Pizzas in 10 Minutes

A Quick Demonstration of Handy Shortcuts and Features for Protege 4,5 and newer.

by Alan Rector

Back to Protege Desktop User Documentation

Contents [hide]

Introduction

Protégé is a powerful editing environment with many features. So many features that many users have not found most of them. This is a quick trip through how to build our standard Pizza ontology in 10 minutes or less of actual typing.

If you have any problems with this walkthrough it is advised that you download and run through the much more substantial Protege OWL tutorial. That tutorial is intended to teach you about OWL. This brief note is to show how to use Protégé efficiently.

We assume you already know what you are doing and have a list of the main elements of the ontology written down in pencil or sketched in a text editor, mind-map, concept map or whatever tool you prefer. (Of course you have to have done the thinking first. This is just how to get the results of thinking into Protégé as easily as possible.)

We assume you have a basic structure already defined “on paper”

Main categories

- Pizza
- Pizza_base
- Pizza_topping

Lists of each kind of main topic

- Pizza_base ⟵ Thick_crust, Thin_crust,…
- Pizza_topping ⟵ Tomato_topping, Mozarella_topping, Spicy_beef_toping, Pepperoni_topping, …
- Pizzas (primitive) ⟵ Margherita, Hot_and_spicy, Seafood, …
- Pizzas (define) Vegetarian*, Cheesey*

It is useful for a very fast version to know which are going to be defined classes. -- deal with primitive and defined classes separately.

A list of the properties and their domain and ranges

- has_topping
- has_base

Start Protégé and configure it

Before beginning this part of the tutorial please familiarise yourself with Protege Desktop with the Quick start Guide.

Install the appropriate plugins

If you don't already have them, download and unzip the following plugins into your plugins directory

- Annotation template
- Matrix

Start Protege and create a new ontology

- Start Protege (preferably from a shell using one of the scripts provided, as this will give you additional feedback)
- Select **Create new OWL ontology**
- Give the ontology a pizza-related URI
- Specify where you wish to save the ontology

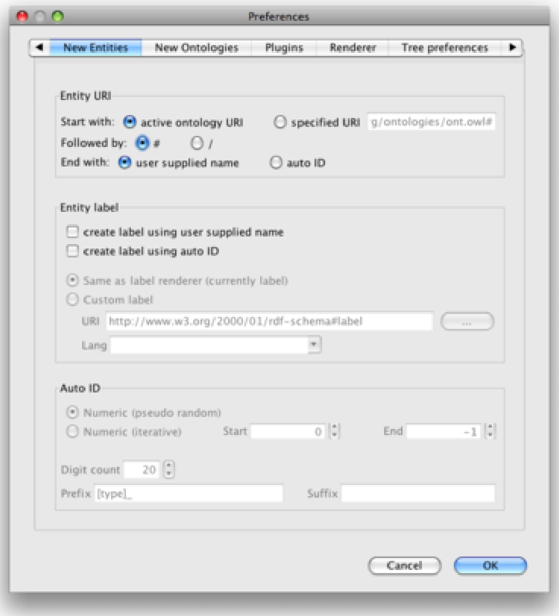
Configure the UI

Setup tabs you will need (and not need)

- Enable the matrix views **Window | Tabs | Matrix** and **Window | Tabs | Property Matrix** that will be used later for fast entry
- Disable the individuals, object and data property and classes tabs (you can use the entities tab for all of these)

Setup the renderer and how new entities will be created

- Open preferences (**File | Preferences** on windows, **Protege | Preferences** on mac)
- Select the **new entities** tab
- In the **Entity URI** pane, select **auto ID**. When you create a new class, property or individual, Protege will give it a meaningless URI and a readable label. That way if you exchange ontologies, correcting spelling mistakes (by merely changing labels) won't cause the links between the ontologies to break
- Select the **renderer** tab
- Select **Render entities using annotation values**



Entity URI
Start with: [✓ active ontology URI] specified URI g/ontologies/ont.owl#
Followed by: [✓ #] /
End with: [✓ user supplied name] auto ID

Entity label
☐ create label using user supplied name
☐ create label using auto ID
[✓ Same as label renderer (currently label)]
Custom label
URI http://www.w3.org/2000/01/rdf-schema#label
Lang

Auto ID
[✓ Numeric (pseudo random)]
Numeric (iterative) Start 0 End -1
Digit count 20
Prefix [type]_ Suffix

Setup the entities tab

- Select the entities tab
- Add the annotation template view from **View | Misc Views | "Annotation Template** in the top part of the right hand pane (see configuring the interface if you've not done this before).

The default fields in the annotations view should be suitable but if you wish to change them you can do this in **Preferences | Annotation Template**

If you want to save this set up, use **Tabs | Store current layout**. Note when you quit Protege this is automatically stored and will persist for future versions of Protege.

Build the ontology

Create the top hierarchy

You don't need to do this, but for a variety of reasons it is "good practice".

- Select the entities tab
- Make sure **Thing** is selected in the class hierarchy
- Select **Tools | Create class hierarchy**
- Create the hierarchy with tab indenting to denote subclasses:

```
Domain_entity
  Independent_entity
  Value
```

Create the skeleton hierarchy

Create your top classes

- Select **Independent_entity**
- Select **Tools | Create Class Hierarchy...**
- Create Pizza, Pizza_topping, and Pizza_base

(If you want you can create the entire hierarchy here using tabs to indicate the hierarchy, but if you do you can't take advantage of the automatic suffix mechanism and will have to type "_base", "_topping", "_pizza" for each entity by hand.)

- Finish the wizard and leave **Make classes disjoint** box ticked

Create each branch of the taxonomy

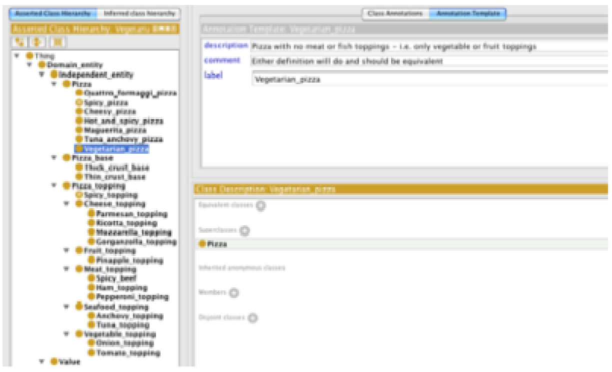
- Select Pizza_topping

- Select **Tools | Create Class Hierarchy…**
- In the **Suffix** box type "_topping"
- Enter the list of toppings and finish as before, using tabs to indicate subclasses
- Repeat for Pizza_base
- Repeat for Pizza, but untick **Make classes disjoint** because we will make some of these classes defined, so we will add the disjoints afterwards.
- Select Value and create the hierarchy for Spiciness← Hot, Medium, Mild, all with the suffix "_value"
- Close the value partition by selecting Spiciness_value and select **Edit | Add covering axiom**

Open the class hierarchy and check

- If you have made any spelling errors you can just correct them in the label field
- Type convenient descriptions in the kinds of pizzas and anything else that is not unambiguous or will be defined.

The ontology now looks as show below for one choice of toppings and kinds of pizza.



```
Thing
  Domain_entity
    Independent_entity
      Pizza
        Quattro_formaggi_pizza
        Spicy_pizza
        Cheesy_pizza
        Hot_and_spicy_pizza
        Magueritta_pizza
        Tuna_anchovy_pizza
        Vegetarian_pizza
      Pizza_base
        Thick_crust_base
        Thin_crust_base
      Pizza_topping
        Spicy_topping
        Cheese_topping
          Parmesan_topping
          Ricotta_topping
          Mozzarella_topping
          Gorgonzolla_topping
        Fruit_topping
          Pinapple_topping
        Meat_topping
          Spicy_beef
          Ham_topping
          Pepperoni_topping
        Seafood_topping
          Anchovy_topping
          Tuna_topping
        Vegetable_topping
          Onion_topping
          Tomato_topping
    Value
```

description: Pizza with no meat or fish toppings – i.e. only vegetable or fruit toppings
comment: Either definition will do and should be equivalent
label: Vegetarian_pizza

Superclasses: Pizza

Create the object properties

- In the object properties view create two top object properties

```
relational_property
modifier_property
```

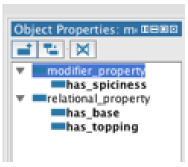
- Create subproperties of relational_property

```
has_topping
has_base
```

- Create a subproperty of modifier_property

```
has_spiciness
```

The object properties tab should now look roughly like

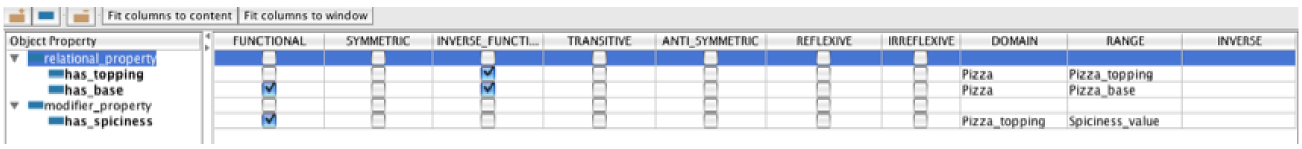


```
modifier_property
  has_spiciness
relational_property
  has_base
  has_topping
```

Add the property characteristics

- Go the **Property Matrix** tab
- Make has_spiciness and has_base both functional
- Fill in the appropriate domains and ranges for each property (You can drag and drop classes from the classes palette into the domain and range)

The whole tab should then look roughly as shown below:



Object Property	FUNCTIONAL	SYMMETRIC	INVERSE_FUNCTI...	TRANSITIVE	ANTI_SYMMETRIC	REFLEXIVE	IRREFLEXIVE	DOMAIN	RANGE	INVERSE
relational_property	☐	☐	☐	☐	☐	☐	☐			
has_topping	☐	☐	✓	☐	☐	☐	☐	Pizza	Pizza_topping	
has_base	✓	☐	✓	☐	☐	☐	☐	Pizza	Pizza_base	
modifier_property	☐	☐	☐	☐	☐	☐	☐			
has_spiciness	✓	☐	☐	☐	☐	☐	☐	Pizza_topping	Spiciness_value	

Add the toppings and the spiciness of toppings

- Go to the **Matrix tab**

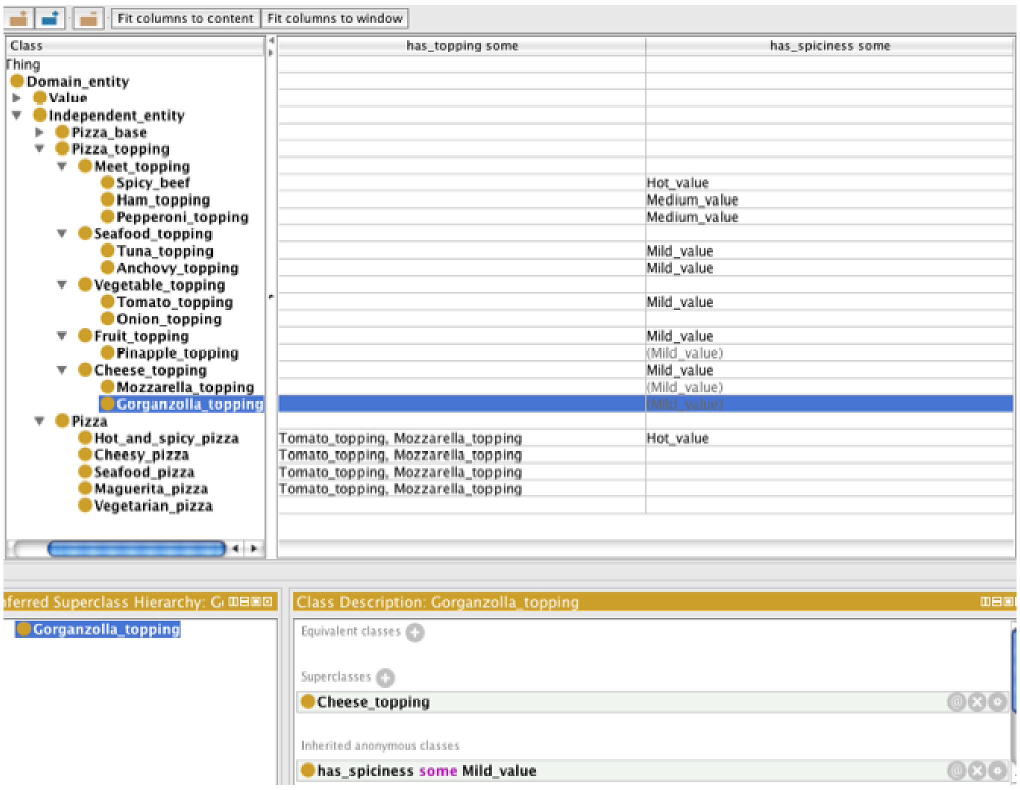
Optional additional setup. You may find it easier to have an additional class description view at the bottom of the matrix tab as shown at the end of this section. If so, go to **View | Class views | Class Description**, and place it in the bottom half so it goes under the main part of the matrix tab.

- Add columns for the properties has_topping and has_spiciness to the matrix
  - Find the properties in the **Object properties palette** on the right of the tab and drag them to the central pane
- Add Mild, Medium or Hot for the various pizza toppings
  - You can now drag and drop values into the matrix - each comma-separated value in a cell is the filler of a someValuesFrom restriction on the class along the property
    - You can multiselect the target classes in order to add the same value to multiple cells at once
    - You can multiselect filler classes to be dragged
  - It might be quicker to edit the cells in place - click a cell and start typing
    - the editor support autocomplete just like expression editors (Ctrl-Space)
    - separate fillers with commas

- If you make a mistake you can either delete the item in place, or you can change it in the class description window if you have placed it at the bottom
- You only need to drag values to parent classes as they will be inherited.

- Add the basic toppings to the Pizzas as well, except for the classes that will have definitions, Vegetarian Pizza, Spicy Pizza, and Cheesy Pizza.

At the end you should see roughly as below:



Class	has_topping some	has_spiciness some
Thing		
Domain_entity		
Value		
Independent_entity		
Pizza_base		
Pizza_topping		
Meet_topping		
Spicy_beef		Hot_value
Ham_topping		Medium_value
Pepperoni_topping		Medium_value
Seafood_topping		
Tuna_topping		Mild_value
Anchovy_topping		Mild_value
Vegetable_topping		
Tomato_topping		Mild_value
Onion_topping		
Fruit_topping		Mild_value
Pinapple_topping		(Mild_value)
Cheese_topping		Mild_value
Mozzarella_topping		(Mild_value)
Gorganzolla_topping		(Mild_value)
Pizza		
Hot_and_spicy_pizza	Tomato_topping, Mozzarella_topping	Hot_value
Cheesy_pizza	Tomato_topping, Mozzarella_topping	
Seafood_pizza	Tomato_topping, Mozzarella_topping	
Maguerita_pizza	Tomato_topping, Mozzarella_topping	
Vegetarian_pizza		

nferred Superclass Hierarchy: G

Gorganzolla_topping

Class Description: Gorganzolla_topping

Equivalent classes

Superclasses

Cheese_topping

Inherited anonymous classes

has_spiciness some Mild_value

Finish off the pizza definitions

- Return to the class tab
- Close the pizzas toppings so that no others can be added
  - Close Margherita_pizza
    - Select Margherita_pizza
    - right click on one of the has_topping restrictions in the **Description** view (on the mac you might need to first select the restriction, then right click).
    - In the menu that appears select **Create closure axiom**. This creates an AllValuesFrom restriction along the property in the restriction you selected. The filler of this new restriction will be a union of all the fillers along this property.
  - Repeat for Quatro_formaggi_pizza and Seafood_pizza.

Add the definitions for the defined classes based on existentials

It is usually easier to create defined classes by creating the restrictions individually and then convert it the result to a defined class. If you don't want to use all the restrictions in the definitions, just select the ones you do want to convert, right click and use the convert).

-
  - Create a sublclass of Pizza_topping, Spicy_topping
  - Add a superclass in the description view, *has_spiciness some Hot_value*
  - Choose **Edit | Convert to defined class** (or press the short cut key – **CMD-D** on the Mac, **CTRL-D** on a Windows). All superclasses are merged into a single equivalent class - you will see this move in the description view
  - Select or create a Spicy_pizza
  - Add a superclass in the description view, *has_topping some Spicy_topping*
  - Make it defined by pressing **CMD-D** or **CTRL-D**.

Add the definitions for other defined classes

If the definitions are more complicated, e.g. for Vegetarian Pizza, then you just have to do it the long way. It is still probably easier to add them as individual restrictions and then convert to a defined class

- Add a definition for a Cheesey_pizza = *Pizza and has_topping min 2 Cheese_topping* e.g. that it has at least two kinds of cheese
- Add a definition for Vegetarian Pizza = *Pizza and not(has_topping some (Meat_topping or Fish_topping))* e.g. that it has no meat or fish toppings

Save, Classify and check

The ontology isn't complete until it's been classified and checked!

- Save your work first! In fact, if you have been wise you will have been saving your work as went along.
- Choose a reasoner from the Reasoner menu. This will automatically cause the reasoner to classify your ontology.
- If something turns red, don't panic. It is probably a disjoint axiom. Check them first.

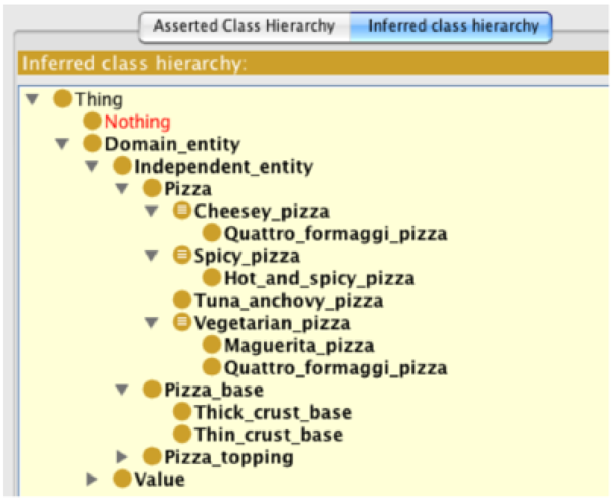
Making corrections

It is the disjoint axioms that are troublesome. If you are changing the class hierarchy:

- If you are moving a class, first remove its disjoint axioms. They will almost certainly be wrong for its new location
- When you are finished, go to a sibling class with disjoint axioms, remove the long disjoint axiom and press **CTRL-J**/**CMD-J** to add back the disjoint axioms on all primitive siblings.

Anything else, just fix in the usual Class Description view.

Back to Protege Desktop User Documentation



```
▼ Thing
    Nothing
  ▼ Domain_entity
    ▼ Independent_entity
      ▼ Pizza
        ▼ Cheesey_pizza
            Quattro_formaggi_pizza
        ▼ Spicy_pizza
            Hot_and_spicy_pizza
          Tuna_anchovy_pizza
        ▼ Vegetarian_pizza
            Maguerita_pizza
            Quattro_formaggi_pizza
      ▼ Pizza_base
          Thick_crust_base
          Thin_crust_base
      ▶ Pizza_topping
    ▶ Value
```