

---

# Ontology Engineering

**Seminar 2010 / 2011**  
**Dec 10, 2010**

**Katharina Siorpaes**

- The course is structured in 3 blocks.
  - Di 12.10.2010 14.00 - 16.00 3W03
  - **Fr 10.12.2010 08.00 - 18.00 3W03**
  - Fr 28.01.2011 08.00 - 18.00 3W03
- First session: experiment, topic assignment.
- **Second session: presentation of theoretical topics, submission of short papers, ontology project assignment.**
- Third session: ontology project assignment final presentations (teams).
- **Send slides of the 2nd session to [carmen.brenner@sti2.at](mailto:carmen.brenner@sti2.at) providing title, your name(s) and PDF**

# Assignment: Linked Open Innsbruck

- Make Innsbruck and relevant data a part of the LOD cloud
- Imagine various services on top of that data, such as events close to a certain location combined with transportation information or restaurants recommended by friends etc.
- Choose one area and build the vocabulary for this.
- However: keep the data in mind! Think about possible data sources that are already available.
- Develop at least one usage scenario for your ontology.

# Assignment: Linked Open Innsbruck

- Build a team of 4
- Choose topic + send me an email with group members + topic (first come, first serve)
- Download + install Protégé OWL (4.x?) (alternatively you can set up collaborative Protégé) (<http://protege.stanford.edu>)
- Build ontology collaboratively
- Document well!
- Competency questions! (ALSO for evaluation)
- Instances!
- Visualize!
- Presentation incl. ontology project, online version of ontology
  - Explain your choice of tool
  - Visualize ontology (MEANINGFUL!)
  - Explain at least 1 usage scenario
  - Explain which data sources could be used for creating this part of the data set

## Mandatory reading

- Read about Ontology 101, Stanford University (Google)
- Christian Bizer, Tom Heath and Tim Berners-Lee. Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems, Special Issue on Linked Data.

- Building vocabularies to present Innsbruck in the LOD cloud
  - Accommodation
  - Transport
  - Gastronomy
  - Sight seeing
  - Outdoor activities
  - Other?

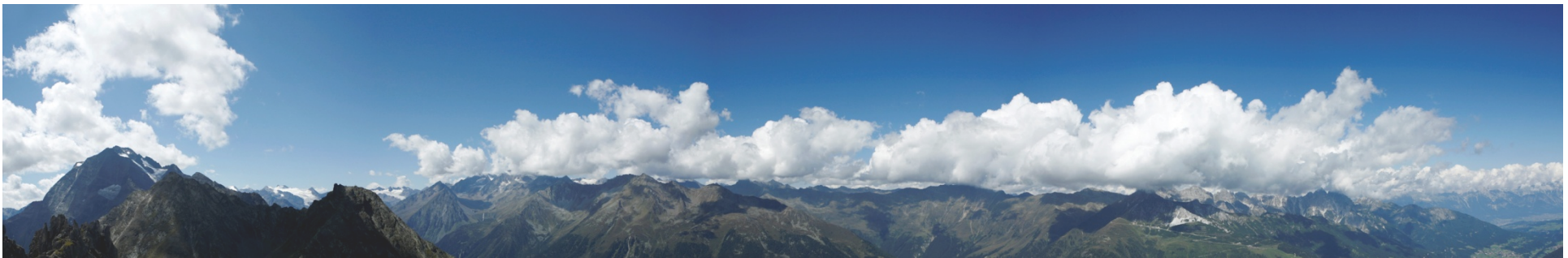


STI · INNSBRUCK

---

# How to build an ontology

[Natalya F. Noy](#) and [Deborah L. McGuinness](#). "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report [KSL-01-05](#) and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.



## Step 1: Determine the domain and scope of the ontology

---

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers?
- Who will use and maintain the ontology?



- A set of queries which place demands on the underlying ontology.
- Ontology must be able to represent the questions using its terminology and the answers based on the axioms
- Ideally, in a staged manner, where consequent questions require the input from the preceding ones.
- A rationale for each competency question should be given.

## Step 2: Consider reusing existing ontologies

- Reuse ensures interoperability and reduces costs
- VOCAMP initiative
- Ontology libraries and tools for customization are required for this step
- Sub-steps
  - Discover potential reuse candidates
  - Evaluate their usability
  - Customize ontologies to be reused
  - Integrate and merge to the target ontology
- Dr. Watson (Open University, Knowledge Media Institute)
- SWOOGLE

## Step 3: Enumerate important terms in the ontology

---

- What are the terms we would like to talk about?
- What properties do those terms have?
- What would we like to say about those terms?

## Step 4: Define classes and class hierarchy

- A top-down development process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts.
- A bottom-up development process starts with the definition of the most specific classes, the leaves of the hierarchy, with subsequent grouping of these classes into more general concepts.
- Middle-out approach: define the more salient concepts first and then generalize and specialize them appropriately.

## Step 4: Define classes and class hierarchy (ii)

- From the list created in Step 3, select the terms that describe objects having independent existence rather than terms that describe these objects.
  - These terms will be classes in the ontology.
- Organize the classes into a hierarchical taxonomy by asking if by being an instance of one class, the object will necessarily (i.e., by definition) be an instance of some other class.
  - *If a class A is a superclass of class B, then every instance of B is also an instance of A.*
- Classes as unary predicates—questions that have one argument. For example, “Is this object a wine?”
  - Later: binary predicates (or slots)—questions that have two arguments. For example, “Is the flavor of this object strong?” “What is the flavor of this object?”

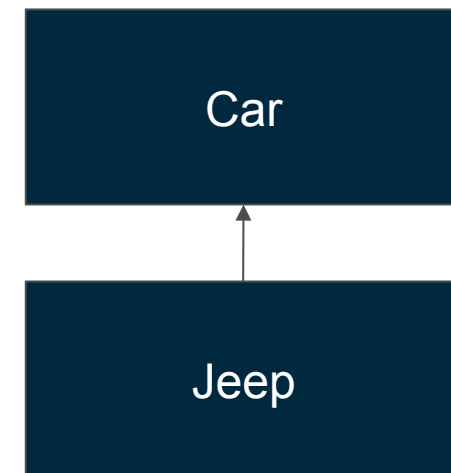
- Concepts ( classes)
- Datatype properties (string, int etc.)
  - Domain and range
  - Person hasBirthdate int
- Object properties (2 classes)
  - Person isMarried Person
- Instances
- Cardinality
- Templates

- Interview: talk to subject matter experts.
- Documentation: read what experts have written about the subject matter, read the requirements documentation, read proposals and invitations to tender.
- Observation and reflection.
- Typical candidates for classes: NOUNS.
  - But: actors of use cases do not necessarily correspond to classes.
- Other terms as well
  - Verbs: an association which starts to take on attributes and associations of its own turns into an entity: „Officer arrests suspect“.
  - Verbs: events: „Illness episode“.
  - Passive form: re-formulate in active form.

- A subclass of a class represents a concept that is a “kind of” the concept that the superclass represents.
- Classes represent concepts in the domain and not the words that denote these concepts. Synonyms for the same concept do not represent different classes.
- All the siblings in the hierarchy (except for the ones at the root) must be at the same level of generality.
- If a class has only one direct subclass there may be a modeling problem or the ontology is not complete.
- If there are more than a dozen subclasses for a given class then additional intermediate categories may be necessary.
- Subclasses of a class usually (1) have additional properties that the superclass does not have, or (2) restrictions different from those of the superclass, or (3) participate in different relationships than the superclasses.



# Examples



## Step 5: Define attributes and relationships



STI · INNSBRUCK

- Step 4 selected classes from the list of terms we created in Step 3.
  - Most of the remaining terms are likely to be properties of these classes.
  - For each property in the list, we must determine which class it describes.
- Types of properties
  - Attributes (Data properties, OWL )
  - Relationships (Object properties, OWL)
- Properties are inherited and should be attached to the most general class in the hierarchy.

- Two types of principal characteristics:
  - Measurable properties: attributes.
  - Inter-class connections: relationships.
  
- Color of an image as attribute vs. class.

- Interview: talk to subject matter experts.
- Documentation: read what experts have written about the subject matter, read the requirements documentation, read proposals and invitations to tender.
- Observation and reflection.
- Nouns in „-ness“
  - Velocity-ness, job-ness, arrested-ness...
- „How much, how many“ test.
  - If you evaluate this, then it is probably an attribute.
  - If you enumerate these, it is probably an entity.

- Are defined on sets of instances.
- Properties: reflexivity, cardinality, many-to-many, all values from, some values of, transitivity, symmetry etc.
- Arity.

- Interview: talk to subject matter experts.
- Documentation: read what experts have written about the subject matter, read the requirements documentation, read proposals and invitations to tender.
- Observation and reflection.
- Verbs, verbal phrases and things that could have been verbs.
  - „*The butler murdered the duchess*“

## Step 6: Define the restrictions of the properties

- Refine the semantics of the properties
  - Cardinality
  - Domain and range
    - When defining a domain or a range for a slot, find the most general classes or class that can be respectively the domain or the range for the slots .
    - Do not define a domain and range that is overly general
- Example
  - Class Dog
  - Attribute Age
  - Relationship hasOwner
  - Dog hasAge Integer
  - Dog hasOwner Person
  - Domain: Dog in both cases
  - Range: XML data type: integer; Class Person

## Step 7: Create instances

- Define an individual instance of a class requires
  - choose a class
  - create an individual instance of that class
  - filling in the values of the properties





STI · INNSBRUCK

---

# Protege and Collaborative Protege



- Protege is a free, open-source platform to construct domain models and knowledge-based applications with ontologies.
- Go to <http://protege.stanford.edu/doc/owl/getting-started.html> to download Protege (version 4.x)

\* Based on the Protege OWL Tutorial on the Protege website

- Protege supports the following ways of OE:
  - Frame-based
  - OWL
- Protege Frames editor: enables users to build and populate ontologies that are frame-based, in accordance with OKBC (Open Knowledge Base Connectivity Protocol).
  - Classes
  - Slots for properties and relationships
  - Instances for class
- Protege OWL editor:
  - Classes
  - Properties
  - Instances
  - Axioms

# Interface



The screenshot shows the Protégé 3.1 interface with the CLASS EDITOR for the 'Country' class. The interface includes a menu bar (File, Edit, Project, OWL, Code, Window, Help), a toolbar, and a main workspace divided into several panes.

**Subclass Relationship Pane:** Shows the asserted hierarchy for the project. The hierarchy is: owl:Thing (parent) -> Area (child) -> Country (child of Area) -> City (child of Area). The 'Country' class is currently selected.

**Class Editor Pane:** Shows the details for the 'Country' class (instance of rdfs:Class). It includes a 'Name' field containing 'Country', an 'rdfs:comment' field, and an 'Annotations' table.

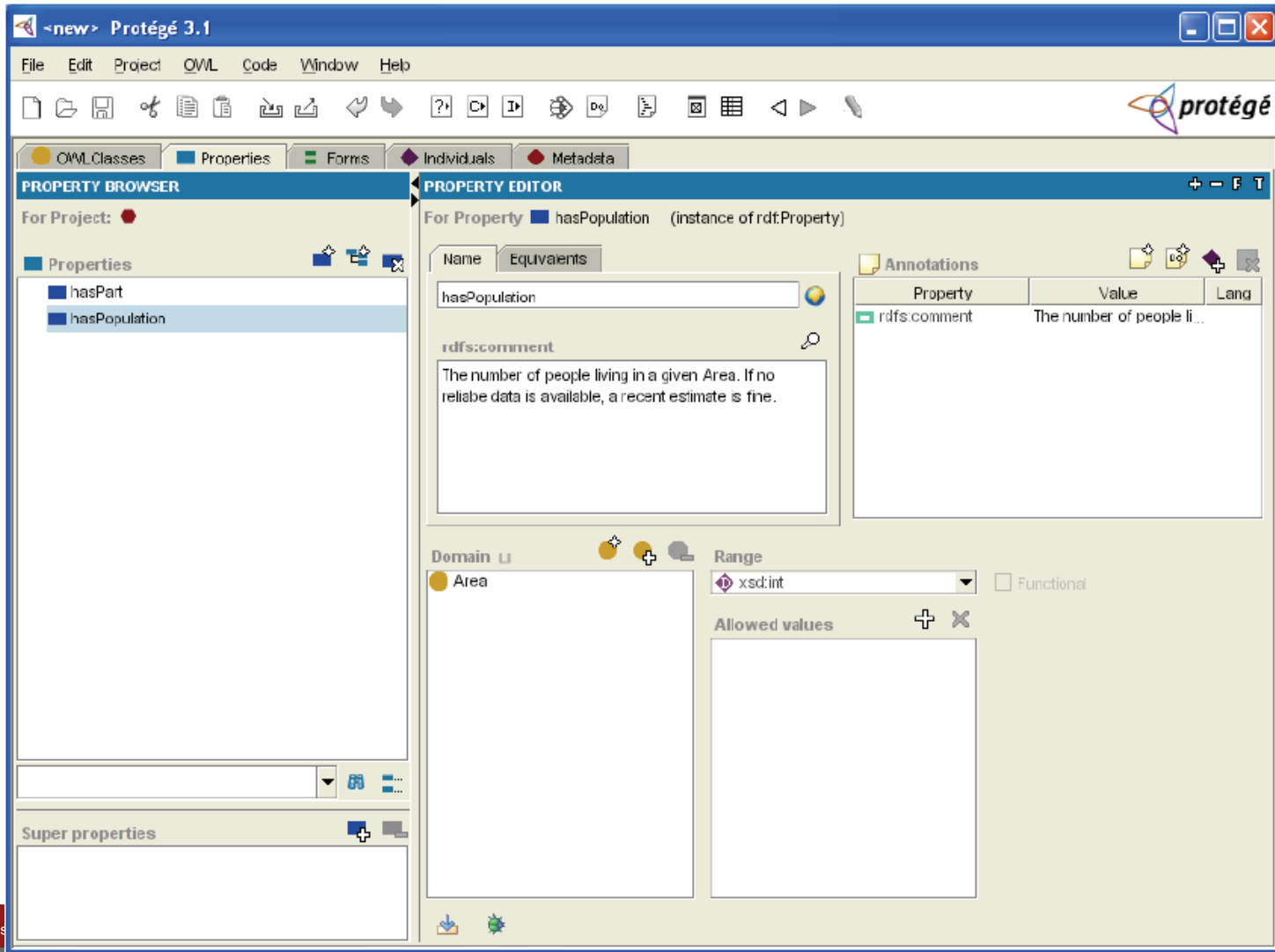
Property	Value	Lang
----------	-------	------

**SubclassOf Pane:** Shows the 'rdfs:subClassOf' relationship. The 'Country' class is a subclass of the 'Area' class.

**Properties Pane:** Shows the properties of the 'Country' class in a table.

Property	Cardinality	Type
hasPart	Multiple	Area
hasPopulation	Multiple	xsd:int

# Properties tab

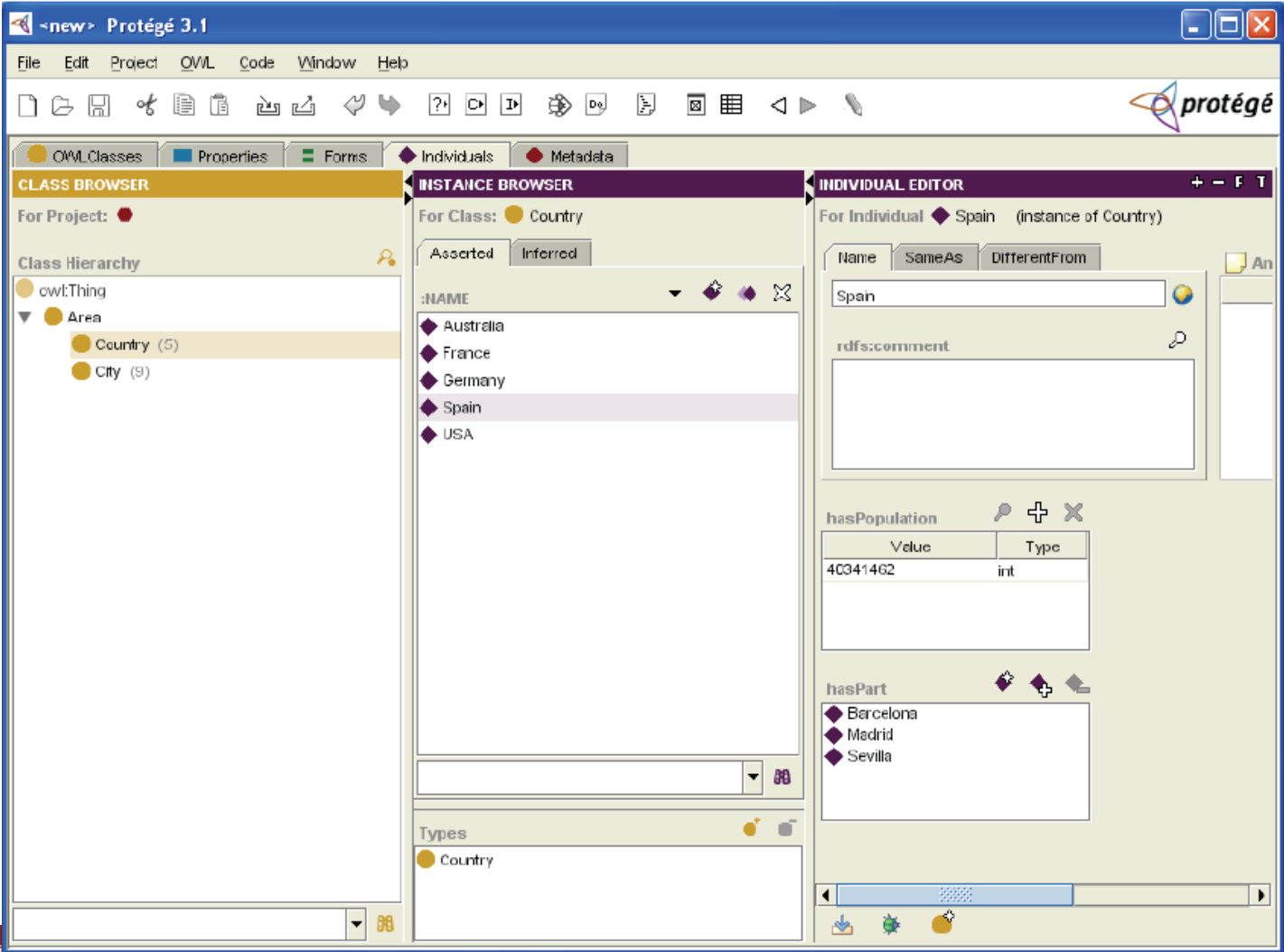


The screenshot shows the Protégé 3.1 interface with the Properties tab selected. The main window is titled "Protégé 3.1" and contains a menu bar (File, Edit, Project, OWL, Code, Window, Help) and a toolbar. The interface is divided into several panes:

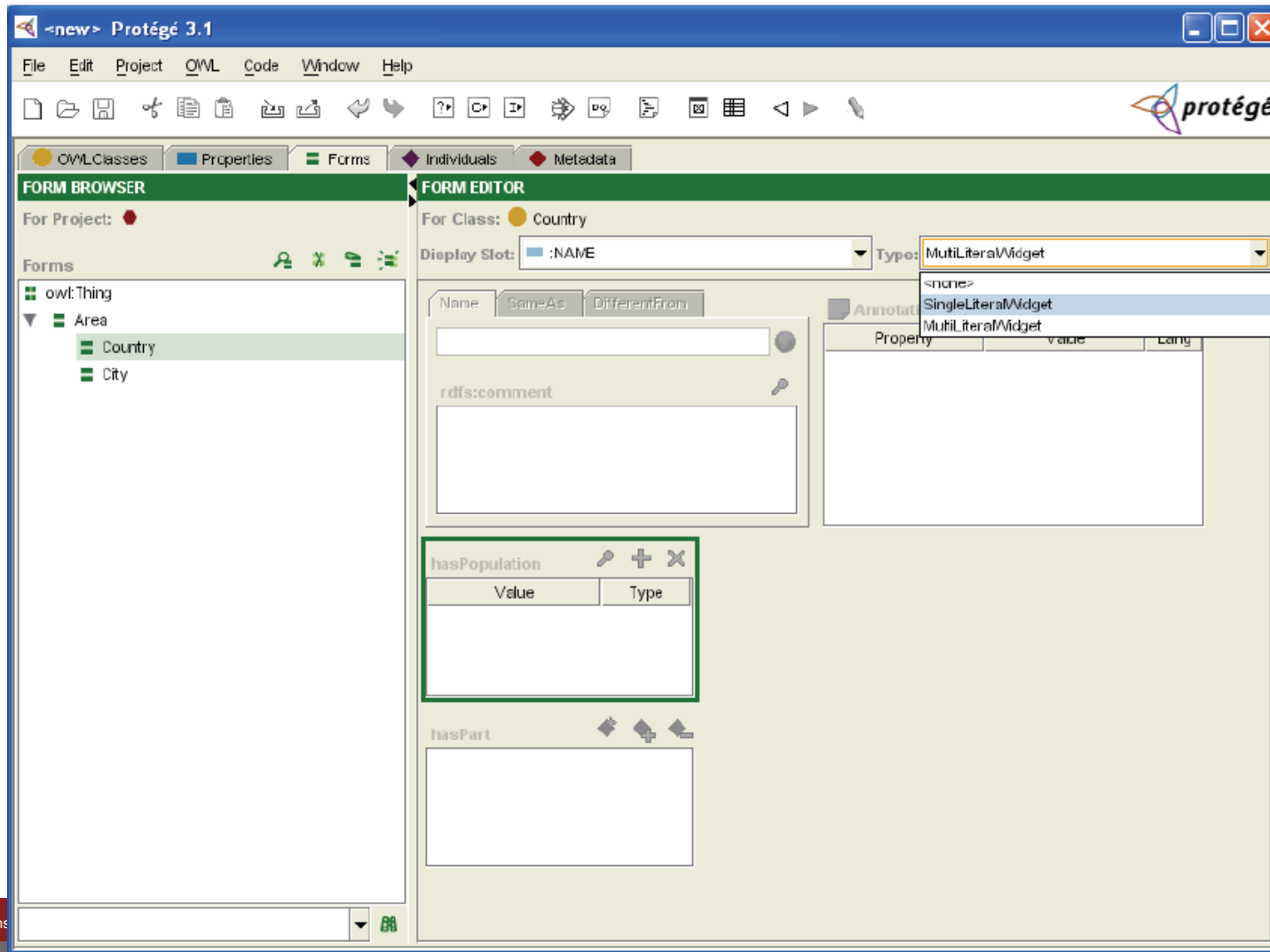
- PROPERTY BROWSER:** Shows a list of properties for the current project, including "hasPart" and "hasPopulation".
- PROPERTY EDITOR:** Displays the configuration for the selected property, "hasPopulation". It includes:
  - Name:** "hasPopulation"
  - Equivalents:** A search field for equivalent properties.
  - Annotations:** A table showing annotations for the property.
  - Domain:** A list of classes that the property applies to, currently showing "Area".
  - Range:** A dropdown menu showing the data type, currently "xsd:int".
  - Allowed values:** A list of specific values that the property can take.
  - Functional:** A checkbox to indicate if the property is functional.
- Annotations Table:** A table with columns for Property, Value, and Lang. It shows one annotation: "rdfs:comment" with the value "The number of people li...".
- Super properties:** A list of properties that are super-properties of the selected property.

- Object properties
  - Domain: Class
  - Range: Class
  - E.g.: Dog hasOwner Person
- Datatype properties
  - Domain: Class
  - Range: Datatype
  - E.g. Dog hasBirthyear integer
- Property hierarchies

# Individuals tab



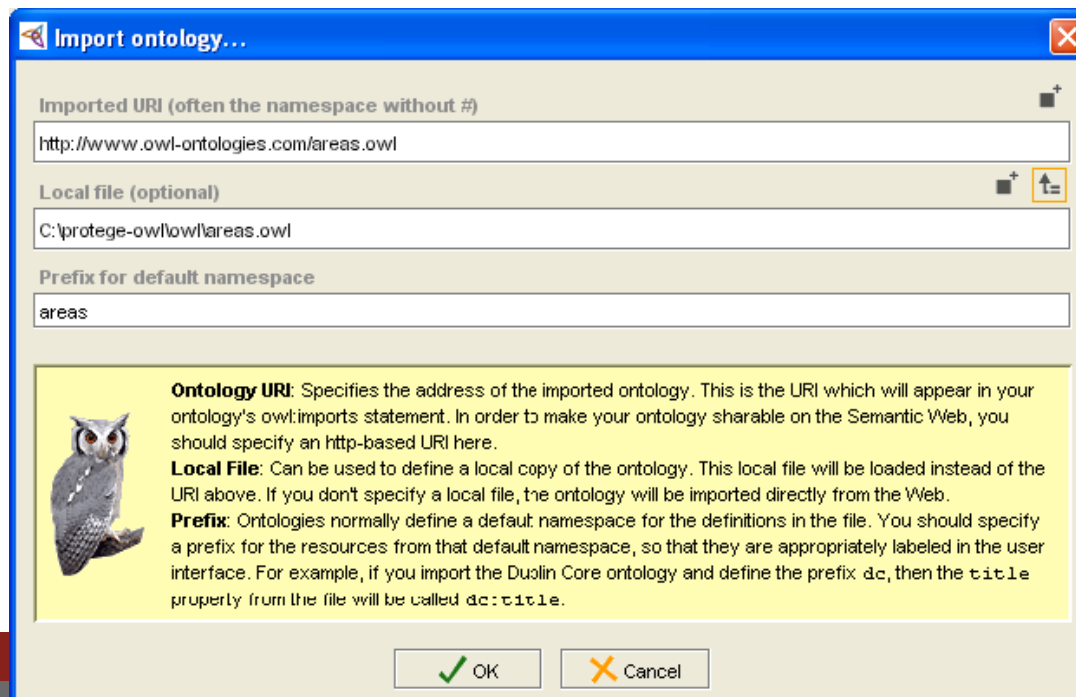
# Forms tab





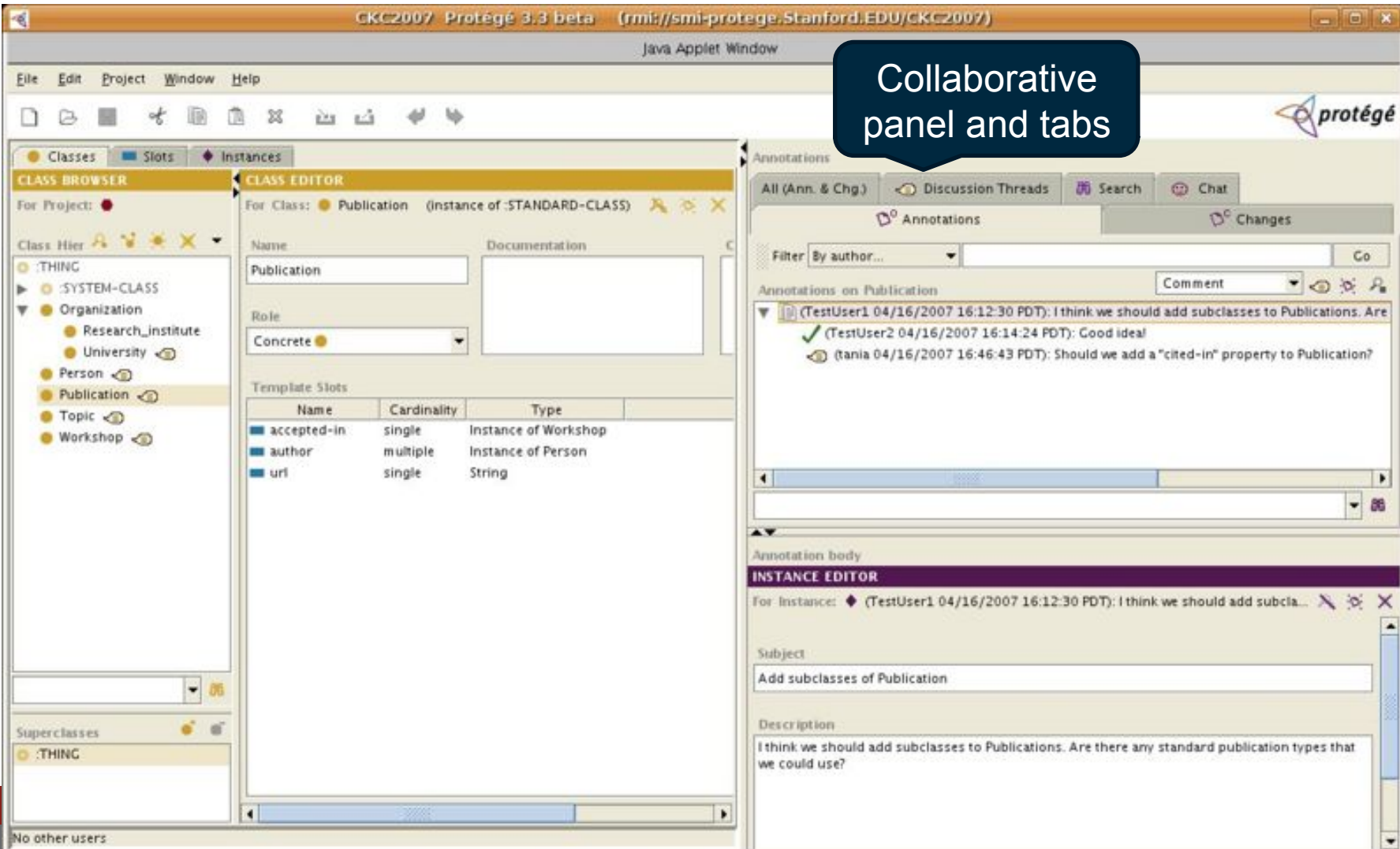
# Importing ontologies

- Re-use of ontologies
- Local or on the Web



- Extension of Protege
- Supports:
  - Annotating ontology components and changes in the ontology
  - Discussion threads (live chat)
  - Proposal and voting
  - Searching and filtering
  - Defining users, groups, policies
- Works with Protege OWL and Frames
- <http://protege.stanford.edu/doc/collab-protege/>

# Collaborative Protege



The screenshot displays the Protege 3.3 beta interface. A dark blue callout box with white text points to the right-hand side of the window, stating "Collaborative panel and tabs".

The interface includes several panels:

- CLASS BROWSER:** Shows a class hierarchy for the project. The "Publication" class is selected.
- CLASS EDITOR:** Shows the details for the "Publication" class, including its name, role (set to "Concrete"), and template slots.
- Annotations:** A panel on the right showing a list of annotations for the "Publication" class. It includes a filter "By author...", a "Go" button, and a list of annotations with their authors and timestamps. The first annotation is "(TestUser1 04/16/2007 16:12:30 PDT): I think we should add subclasses to Publications. Are".
- INSTANCE EDITOR:** A panel at the bottom right showing the details for the selected annotation instance. It includes a "Subject" field with the text "Add subclasses of Publication" and a "Description" field with the text "I think we should add subclasses to Publications. Are there any standard publication types that we could use?".

Name	Cardinality	Type
accepted-in	single	Instance of Workshop
author	multiple	Instance of Person
url	single	String