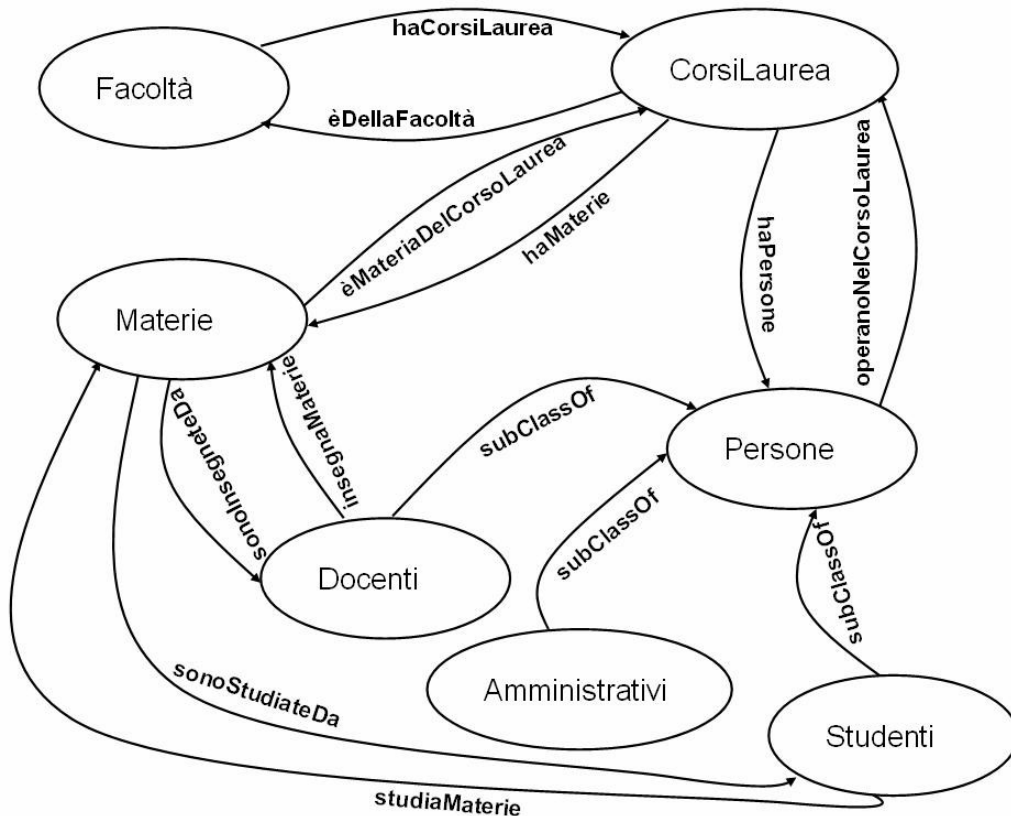


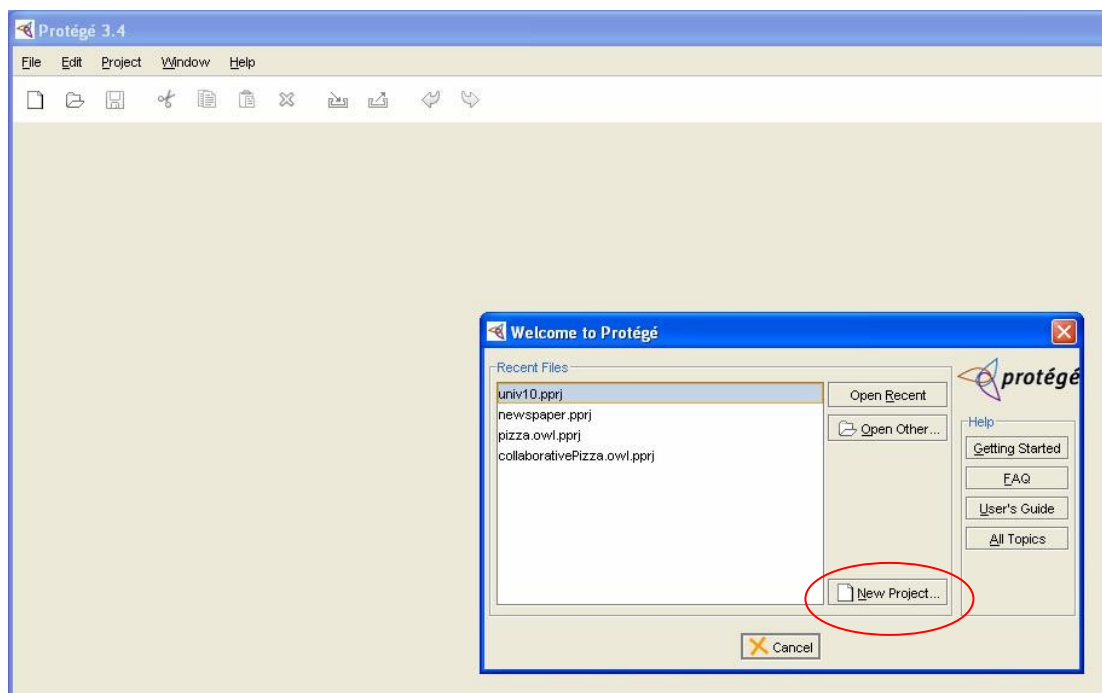
Creazione di una ontologia con Protégé.

Si è scelto di lavorare con Protégé 3.4, piuttosto che con l'ultima release (4.0), in quanto questa non permette ancora l'interazione con il linguaggio SPARQL, che viene usato per interrogare una ontologia (in OWL).

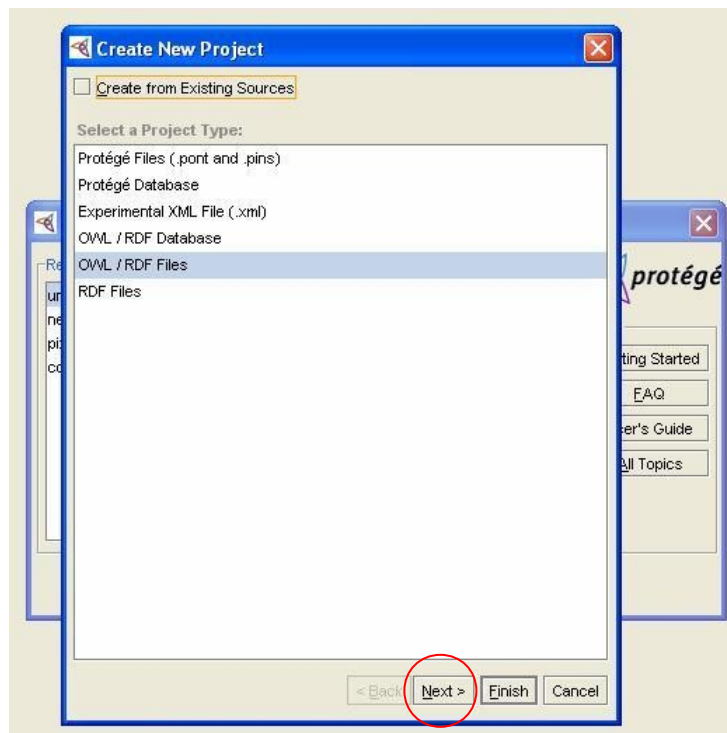
Facciamo riferimento all'ontologia progettata in precedenza che qui sotto riportiamo:



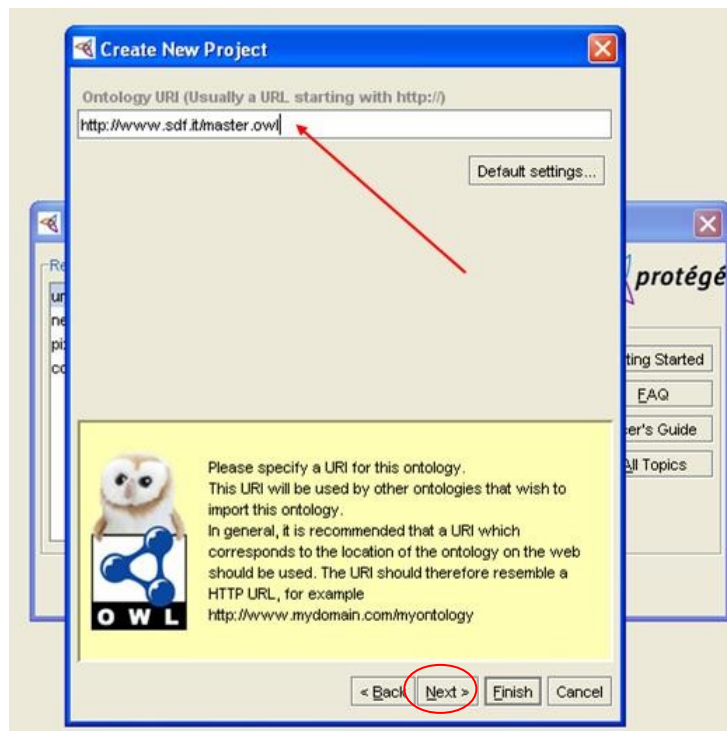
Lanciare Protégé e selezionare **New Project**



Selezionare **OWL/RDF Files** e **Next**

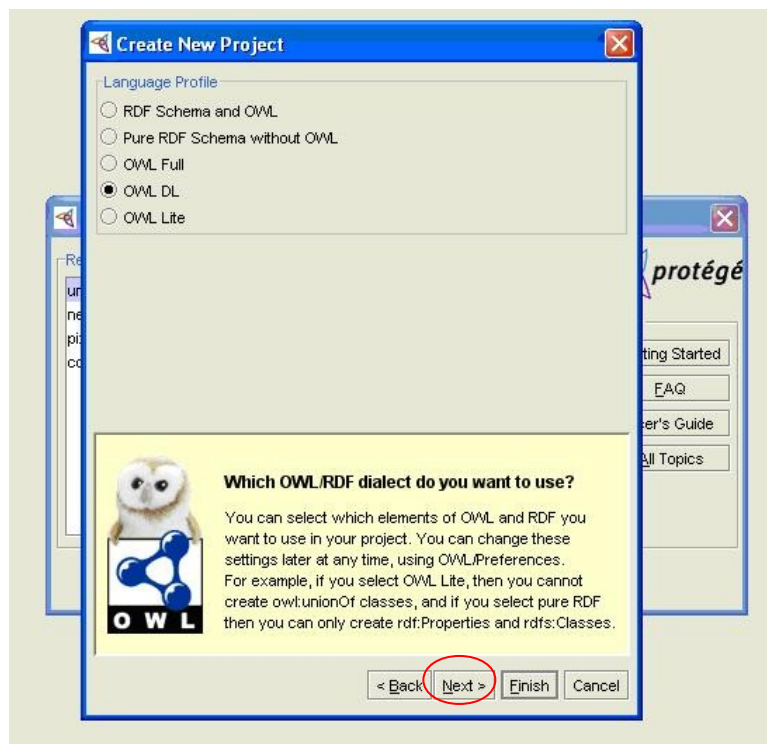


Scrivere **<http://www.sdf.it/master.owl>**¹ (il nome assegnato all'ontologia e master.owl), così come indicato, e poi click su **Next**:

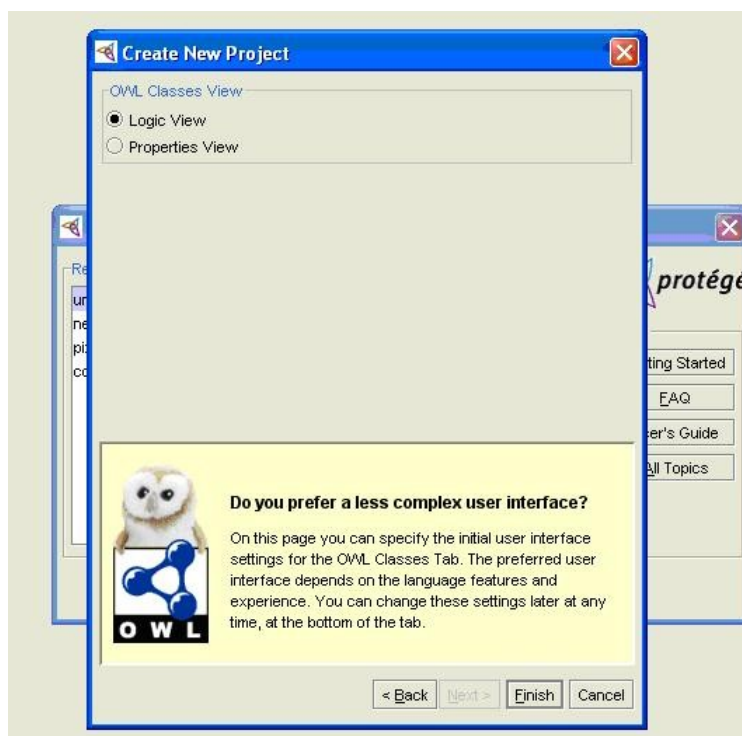


¹ Leggere paragrafo NAMESPACES, in fondo al capitolo.

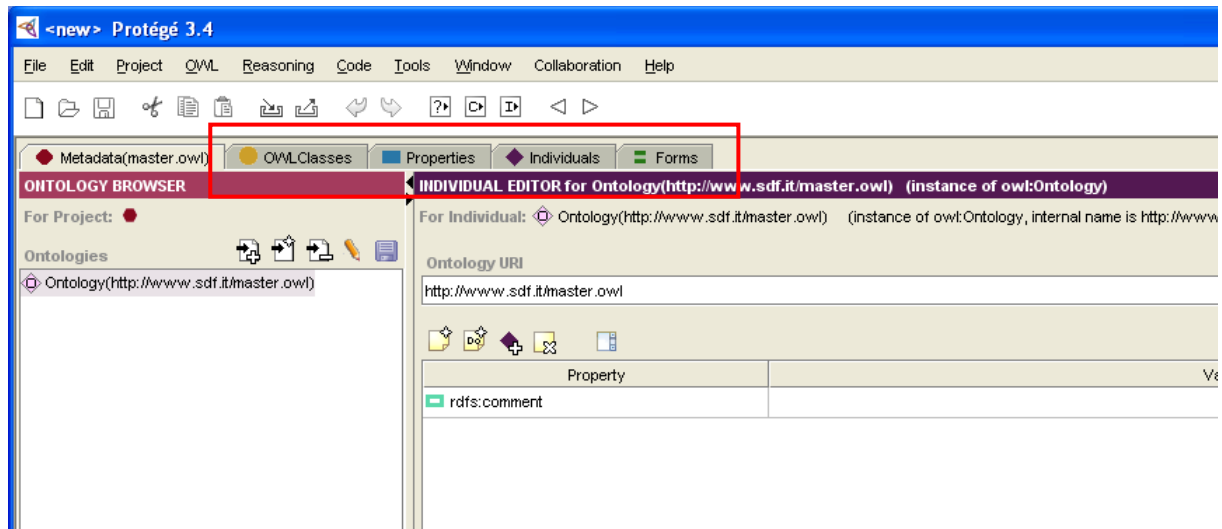
Poi selezionare **OWL DL** e click su **Next**



Confermare **Logic View** e quindi click su **Finish**

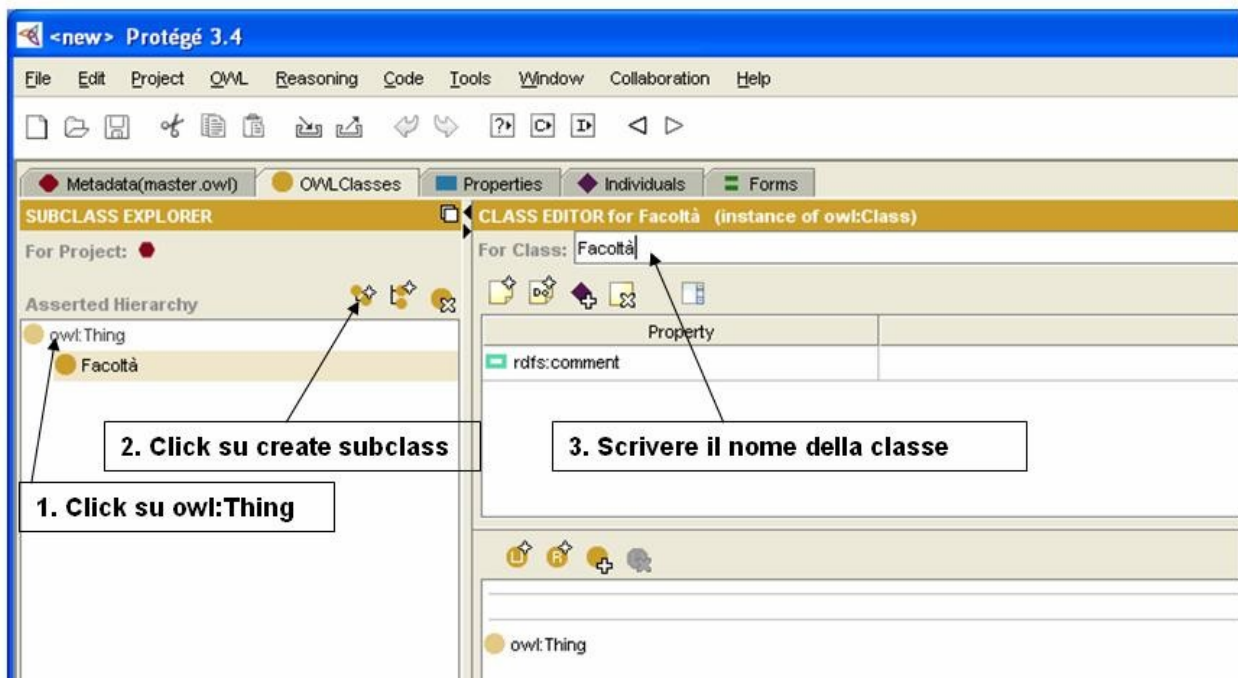


Viene visualizzata la seguente pagina, che rappresenta l'ambiente di lavoro di Protégé. Selezionando via via i tab evidenziati è possibile inserire e gestire classi, relazioni e individui.

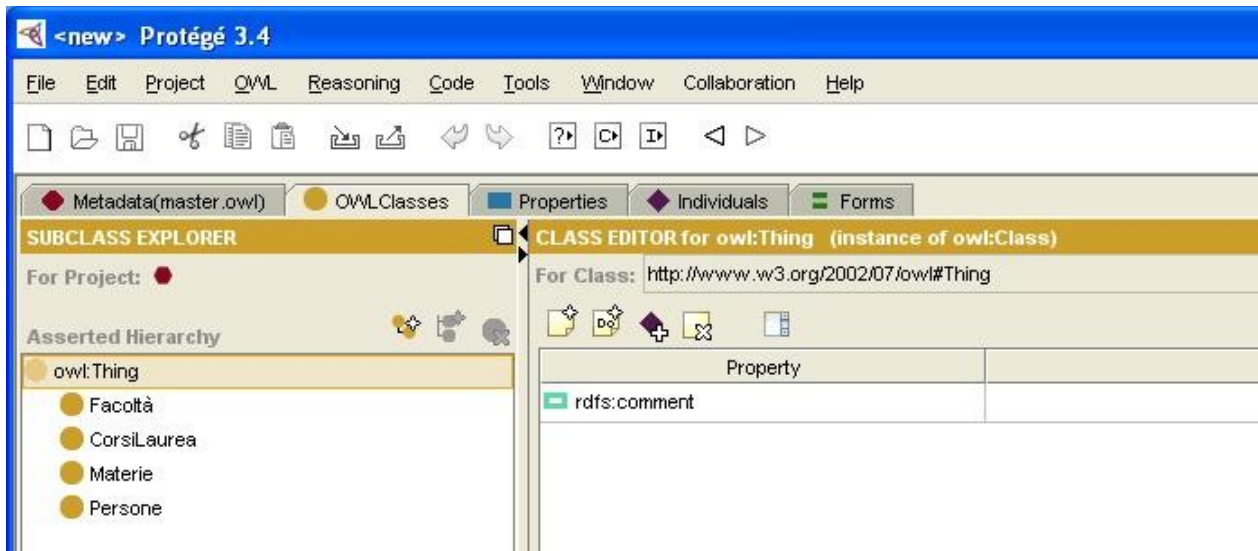


Creare classi

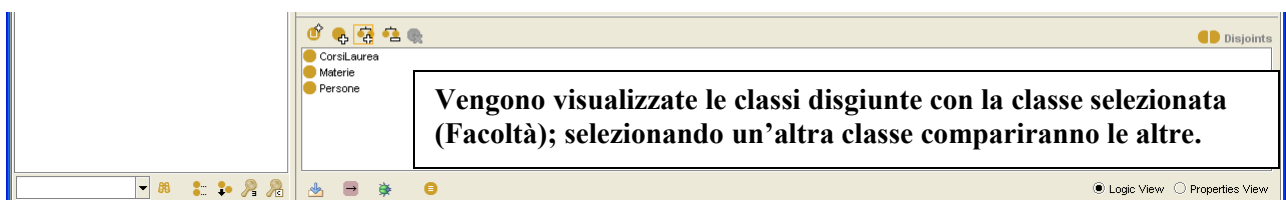
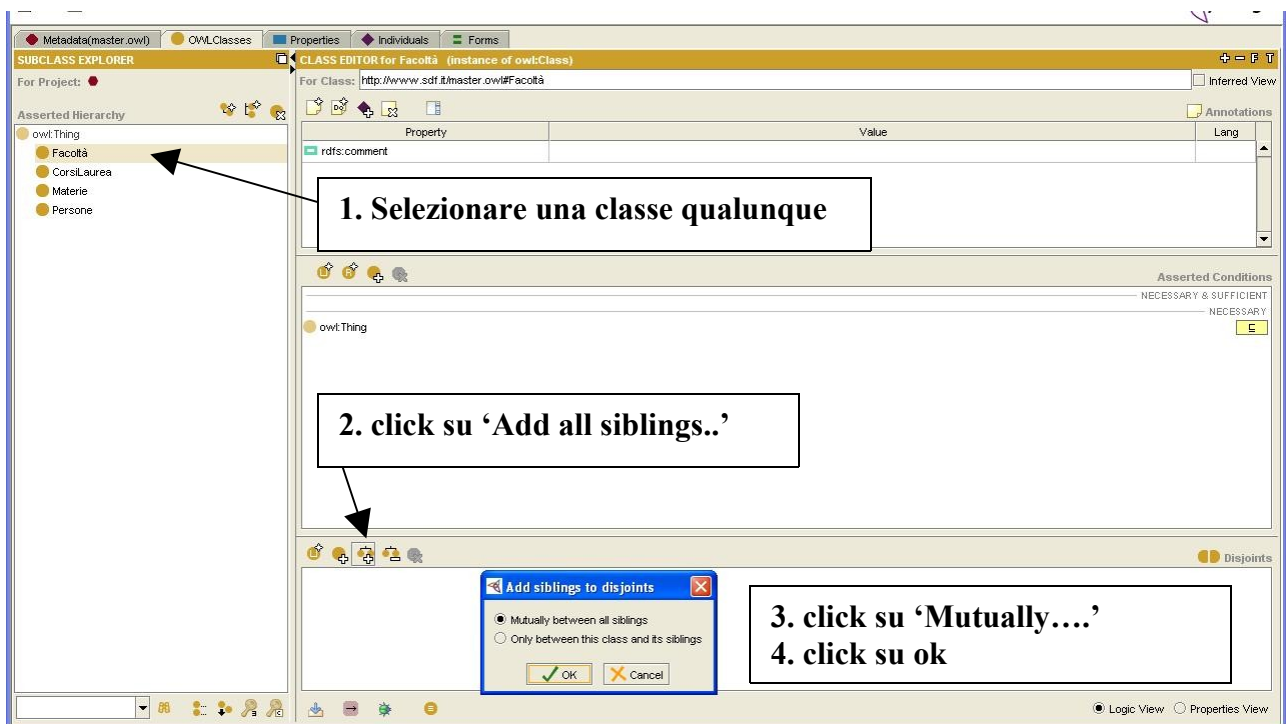
Selezionare OWL Classes; quindi:



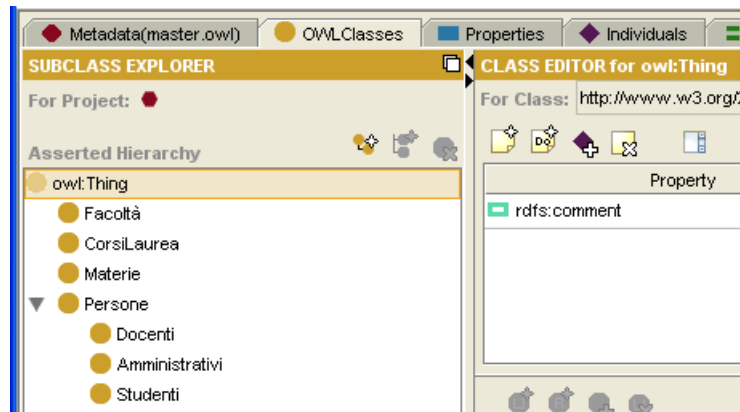
Tutte le classi sono considerate sottoclassi di 'Thing'. Per creare le altre classi (CorsiLaurea, Materie, Persone), ripetere il procedimento appena visto.



Dichiarare le 4 classi, sottoclassi di Thing, disgiunte fra loro; Owl, se non esplicitamente dichiarato, assume che un elemento di una classe possa appartenere a più classi contemporaneamente.



Per aggiungere le sottoclassi Docenti, Amministrativi, Studenti, selezionare la classe Persona e aggiungere le sottoclassi con analogo procedimento a quello adottato in precedenza per aggiungere classi che sono sottoclassi di Thing.



Creare Relazioni

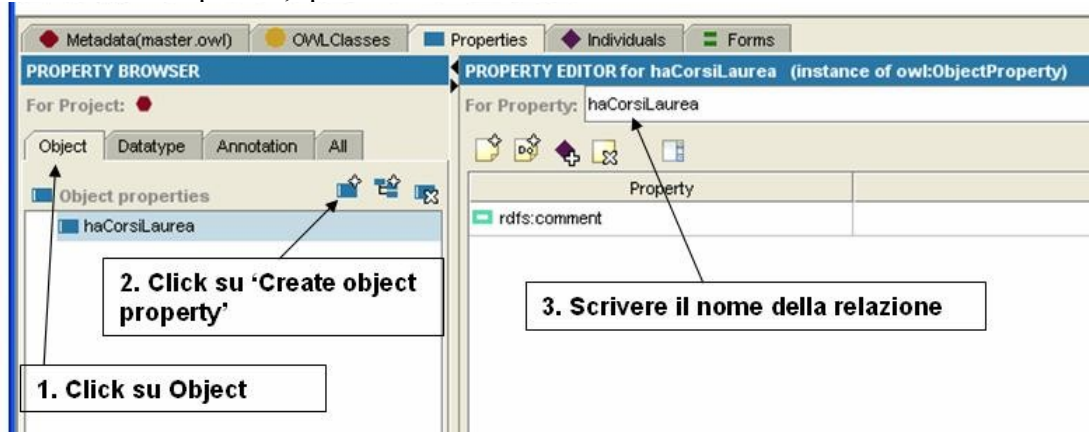
In OWL le relazioni sono proprietà. OWL distingue due tipi di proprietà:

object property: sono le relazioni fra individui appartenenti alle classi (genericamente fra classi);

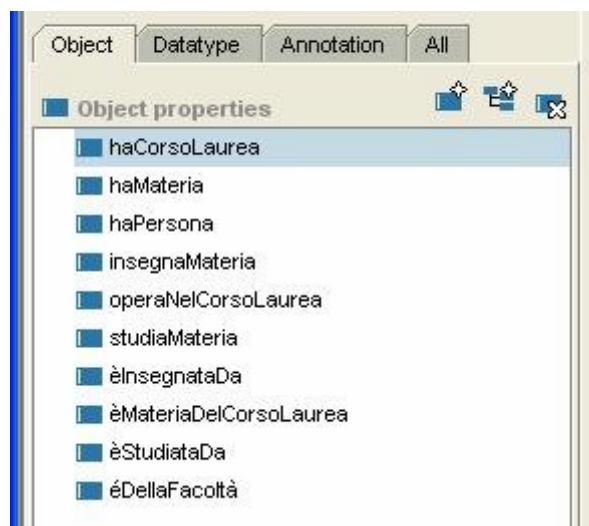
datatype property: sono le relazioni che collegano *gli attributi* degli individui ai tipi ammessi.

Creare *object property* (relazioni).

Selezionare Properties; quindi:

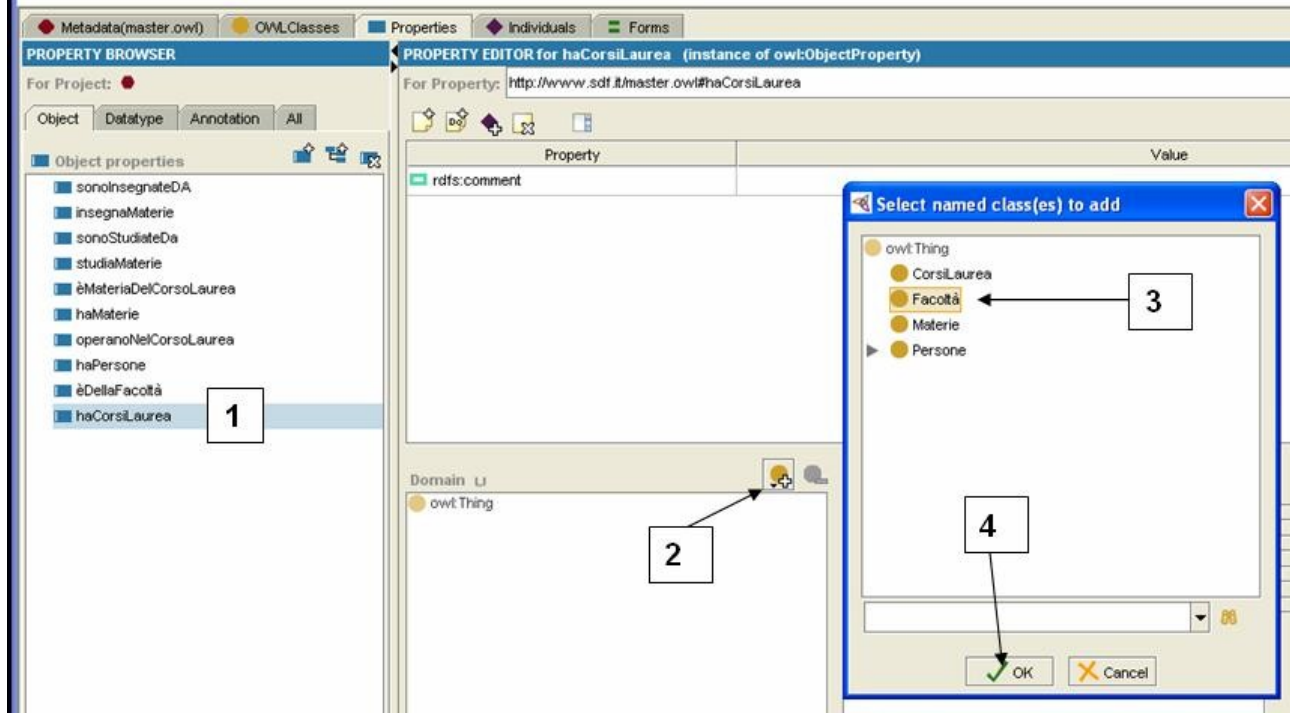


Ripetere il procedimento per tutte le altre relazioni.

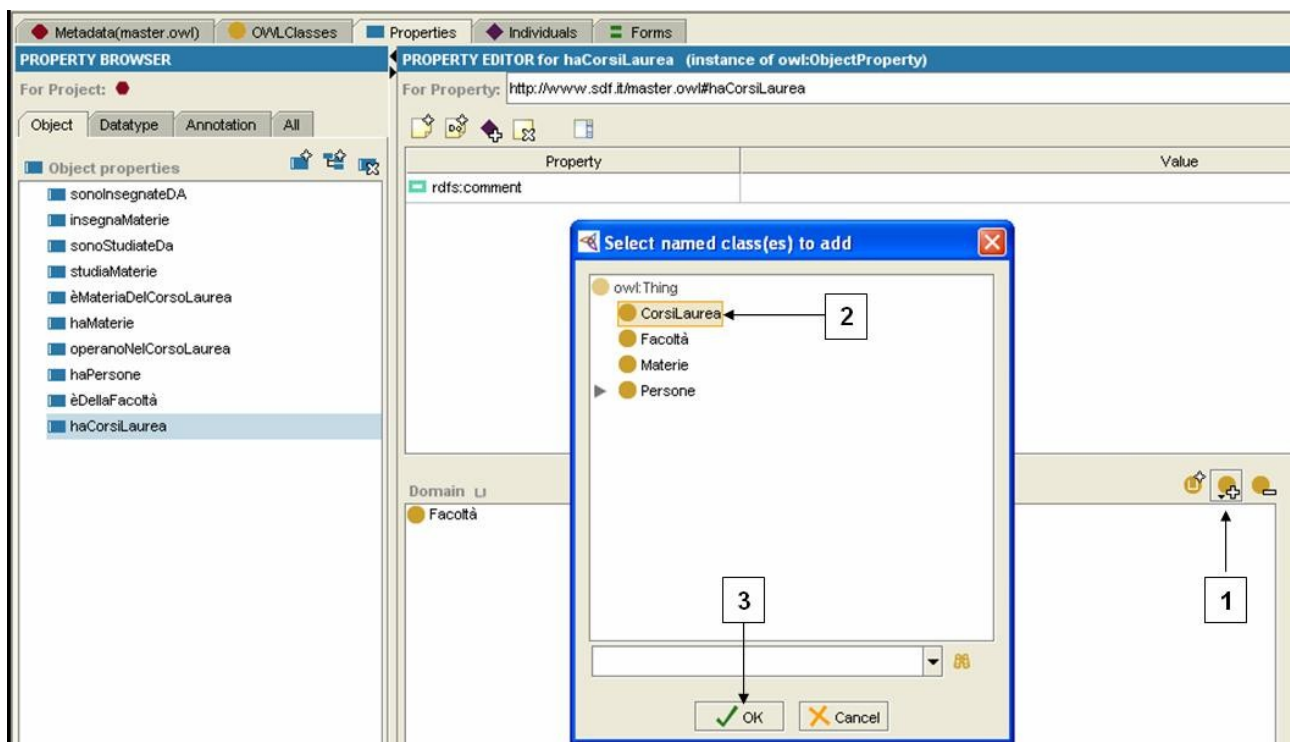


Dobbiamo ora associare le classi alle relazioni create. Dobbiamo cioè stabilire, ad esempio, che la relazione *haCorsiLaurea* si applica agli individui della classe *Facoltà* e si riferisce agli individui della classe *CorsiLaurea*. La classe, ai cui individui si applica la relazione, si chiama *dominio* (*domain*), la classe, ai cui individui si riferisce la relazione, si chiama *condominio* (*range*).

Operare nel seguente modo per determinare il *dominio*:



Comparirà *Facoltà* in Domain; operare nel seguente modo per selezionare il condominio (Range):



In *Range* comparirà *CorsiLaurea*

Allo stesso modo occorre operare per tutte le altre relazioni.

“Each object property may have a corresponding inverse property. If some property links individual a to individual b then its inverse property will link individual b to individual a. For example, Figure 4.16 shows the property `hasParent` and its inverse property `hasChild` — if Matthew `hasParent` Jean, then because of the inverse property we can infer that Jean `hasChild` Matthew.”²

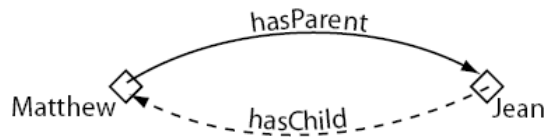
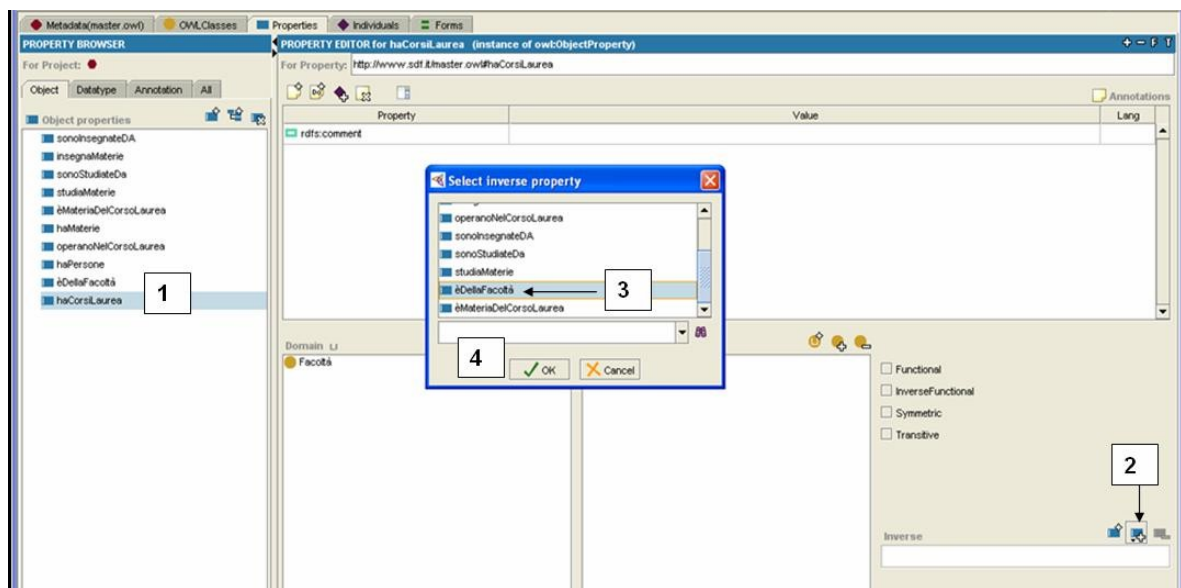
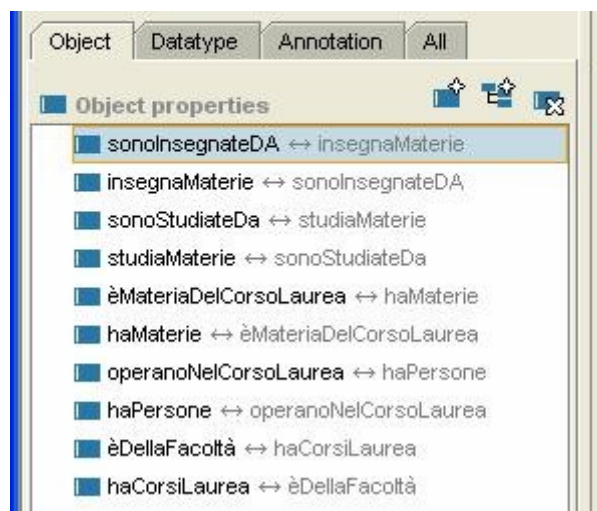


Figure 4.16: An Example Of An Inverse Property: `hasParent` has an inverse property that is `hasChild`

Per creare relazioni inverse operare nel seguente modo:



Allo stesso modo per le altre relazioni.



² A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools, Edition 1.1, pag. 28 - Matthew Horridge, Simon Jupp, Georgina Moulton, Alan Rector, Robert Stevens, Chris Wroe, The University Of Manchester, <http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf>

Creare individui

La struttura dell'ontologia è creata³, ora è possibile popolarla con individui.

Per fare ciò occorre stabilire come vogliamo strutturare i vari individui nelle diverse classi (ricordare la struttura delle relazioni nel modello relazionale di una base di dati). In pratica dobbiamo stabilire quali sono gli attributi degli individui. Dobbiamo operare attraverso diverse fasi:

1. creare un elenco di attributi,
2. associare gli attributi agli individui,
3. inserire gli individui nell'ontologia,

Creare elenco attributi e associare gli attributi agli individui.

Creare gli attributi significa creare datatype properties.

Per poter creare un elenco dobbiamo prima decidere quale sia la struttura degli individui delle varie classi:

Facoltà: cod, nome, url

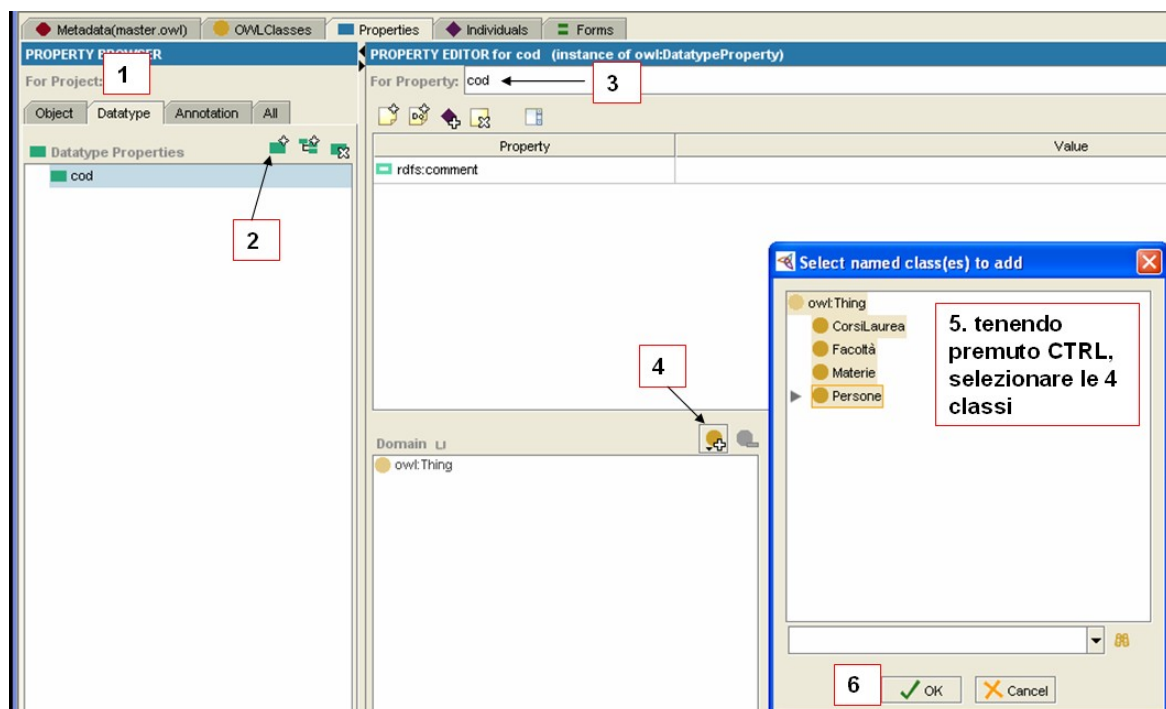
CorsiLaurea: cod, nome, url

Materie: cod, nome, url

Persone: cod, cognome, nome, media, url (le sottoclassi Docenti, Amministrativi, Studenti ereditano la struttura di Persone; la sottoclasse Studenti oltre alle datatype properties ereditate avrà anche *media*).

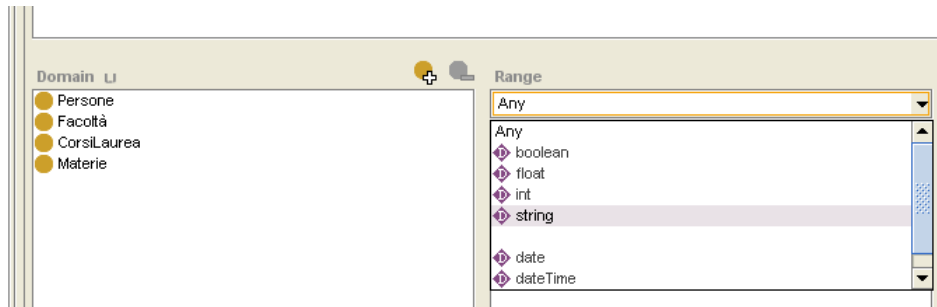
Quindi le datatype properties che dobbiamo creare sono: cod, cognome, nome, media, url.

Procedere nel seguente modo:



Dopo le fasi 1, 2, 3 premere *invio* (*enter*) e procedere con 4, 5, 6. Quindi:

³ E' possibile perfezionare l'ontologia; per approfondimenti leggere il testo citato in nota 3 e 'Ontologie OWL: Teoria e Pratica' - Prima puntata e Seconda puntata, http://www.capuano.biz/papers/CP_Ontologie_1.pdf, http://www.capuano.biz/papers/CP_Ontologie_2.pdf,

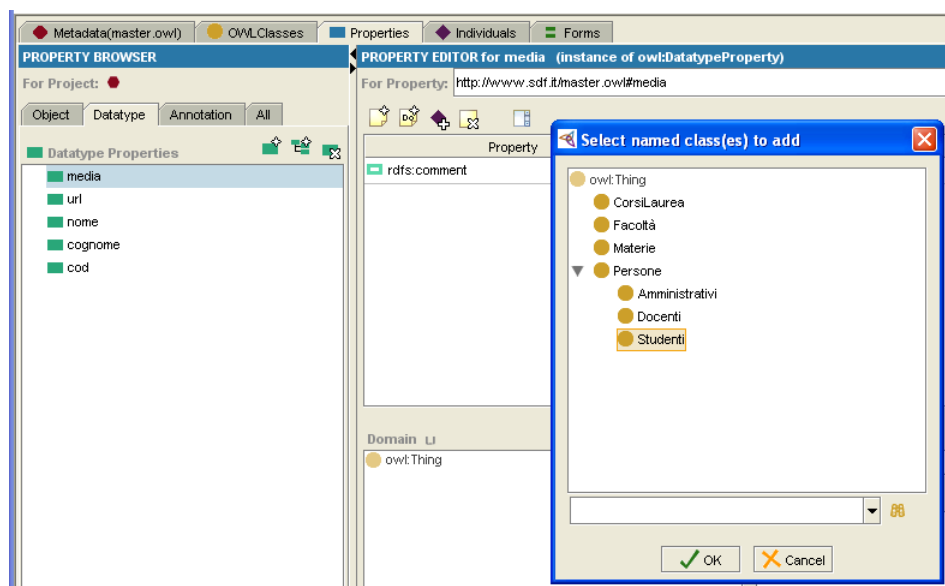


Abbiamo creato un attributo (*cod*), abbiamo stabilito il *range* ed abbiamo associato l'attributo agli individui delle classi *Persone*, *Facoltà*, *CorsiLaurea*, *Materie*.

Cognome dovrà essere associato solo a *Persone*, il *range* è *string*;

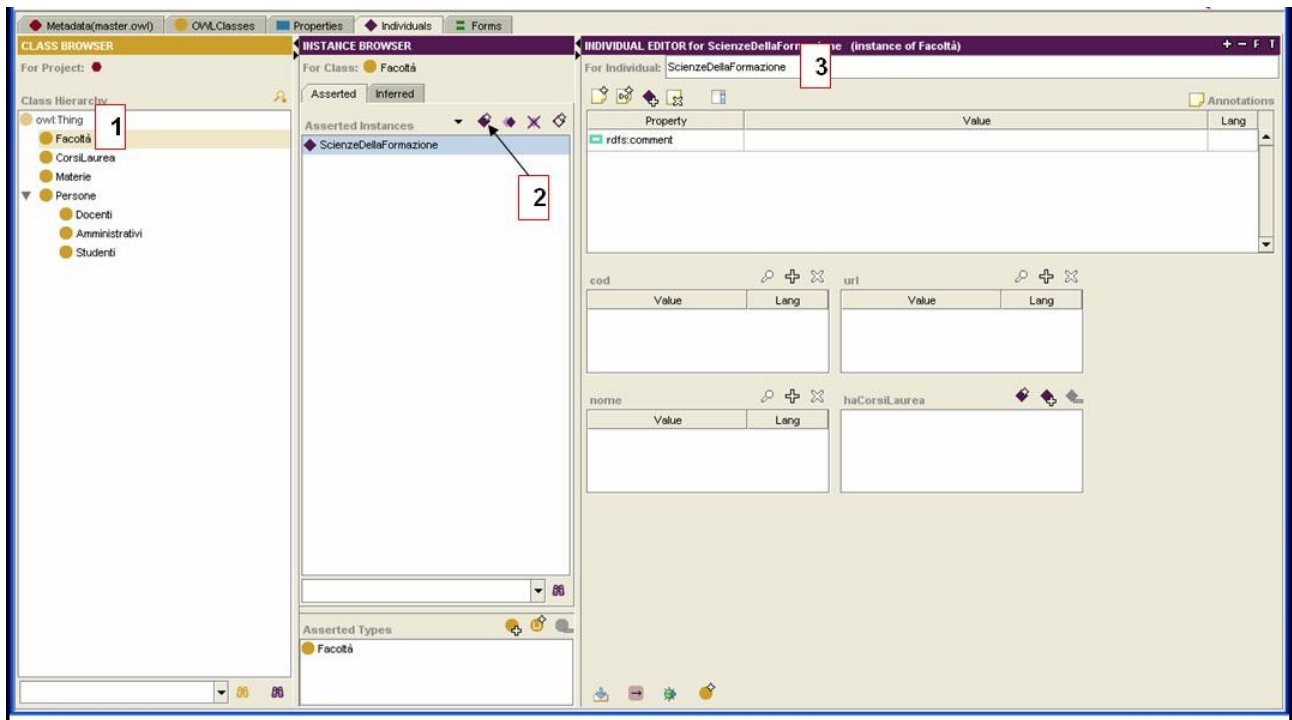
nome e *url* dovranno essere associati a tutte le classi, il *range* è *string*;

media dovrà essere associato solo a *Studenti*, il *range* è *int*.



Inserire gli individui nell'ontologia

Selezionare il tab Individuals, quindi



Notare che, automaticamente, sul lato destro vengono visualizzati gli attributi e le relazioni (in questo caso solo *haCorsiLaurea*) associati agli individui della classe di appartenenza (*Facoltà*). E' possibile valorizzare gli attributi (click sul simbolo '+'); Per inserire i corsi di laurea da associare alla Facoltà occorre cliccare sulla seconda iconcina sopra al relativo campo; in questo caso l'elenco che viene visualizzato è vuoto in quanto non sono ancora stati inseriti corsi di laurea.

Notare che il nome associato all'individuo (in Asserted Instances) appena inserito (*ScienzeDellaFormazione*) è completamente libero; è possibile inserire qualsiasi nome, è comunque preferibile inserirlo tenendo presente il contenuto (come abbiamo fatto in questo caso)

Il criterio consigliato da seguire è il seguente:

1. creare gli individui e valorizzare gli attributi;
2. per quanto riguarda le relazioni valorizzare quelle possibili, per poi tornare a completare le altre appena possibile.

Ad esempio, se provassimo ad inserire, in questo momento, un individuo della classe *Studenti* avremmo la situazione illustrata nella seguente immagine. In essa non è possibile valorizzare le relazioni evidenziate in quanto non abbiamo ancora inserito dei corsi di laurea e delle materie.

Notare che, automaticamente, vengono visualizzati gli attributi associati agli individui della classe e le relazioni che hanno per dominio la classe di appartenenza dell'individuo che si sta valorizzando. In particolare si noti l'attributo *media* che viene visualizzato in questo caso e non verrebbe visualizzato qualora si stesse valorizzando un individuo della classe *Docenti* o uno della classe *Amministrativi*.

12

Dopo aver caricato diversi individui nelle varie classi è possibile interrogare l'ontologia.

Un linguaggio per interrogare una ontologia è SPARQL; il linguaggio permette l'interrogazione di ontologie espresse in RDFS e OWL (ricordiamo che owl si basa su rdfs).

Per poter utilizzare SPARQL dall'interno di Protégé occorre:

1. selezionare *Reasoning* nel menù principale,
2. quindi Open SPARQL Query panel.

Si apre in fondo alla pagina un pannello nel quale è possibile impostare ed eseguire queries scritte in SPARQL.

SPARQL è un linguaggio abbastanza ampio ed articolato, in questo contesto lo utilizziamo impostando delle ricerche su triple <classe> <relazione> <classe> (dominio, relazione, condominio) Facciamo degli esempi per capire come impostare delle queries.

```
SELECT ?x
WHERE
{ ?x :insegnaMateria :Informatica1 }
```

Vengono selezionate tutte le triple inserite nel codice owl alla ricerca di quelle che hanno come relazione *InsegnaMateria*, come condominio *Informatica1*; queste sono relazioni fra *Docenti* e *Materie*. In pratica si ricercano tutti gli individui di *Materie* con denominazione *Informatica1* e, per ciascuna tripla individuata, viene associato ad *x* il valore dell'individuo che è in relazione con *Informatica1* attraverso *insegnaMateria*.

```
SELECT ?x
WHERE
{
    ?x :insegnaMateria :Informatica1.
    ?x :operaNelCorsoLaurea :FormazioneGestioneRisorseUmane.
}
```

restituisce chi insegna *Informatica1* nel Corso di Laurea *Formazione e Gestione delle Risorse Umane*;

Notare la possibilità di inserire più terne; in questo caso ciascuna deve terminare con un punto. *?x* deve soddisfare e la prima (*?X :insegnaMateria :Informatica1.*) e la seconda condizione (*?x :operaNelCorsoLaurea :FormazioneGestioneRisorseUmane.*).

```
SELECT ?x
WHERE {?x :haCorsoLaurea :ScienzeDellaFormazionePrimaria}
```

Restituisce la Facoltà che ha il Corso di Laurea “*Scienze Della Formazione Primaria*” (individuo chiamato *ScienzeDellaFormazionePrimaria*).

```
SELECT ?x ?y
WHERE {?x :haCorsoLaurea ?y. ?y :haPersona :AlessandriGiuseppe}
```

Restituisce il nome della Facoltà (?x) ed il nome del Corso di Laurea (?y) dove lavora Alessandri Giuseppe.

```
SELECT ?x ?y ?z  
WHERE {?x :haCorsoLaurea ?y. ?y :haPersona ?z. ?z :insegnaMateria :Informatica1}
```

Restituisce la Facoltà (?x) e il Corso di Laurea (?y) dove esiste un insegnante che insegna Informatica1 e come si chiama

La struttura di una query in SPARQL è la seguente:

SELECT *elenco di ciò che si vuole ottenere*
FROM *quale file OWL utilizziamo per la ricerca*
WHERE *condizioni che ci permettono di rintracciare ciò che stiamo cercando*

Negli esempi visti appena qui sopra, non abbiamo inserito la clausola **FROM** in quanto, automaticamente, si fa riferimento alla ontologia che si sta gestendo con Protégé

Prima di uscire da Protégé ricordarsi di salvare il progetto:

File→Save project

Assegnare un nome al progetto di Protégé (ha estensione .pprj e non è il file owl); automaticamente viene creato anche il file .owl. E' possibile selezionare la cartella dove memorizzare: cliccare sopra al campo 'Project', sulla destra, sul quadratino nero con un + sopra.

Quando, in seguito, si vorrà modificare l'ontologia si aprirà il file con estensione .owl.

NAMESPACES

This section describes namespaces, which are a general naming mechanism and are usually used to facilitate ontology importing. It then describes how to import ontologies in general.

7.2.1 Namespaces

Every ontology has its own namespace — this is known as the default namespace. An ontology may also use other namespaces. A namespace is a string of characters that prefixes the class, property and individual identifiers in an ontology. By maintaining different namespaces for different ontologies it is possible for one ontology to reference classes, properties and individuals in another ontology in an unambiguous manner and without causing name clashes. For example, all OWL ontologies (including the Pizza ontology developed in this tutorial) reference the class `owl:Thing`. This class resides in the OWL vocabulary ontology that has the namespace `http://www.w3.org/2002/07/owl#`.

In order to ensure that namespaces are unique they manifest themselves as Unique Resource Identifiers 100 (URIs)¹ ending in either `'/'` or `'#'`. For example, the default namespace in Protégé-OWL (the namespace that is assigned to newly created ontologies in Protégé-OWL) is `http://a.com/ontology#`. This means that all identifiers for classes, properties and individuals that are created in Protégé-OWL (by de-default) are prefixed with `http://a.com/ontology#`. For example, the full name for the class `PizzaTopping` is `http://a.com/ontology#PizzaTopping`. The full name for the class `MargheritaPizza` is `http://a.com/ontology#MargheritaPizza`.

Fortunately, Protégé-OWL hides these namespace prefixes which means that we don't have to type in these long winded names every time we want to use a class, property or individual identifier. Namespaces help to avoid name clashes when one ontology references classes, properties and individuals in another ontology. For example, suppose an ontology about aircraft, `AircraftOntology` has a class named `Wing`, which describes the wing of an aeroplane. An ontology about birds, `BirdOntology` also has a class named `Wing`, which describes the wing of a bird. The namespace for the `AircraftOntology` is `http://www.ontologies.com/aircraft#`. The namespace for the `BirdOntology` is `http://www.birds.com/ontologies/BirdOntology#`.

Evidently, the `Wing` class in the `AircraftOntology` is not the same as the `Wing` class in the `BirdOntology`.

Now suppose that the `AircraftOntology` imports the `BirdOntology`. Because of the namespace mechanism, the full name for the `Wing` class in the `AircraftOntology` is `http://www.ontologies.com/aircraft#Wing`. The full name for the `Wing` class in the `BirdOntology` is `http://www.birds.com/ontologies/BirdOntology#Wing`.

Hence, when the `AircraftOntology` refers to classes in the `BirdOntology` no name clash will occur. Note that neither of the above namespace URIs necessarily have to be URLs i.e. they don't necessarily have to have a physical location (on the web) — URIs are used because they ensure Uniqueness.

In order to make referencing classes, properties and individuals more manageable when using multiple namespaces, namespace prefixes are used. A namespace prefix is a short string, usually a sequence of around two or three characters that represents a full namespace. For example, we could use `"ac"` to represent the above 'aircraft ontology' namespace `http://www.ontologies.com/aircraft#` and the prefix `"bird"` to represent the 'bird ontology' namespace `http://www.birds.com/ontologies/BirdOntology#`. When we now use identifiers such as class names, we prefix the identifier with the namespace prefix and a colon.

For example `ac:Wing` or `bird:Wing`.

For a given ontology, the default namespace is the namespace for that ontology — in Protégé-OWL the default namespace corresponds to the namespace of the ontology that is being edited. When using identifiers that belong to the default namespace (the ontology being edited) a

namespace prefix is not used — classes, properties and individuals are simply referenced using their ‘local’ name. However, for imported ontologies we must use a namespace prefix to refer to classes, properties and individuals in the imported ontology. For example, suppose we were editing the ‘aircraft ontology’, which has a namespace of <http://www.ontologies.com/aircraft#> and we wanted to refer to classes in the ‘bird ontology’ with the namespace of <http://www.birds.com/ontologies/BirdOntology#> and the namespace prefix of “bird”. When we refer to classes without a namespace prefix, for example Wing, we are talking about classes in the aircraft ontology. When we refer to classes with a namespace prefix ‘bird’, for example bird:Wing, we are talking about classes in the bird ontology.