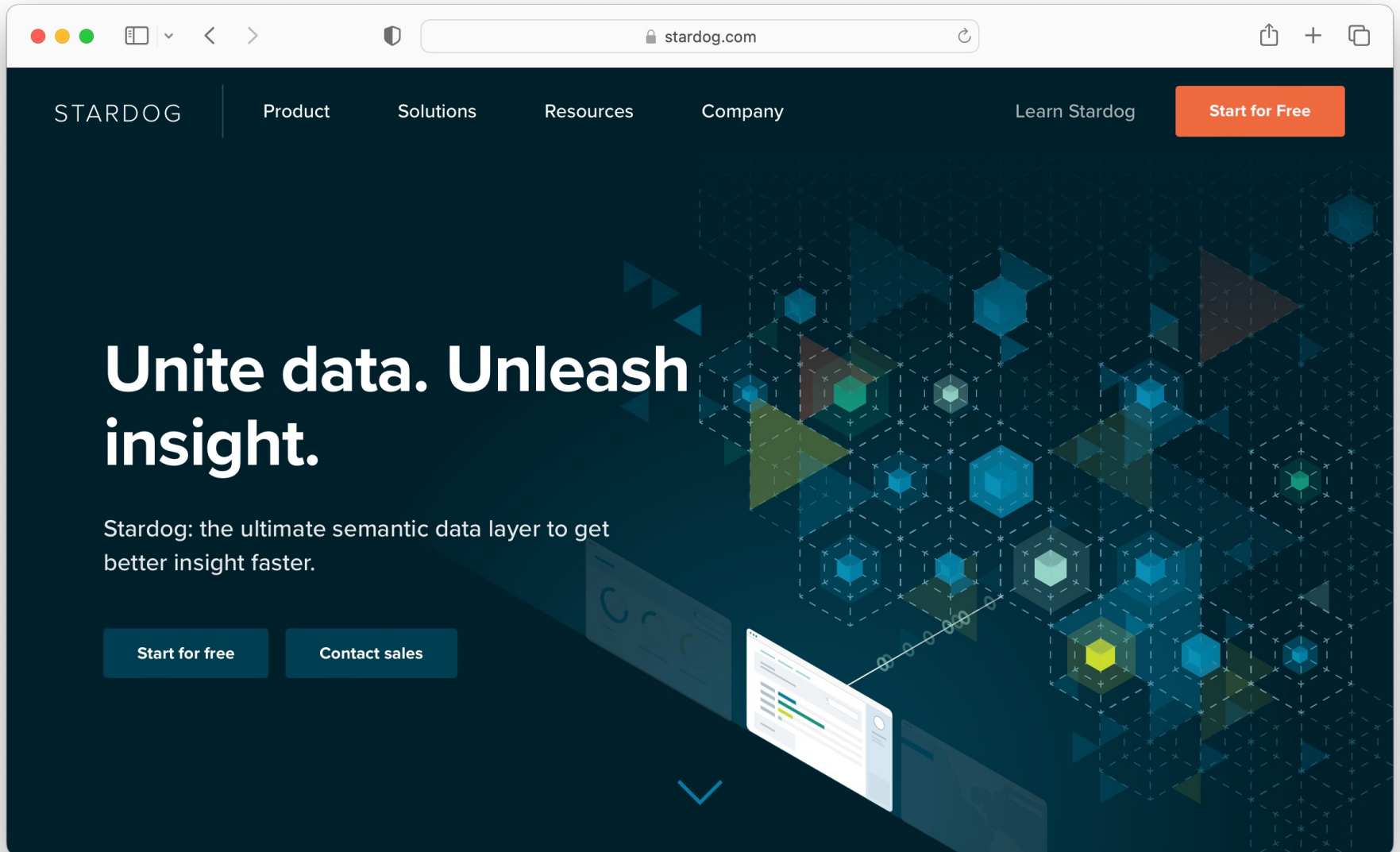
When to import an ontology

- 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 separate web-based interface ([Stardog Studio](#)), command-line tools, a Java API, and can be queried from Python or any language
- We'll look at how to
 - Load the peeps example files
 - Browse the results
 - Query the graph via the Web console

Stardog Knowledge Graph Platform

The image shows a browser window displaying the Stardog website. The browser's address bar shows 'stardog.com'. The website has a dark blue background with a complex geometric pattern of hexagons and triangles in various shades of blue and green. The main heading is 'Unite data. Unleash insight.' in large white text. Below it, a sub-heading reads 'Stardog: the ultimate semantic data layer to get better insight faster.' There are two buttons: 'Start for free' in orange and 'Contact sales' in dark blue. The navigation menu at the top includes 'STARDOG', 'Product', 'Solutions', 'Resources', 'Company', 'Learn Stardog', and 'Start for Free'.

<https://stardog.com/>

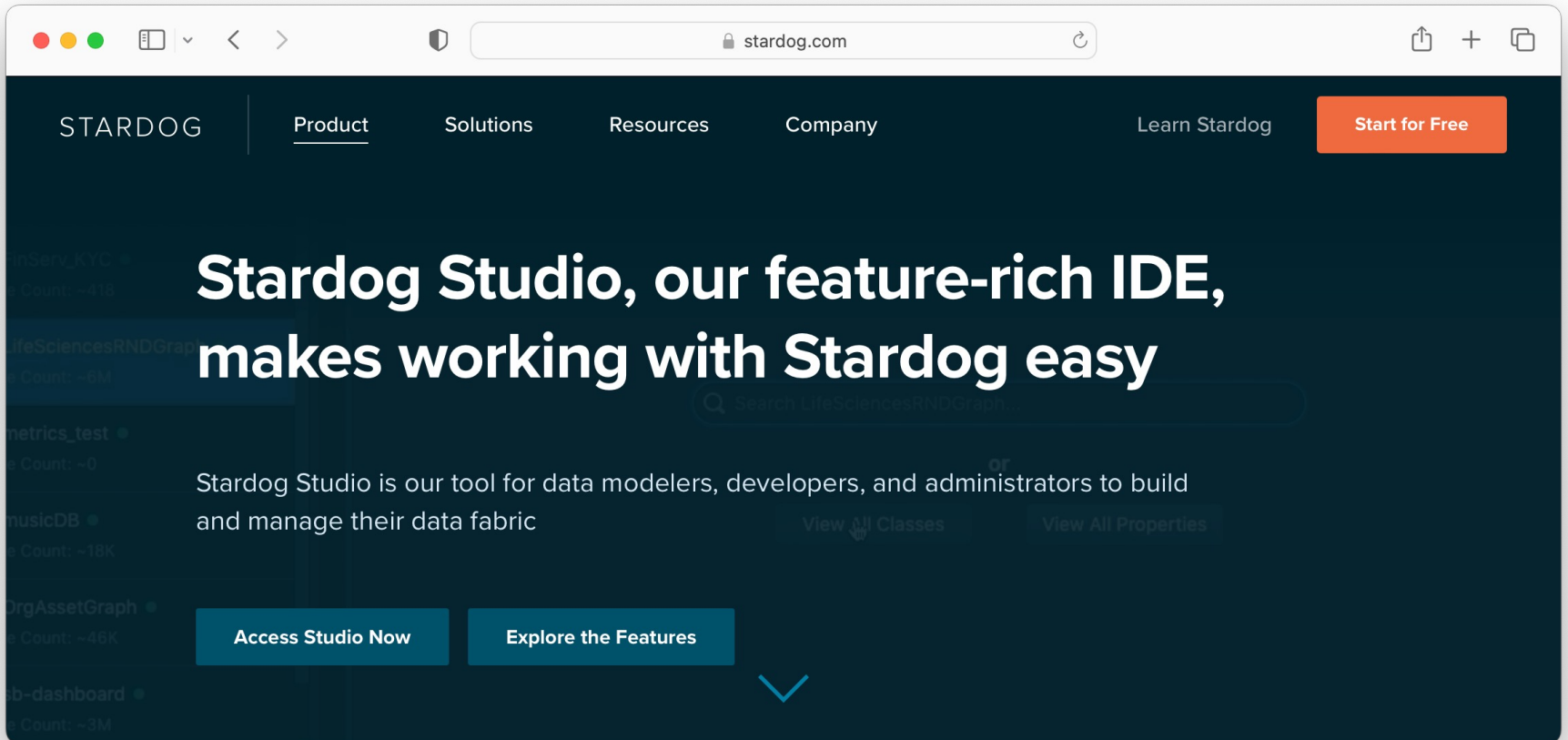
Run in Cloud or Download?

The screenshot shows the Stardog website interface. At the top, there is a navigation bar with the following links: STARDOG, Product, Solutions, Resources, and Company. On the right side of the navigation bar, there are two buttons: "Learn Stardog" and "Start for Free".

The main content area features a dark blue background with a geometric pattern of triangles and lines. The heading "Get Started with Stardog" is prominently displayed in white. Below the heading, the text reads: "It's easy to get started using Stardog's leading Enterprise Knowledge Graph platform. You can create an instance in the cloud. Or you can download and run a self-managed version. It's up to you."

At the bottom of the main content area, there are two buttons: "Cloud" and "Download". The "Cloud" button is currently selected, and the text "Start in the Cloud" is displayed below it.

Stardog Studio



- Click on Access Studio Now

Start Stardog

- Stardog has two CLI commands both with help
 - **stardog-admin** is for administrative controls, like starting or stopping the server, and adding users
 - **stardog** for queries,
- Start Stardog listening to its default port (5820)
stardog-admin server start
- Enter the URL <http://localhost:5820> to access the Web console
Use admin for both the user and password

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 Studio



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 an 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