Building an OWL Ontology in Protégé

Nicolò Pratelli

Istituto di Scienza e Tecnologie dell’Informazione
Consiglio Nazionale delle Ricerche – Pisa

A.A. 2024-2025

In past seminars

So far, you have learned how to:

- write RDF using Turtle syntax
- create an RDFS ontology in Turtle
- write SPARQL queries and run them using Blazegraph

Recap About OWL

OWL (Web Ontology Language) is a semantic web language designed as a more expressive extension of RDF/S, adding both vocabulary and semantics

OWL can be used to build ontologies, by defining:

- **classes expressions** and **class hierarchies**
- **properties expressions** and **property hierarchies**
- **axioms** about classes expressions and properties expressions

OWL can also be used to build knowledge bases with individuals and assertions about them

Protégé Desktop

Today we will use the desktop version of **Protégé**, a tool for creating ontologies based on OWL

Protégé also allows you to populate the ontology with individuals, thereby creating a knowledge base (TBox + ABox)

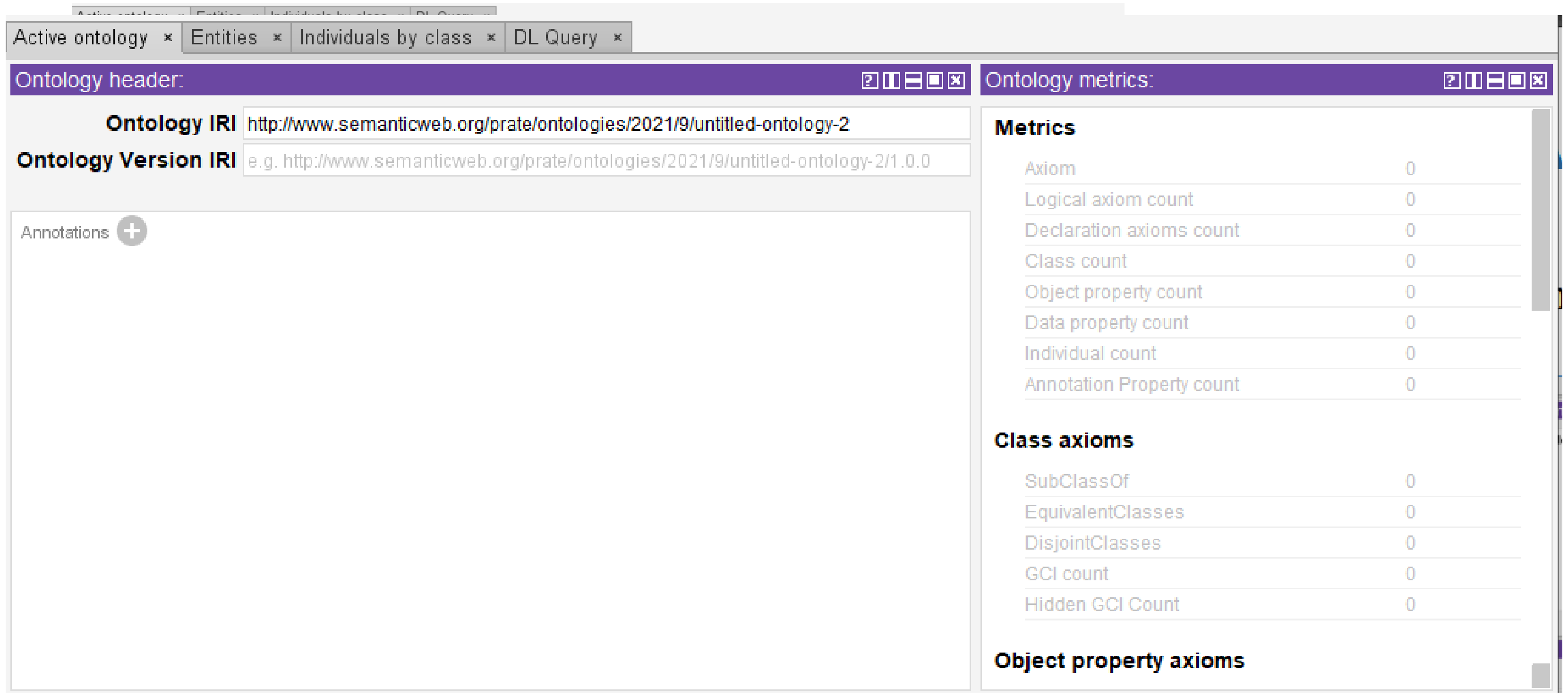
We will use the desktop version because it has better support for the features of OWL

Protégé desktop is available at the following address:

https://protege.stanford.edu/products.php#desktop-protege

Active Ontology Tab

Active Ontology is the first tab that appears at launch


Active ontology × | Entities × | Individuals by class × | DL Query ×

Ontology header:

Ontology IRI http://www.semanticweb.org/prate/ontologies/2021/9/untitled-ontology-2

Ontology Version IRI e.g. http://www.semanticweb.org/prate/ontologies/2021/9/untitled-ontology-2/1.0.0

Annotations

Ontology metrics:

Metrics

Axiom	0
Logical axiom count	0
Declaration axioms count	0
Class count	0
Object property count	0
Data property count	0
Individual count	0
Annotation Property count	0

Class axioms

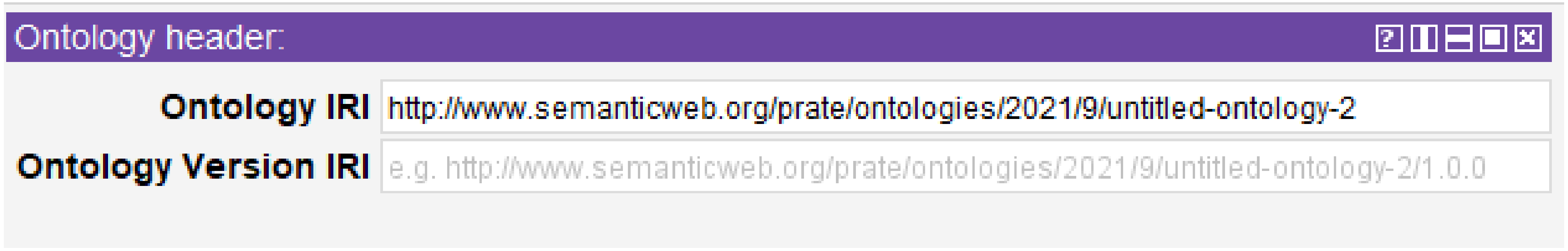
SubClassOf	0
EquivalentClasses	0
DisjointClasses	0
GCI count	0
Hidden GCI Count	0

Object property axioms

We will use it to define the ontology IRI, some annotations about the ontology, and the prefixes

Defining the Ontology IRI

Go to the Active Ontology tab, Ontology header section, and click on the Ontology IRI field:


Ontology header:

Ontology IRI	http://www.semanticweb.org/prate/ontologies/2021/9/untitled-ontology-2
Ontology Version IRI	e.g. http://www.semanticweb.org/prate/ontologies/2021/9/untitled-ontology-2/1.0.0

In general, you can use whatever IRI you like, but in the exercises, you will be asked to use the following IRI:

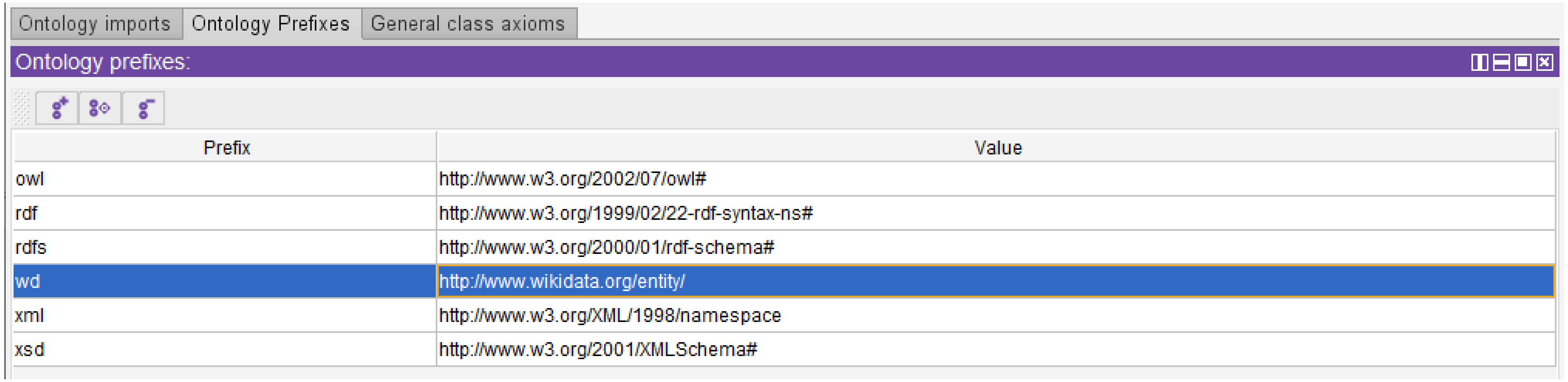
```
http://semanticwebcourse.org/surname/seminar5
```

Define the Project Prefixes

Protégé is already aware of several common prefixes, including `rdf:`, `rdfs:`, `foaf:`, and more

For example, if you insert `rdfs:label` as the name of a property, the software will automatically recognize it

To define a new prefix, go to the Active Ontology tab, Ontology prefixes section, and insert for example `wd:`

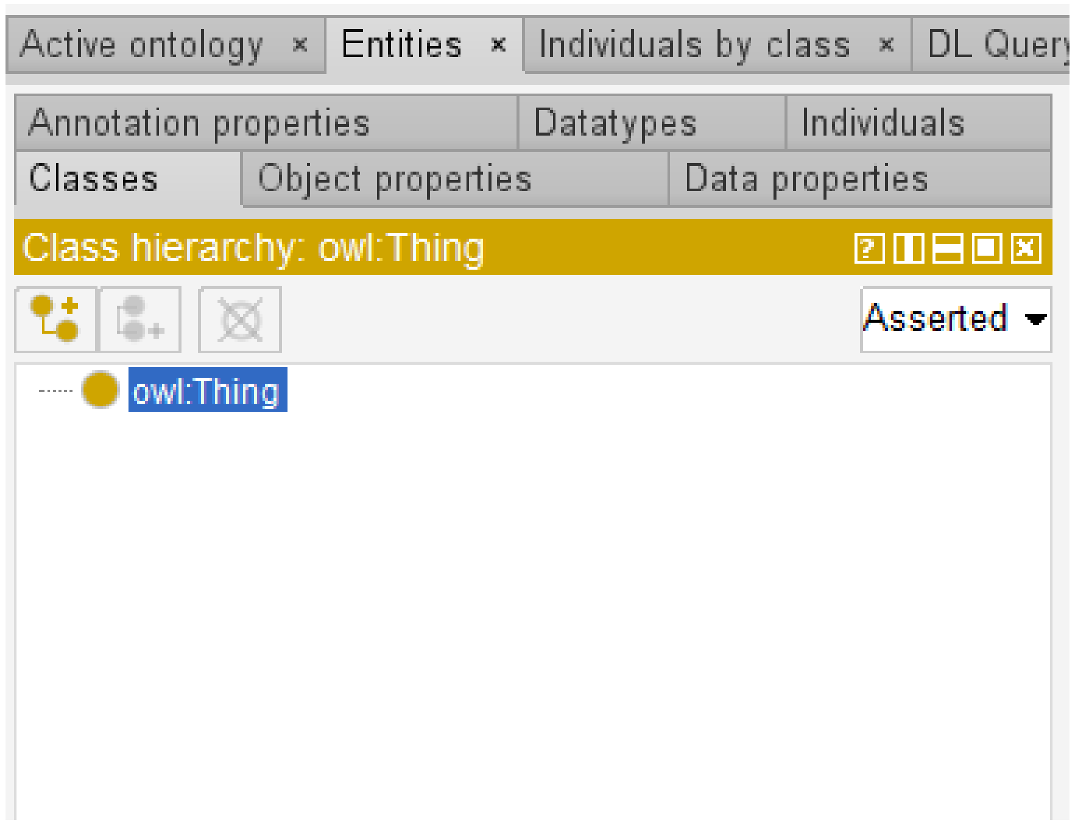


Prefix	Value
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
wd	http://www.wikidata.org/entity/
xml	http://www.w3.org/XML/1998/namespace
xsd	http://www.w3.org/2001/XMLSchema#

Entities Tab

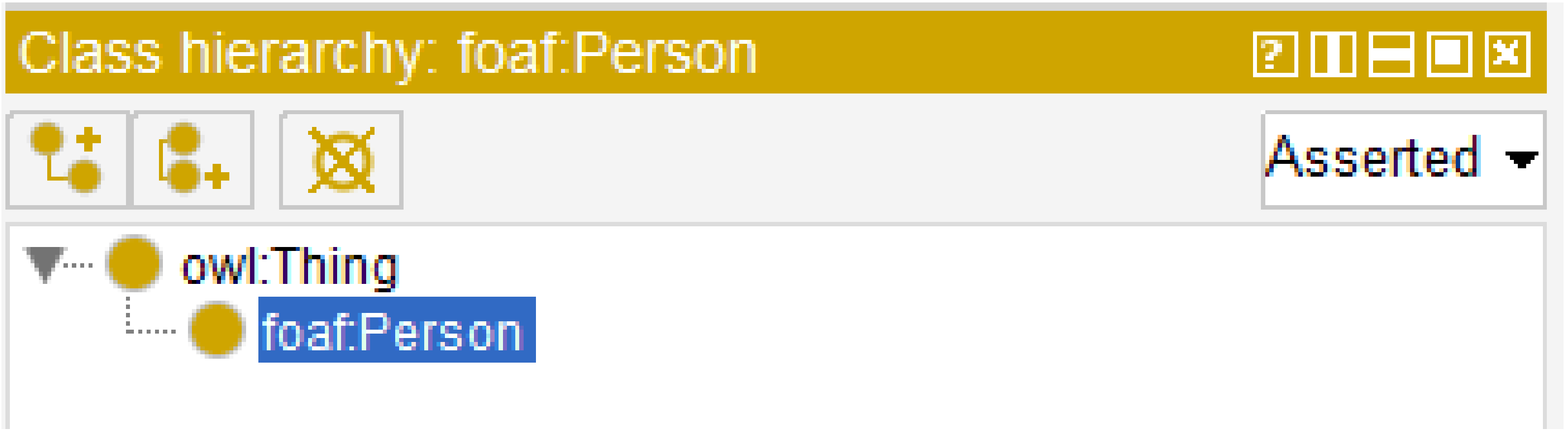
In the **Entities tab**, you can find all the entities of your knowledge base, including:

- classes
- object properties
- data properties
- annotation properties
- datatypes
- individuals



Creating Classes

To create a class, go to the **Classes tab**, click on `owl:Thing`, then on , insert the class name (e.g. `foaf:Person`), and finally click Create



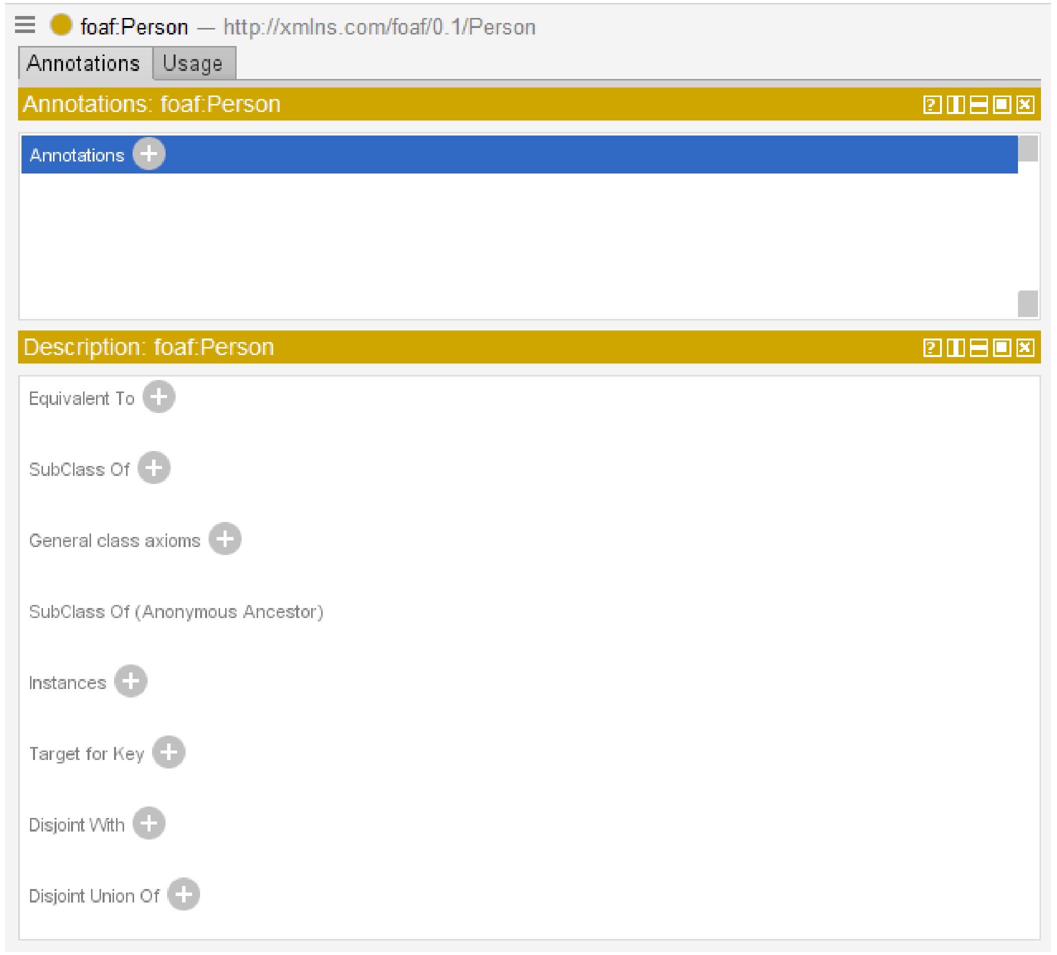
Class hierarchy: foaf:Person

Since Protégé is based on OWL, `owl:Thing` is always the top class and all other classes are created as its subclasses

Class Description

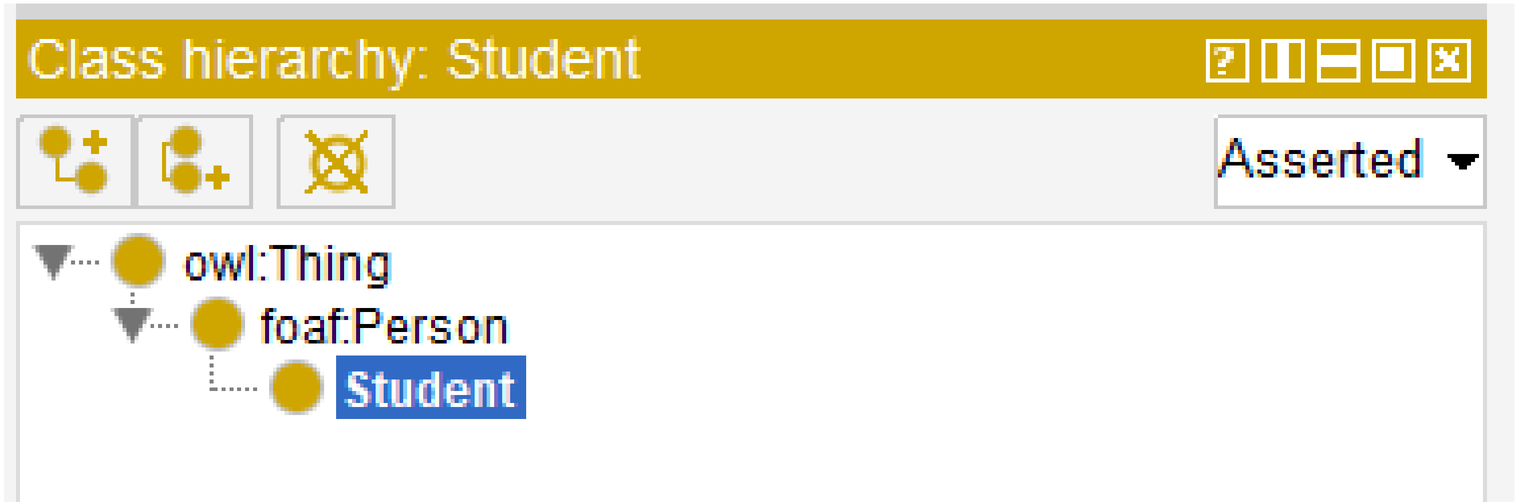
The right-side view describes the selected class

Here you can see the **full IRI** of the class, and also insert annotations and OWL axioms about the class



Class Hierarchy

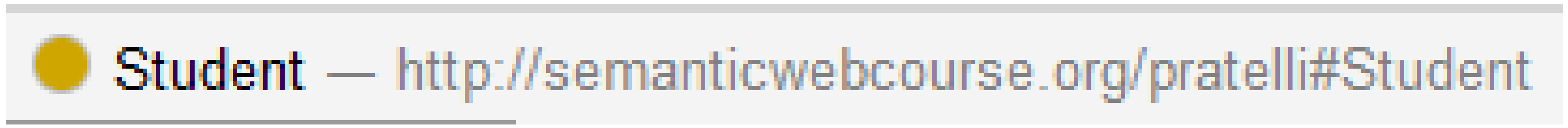
When you click on , the class you insert is added as a subclass of the selected class. For instance, if you select `foaf:Person`, you can add a subclass Student



When you click on , the class you insert is added as a sibling of the selected class. To delete a class, click on

Default IRIs

Notice that if you add a new class without a (known) prefix, for instance “Student”, the desktop version of Protégé assigns to it the IRI of your ontology, plus the name of the class



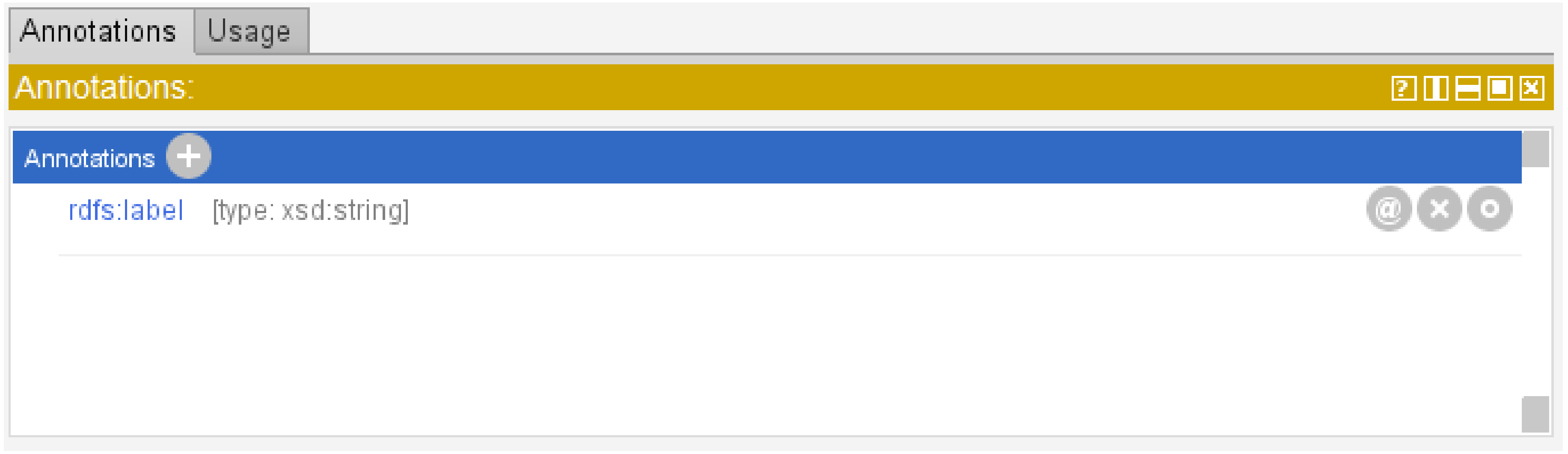
Student — http://semanticwebcourse.org/pratelli#Student

Protégé does not check whether the IRI points to an actual resource or not

Adding a Label

The desktop version of Protégé does not (by default) automatically add labels based on the IRI

You can add a label from the Annotations form in the description of the entity, by clicking on the ⊕ button



rdfs:label [type: xsd:string]

Remember to select the type of the annotation and optionally the language

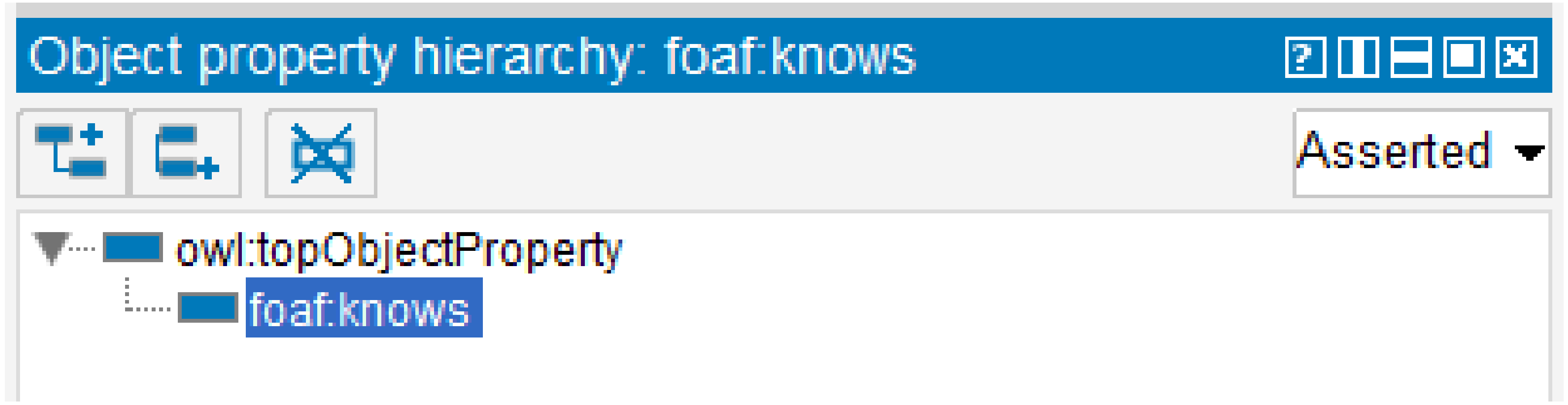
Types of Properties

In Protégé you can define three types of properties:

1. **Object properties** connect an IRI to an IRI (e.g. `foaf:knows`, `dcterms:creator`)
2. **Data properties** connect an IRI to a literal (e.g. `foaf:givenName`, `dcterms:issued`)
3. **Annotation properties** are properties that do not contribute to the "logical" knowledge specified in the ontology, they just provide descriptions (e.g. `rdfs:label`, `rdfs:comment`)

Creating Object Properties

To create an object property, go to the **Object properties tab**, click on `owl:topObjectProperty`, then on , insert the class name (e.g. `foaf:knows`), and finally click Create

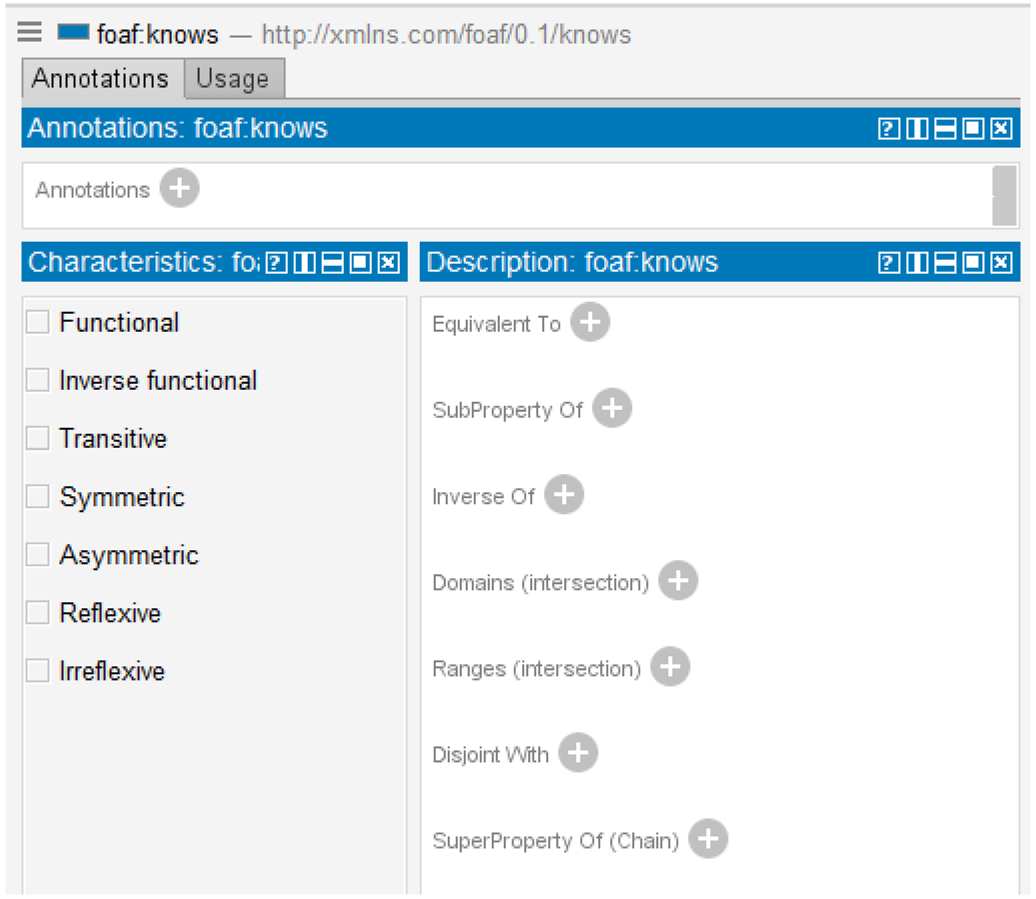


Since Protégé is based on OWL, `owl:topObjectProperty` is always the top object property and all other classes are created as its subproperties

Property Description

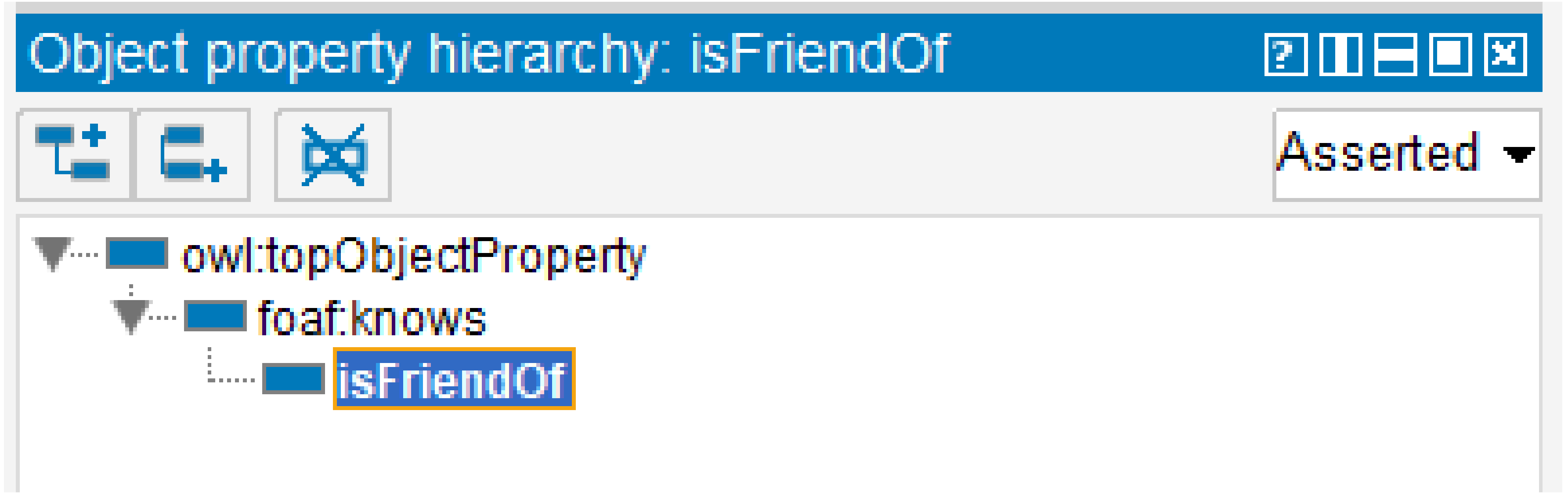
The right-side view describes the selected property

Here you can see the **full IRI** of the property, and also insert annotations and OWL axioms about the properties



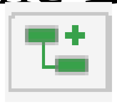
Property Hierarchy

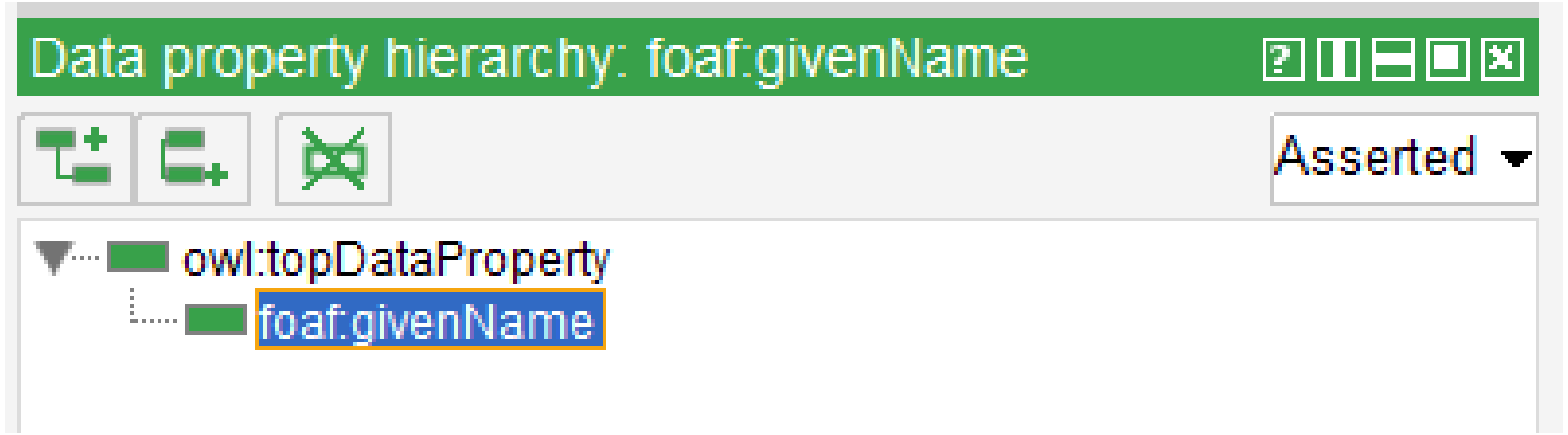
When you click on , the property you insert is added as a subproperty of the selected property. For instance, if you select `foaf:knows`, you can add a subclass `isFriendOf`



When you click on , the property is added as a sibling of the selected property. To delete a property, click on

Creating Data Properties

To create an object property, go to the **Data properties tab**, click on `owl:topDataProperty`, then on , insert the class name (e.g. `foaf:givenName`), and finally click Create



Since Protégé is based on OWL, `owl:topDataProperty` is always the top data property and all other classes are created as its subproperties

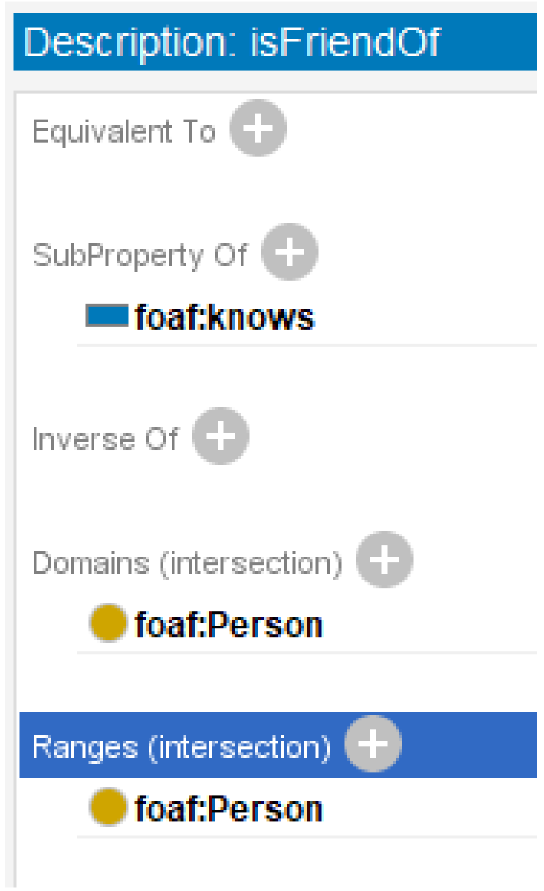
Property Domain and Range

In the right-side view, you also see domain and range

If you click on the ⊕ button, Protégé will let you select a class from the class tree

For example, `:isFriendOf` may have the following domain and range:

- Domain: `foaf:Person`
- Range: `foaf:Person`



Description: isFriendOf

Equivalent To

SubProperty Of
foaf:knows

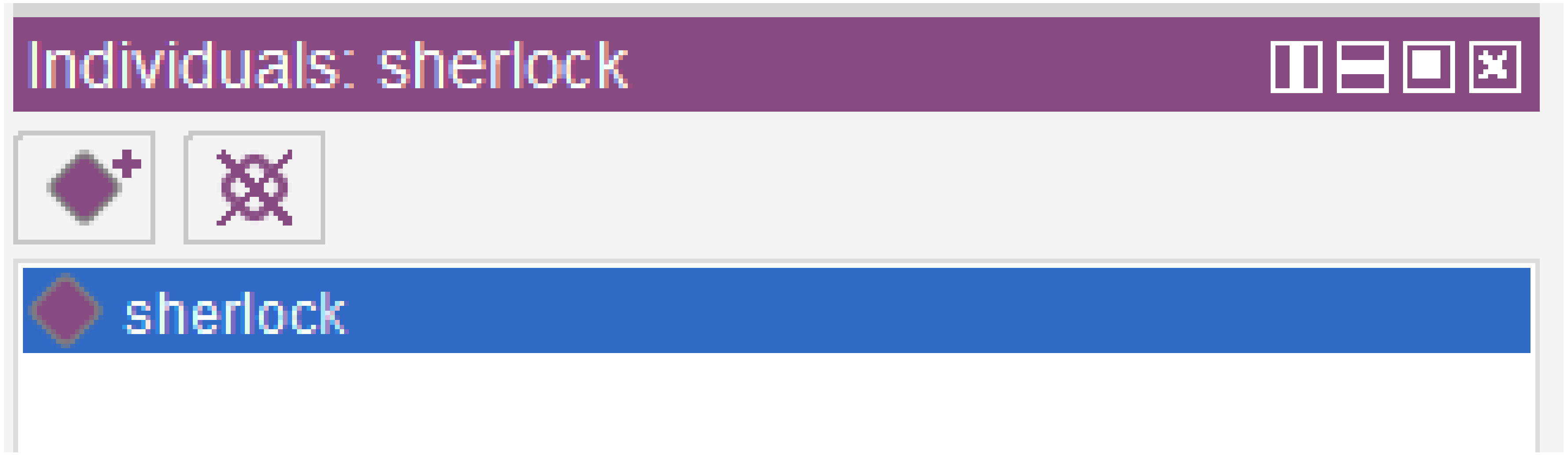
Inverse Of

Domains (intersection)
foaf:Person

Ranges (intersection)
foaf:Person

Creating Individuals

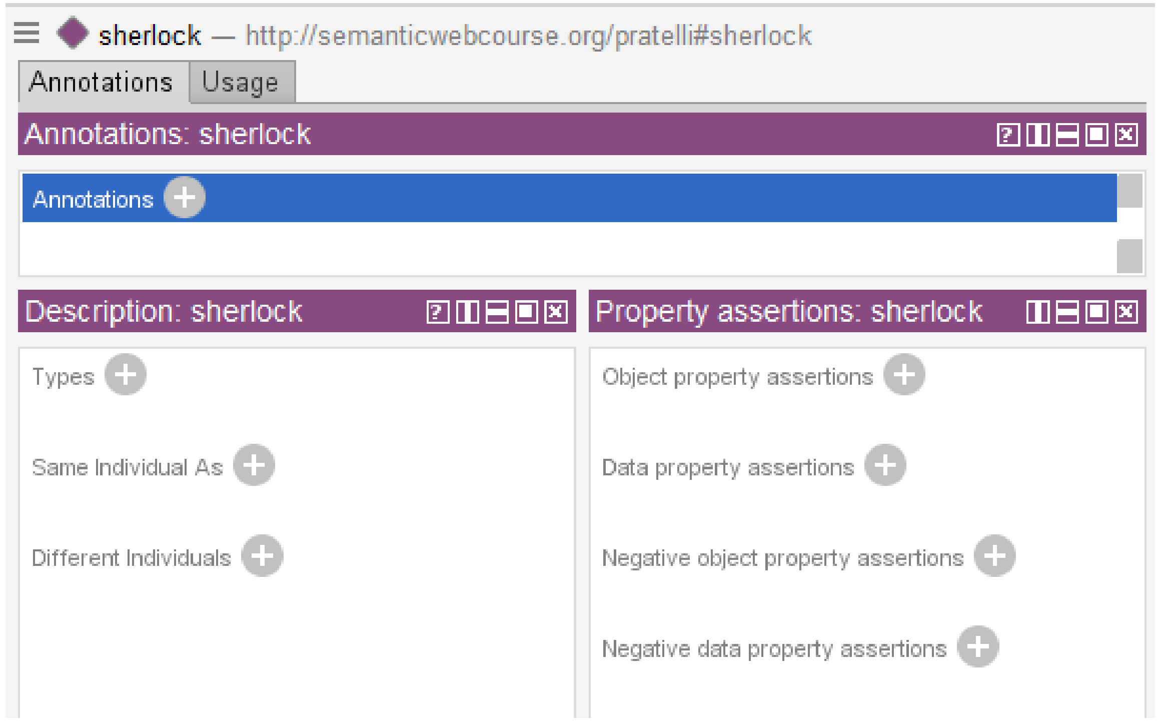
To create an individual, go to the Individuals tab, click on ◆ , insert the individual name (e.g. `sherlock`), and finally click Create



To delete an individual, select it and click on ⊗

Individual Description

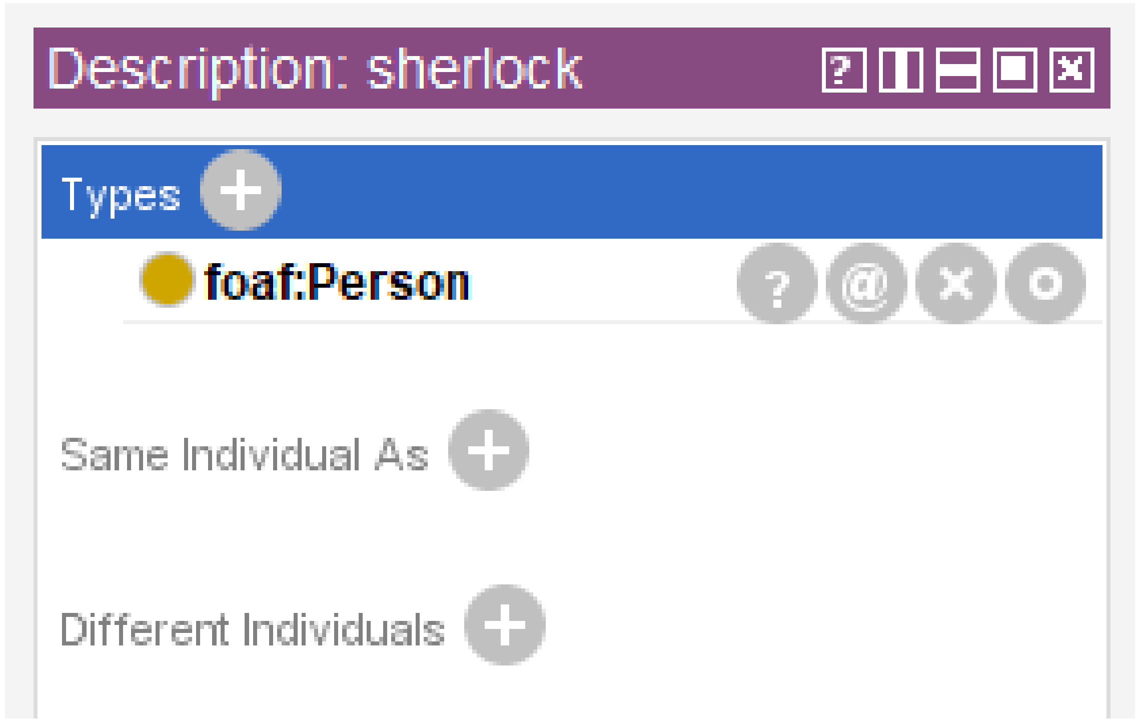
The right-side view describes the selected individual


sherlock — http://semanticwebcourse.org/pratelli#sherlock
Annotations
Usage
Annotations: sherlock
Annotations
Description: sherlock
Types
Same Individual As
Different Individuals
Property assertions: sherlock
Object property assertions
Data property assertions
Negative object property assertions
Negative data property assertions

Here you can see the full IRI of the individual and insert annotations, OWL axioms, and assertions about it

Classifying Individuals

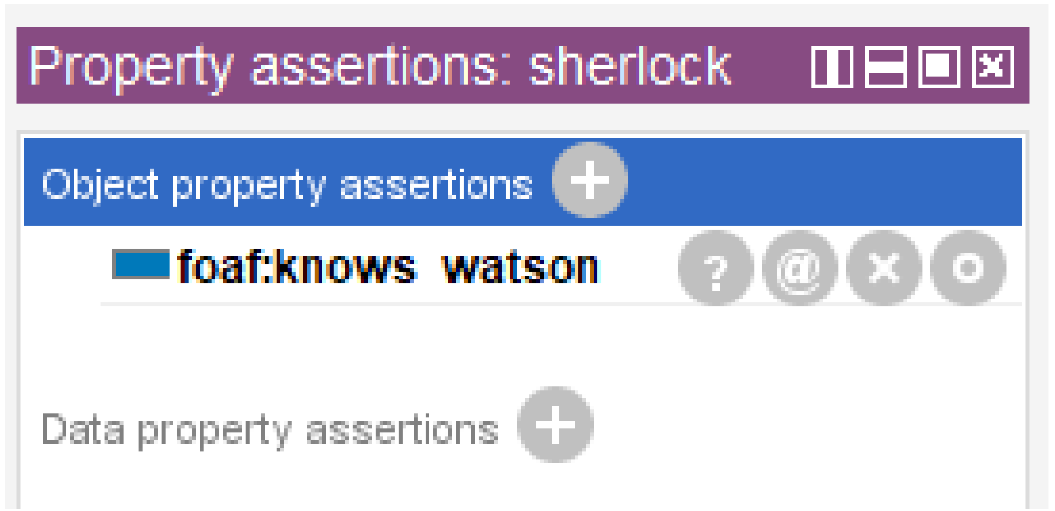
In the **Individuals tab**, under Description, click on the click on the ⊕ button near Types



Protégé will let you select a type from the class tree in order to classify the individual

Creating Assertions

In the Individuals tab, under Property assertions, you can define assertions on the selected individual


Property assertions: sherlock

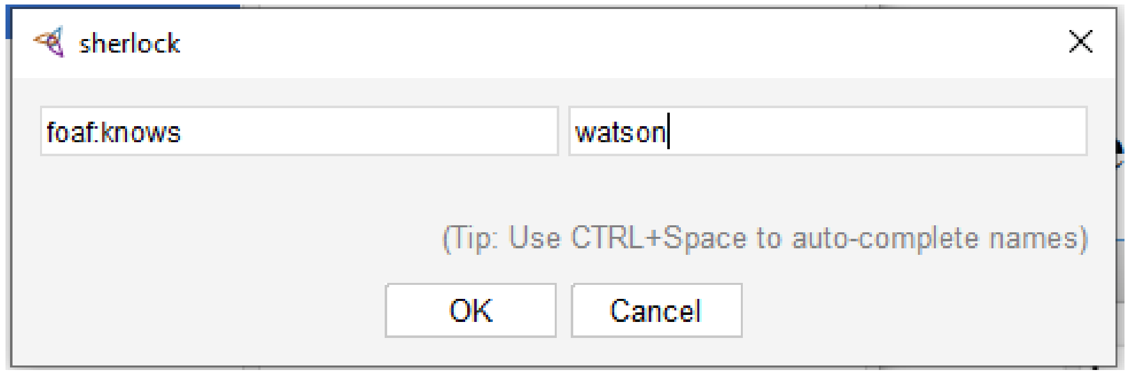
Object property assertions

foaf:knows watson

Data property assertions

For instance, you can state that `sherlock foaf:knows watson`

You can also use auto-complete


sherlock

foaf:knows

watson

(Tip: Use CTRL+Space to auto-complete names)

Instructions for the Exercise

- Today we will create a simple ontology about **pizza** (based on the original Pizza Ontology by Drummond, Horridge, Wroe & Steven)
- Create a new ontology in Protégé using this IRI: `http://semanticwebcourse.org/surname/seminar5`
- Define two additional prefixes:

wd	http://www.wikidata.org/entity/
dctypes	http://purl.org/dc/dcmitype/

- Define the **classes**, **properties**, **individuals**, and **assertions** that you can find in the following slides
- Save the resulting knowledge base in Turtle format and ask us to verify its correctness after you have finished.

Exercise (1/4)

- Define the following **classes** and their **subclasses**:
    - *Food* has subclasses *Pizza*, *PizzaBase*, *PizzaTopping*, *Salad*, and *IceCream*
    - *Pizza* has subclass *NamedPizza*
    - *PizzaBase* has subclasses *RedBase* and *WhiteBase*
    - *PizzaTopping* has subclasses *CheeseTopping*, *MeatTopping*, *FishTopping*, and *VegetableTopping*
    - *CheeseTopping* has subclasses *MozzarellaTopping*, *BrieTopping*, *GorgonzolaTopping*, *ParmigianoTopping*, and *ProvolaTopping*
    - *MeatTopping* has subclasses *HamTopping* and *SausageTopping*
    - *VegetableTopping* has subclasses *BasilTopping*, *OliveTopping*, *CaperTopping*, *GarlicTopping*, *ArtichokeTopping*, and *MushroomTopping*
    - *FishTopping* has subclass *AnchoviesTopping*
    - *NamedPizza* has subclasses *MargheritaPizza*, *MarinaraPizza*, *HamMushroomPizza*, *PisanPizza*, *FourCheesePizza*, *FourSeasonPizza*
    - *foaf:Organization* has subclass *Pizzeria*
    - *dctypes:Location* has subclass *City*
    - *foaf:Person* has no subclasses

Exercise (2/4)

- Define the following object properties, their subproperties, and their domain and range:
  - **hasIngredient** has domain *Food*, range *Food*, and subproperties **hasBase** and **hasTopping**
  - **hasBase** has domain *Pizza* and range *PizzaBase*
  - **hasTopping** has domain *Pizza* and range *PizzaTopping*
  - **wasMadeBy** has domain *Pizza* and range *Pizzeria*
  - **wasBoughtBy** has domain *Pizza* and range *Person*
  - **isBasedIn** has domain *Pizzeria* and range *City*

Exercise (3/4)

- Now define the following individuals and put them in the appropriate classes:
  - instances of *MargheritaPizza*: margherita1, margherita2
  - instance of *MarinaraPizza*: marinara1
  - instances of *Pizzeria*: pizzeriaVesuvio, pizzeriaLaTorre
  - instances of *Person*: johnSmith, aliceBrown
  - instances of *City*: wd:Q2634 (Naples), wd:Q13375 (Pisa)

Exercise (4/4)

- Define the following assertions on the individuals:
  - margherita1 **wasMadeBy** pizzeriaVesuvio
  - margherita2 **wasMadeBy** pizzeriaLaTorre
  - marinara1 **wasMadeBy** pizzeriaLaTorre
  - margherita1 **wasBoughtBy** johnSmith
  - margherita2 **wasBoughtBy** aliceBrown
  - marinara1 **wasBoughtBy** aliceBrown
  - pizzeriaVesuvio **isBasedIn** naples
  - pizzeriaLaTorre **isBasedIn** pisa