

# **Ontology Editors**

# IDEs for Ontologies

- Some people use simple text editors
  - Working with XML serialization will drive you crazy
  - Using Turtle or an abstract syntax works well
- Others prefer an IDE
  - Good IDEs include support for reasoning, visualization, and more
- Protégé is a very popular IDE
  - From Stanford, free, lots of plugins
- TopQuadrant [Composer](#) is also good
  - Feature rich but expensive (\$600 for a single license)

# Protégé 5.1

The screenshot displays the Protégé 5.1 web interface. At the top, the browser address bar shows the URL `http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122`. Below the address bar, a search bar contains the text `untitled-ontology-122`. The main interface is divided into several sections:

- Active Ontology:** A tabbed menu with options: `Active Ontology`, `Entities`, `Classes`, `Object Properties`, `Data Properties`, `Individuals by class`, `OWL Viz`, and `DL Query`.
- Annotations:** A tabbed menu with options: `Annotations`, `Selected entailments`, and `Rules`.
- Ontology header:** A purple header bar containing:
  - Ontology IRI:** `http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122`
  - Ontology Version IRI:** `e.g. http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122/1.0.0`
- Annotations:** A large white area with a `+` icon, currently empty.
- Ontology imports:** A tabbed menu with options: `Ontology imports`, `General axioms`, `RDF/XML rendering`, `OWL/XML rendering`, and `OWL functional syntax rendering`.
- Imported ontologies:** A purple header bar containing:
  - Direct Imports:** A white area with a `+` icon, currently empty.
  - Indirect Imports:** A white area, currently empty.

At the bottom of the interface, a status bar contains the text: `To use the reasoner click Reasoner > Start reasoner` and a checked checkbox labeled `Show Inferences`.

# Protégé 5.2

- <http://protege.stanford.edu/>
- Free, open source ontology editor and KB framework
- Predates OWL, still supports earlier Frames representation
- In Java, extensible, large community of users
- Desktop and Web versions
  - Works will under Linux, Mac OS X and Windows

# Desktop Protégé

The screenshot displays the Protégé desktop application interface for editing the 'peeps' ontology. The browser address bar shows the URL `http://ebiq.org/ontologies/peeps/`. The main window is titled 'peeps (http://ebiq.org/ontologies/peeps/)' and contains several tabs: 'Active Ontology', 'Entities', 'Individuals by class', 'DL Query', 'Individual Hierarchy Tab', and 'OWLviz'. The 'Class hierarchy' tab is active, showing a tree view of classes: `owl:Thing` (parent), `Person` (child), `Adult` (child of Person), `Man` (child of Person, highlighted in blue), `Boy` (child of Man), `Minor` (child of Person), `Boy` (child of Minor), and `Woman` (child of Person). The 'Annotations: Man' panel shows the following annotations:

- `rdfs:label`: Male person
- `rdfs:comment`: A Man is defined as a person with a has sex value equal to "male"

The 'Description: Man' panel shows the following description:

- Equivalent To: `Person and (hasSex value "male")`
- SubClass Of: `Person`
- General class axioms:
  - `hasParent exactly 1 Man`
  - `hasParent exactly 1 Woman`

The 'Data proper' panel shows the `owl:topDataProperty`. The bottom status bar indicates 'To use the reasoner click Reasoner > Start reasoner' and 'Show Inferences' is checked.

# Web Protégé

The screenshot shows the WebProtégé web interface in a browser window. The browser's address bar displays `webprotege.stanford.edu/#Edit:...`. The interface includes a top navigation bar with the Protégé logo, a 'Project' dropdown menu, 'Share' and 'Help' buttons, and a user profile for 'Tim Finin'. Below this is a tabbed interface with the active tab labeled 'UMBC691PeepsExample'. A secondary navigation bar contains tabs for 'Classes', 'Properties', 'Individuals', 'Notes and Discussions', 'Changes By Entity', and 'Project Dashboard'. A toolbar at the bottom of this bar offers 'Add content to this tab' and 'Add tab' options.

The main workspace is divided into three panels:

- Classes Panel:** Shows a hierarchical tree of classes. The root is 'owl:Thing', with sub-classes 'Person' and 'Sex'. The 'Person' class is currently selected.
- Class description for Person Panel:** Contains the following fields:
  - Display name:** 'Person'
  - IRI:** `http://webprotege.stanford.edu/RCcwwdIsdJMKB`
  - Annotations:** A table with one entry: 

rdfs:label	Person	lang
------------	--------	------

 Below this is an input field for adding more annotations with 'Enter property' and 'Enter' buttons.
  - Properties:** An input field for adding properties with 'Enter property', 'Ent', and 'lang' buttons.
- Discussions for Person Panel:** Features a 'Post new topic...' button.

# YAS: Yet Another Syntax

- Neither OWL's official abstract syntax nor XML serialization is easy to read or use
- Protégé uses the Manchester syntax
- Simpler and more compact: “some” and “only”, not “someValuesFrom” and “allValuesFrom”
- A W3C recommendation (<http://bit.ly/manSyn>), used in the OWL 2 Primer (<http://bit.ly/OWL2Pri>)

Class: man

Annotations: rdfs:label "man"

EquivalentTo: adult and male and person

# Manchester OWL syntax

OWL	DL Symbol	Manchester OWL Syntax Keyword	Example
someValuesFrom	$\exists$	<b>some</b>	hasChild <b>some</b> Man
allValuesFrom	$\forall$	<b>only</b>	hasSibling <b>only</b> Woman
hasValue	$\ni$	<b>value</b>	hasCountryOfOrigin <b>value</b> England
minCardinality	$\geq$	<b>min</b>	hasChild <b>min</b> 3
cardinality	$=$	<b>exactly</b>	hasChild <b>exactly</b> 3
maxCardinality	$\leq$	<b>max</b>	hasChild <b>max</b> 3



# Manchester OWL syntax

OWL	DL Symbol	Manchester OWL Syntax Keyword	Example
intersectionOf	$\sqcap$	and	Doctor and Female
unionOf	$\sqcup$	or	Man or Woman
complementOf	$\neg$	not	not Child

# Example

```
Person and
  hasChild some
    (Person and
      (hasChild only Man) and
        (hasChild some Person) )
```

The set of people who have at least one child that has some children that are only men (i.e., grandparents that only have grandsons)

# Data values and datatypes

- Data values typed or untyped (e.g., int, boolean, float)
- Constants with or w/o type, e.g.: hasAge value "21"^^long
- Use datatype names as classes: hasAge some int
- XSD facets, e.g.: Person and hasAge some int[>= 65]

XSD facet	Meaning
< x, <= x	less than, less than or equal to x ( <a href="#">more info</a> )
> x, >= x	greater than, greater than or equal to x ( <a href="#">more info</a> )
length x	For strings, the number of characters must be equal to x ( <a href="#">more info</a> )
maxLength x	For strings, the number of characters must be less than or equal to x ( <a href="#">more info</a> )
minLength x	For strings, the number of characters must be greater than or equal to x ( <a href="#">more info</a> )
pattern regexp	The lexical representation of the value must match the regular expression, regexp ( <a href="#">more info</a> )
totalDigits x	Number can be expressed in x characters ( <a href="#">more info</a> )
fractionDigits x	Part of the number to the right of the decimal place can be expressed in x characters ( <a href="#">more info</a> )

# Demonstration

- We'll use Protégé OWL v5.2 to implement a tiny ontology for people
- Start by downloading and installing Protégé 5.2 (You will need Java)
- You may want to install Graphviz
- Configure Protégé
  - E.g., select a reasoner to use (e.g., Hermit)

# A basic workflow

- Think about usecases
- Preliminaries
  - Choose namespace URL, import other ontologies used
- Identify and define classes
  - Place in hierarchy, add **axioms** and run reasoner to check for errors or omissions
- Identify and define properties
  - Place in hierarchy, add **axioms**, run reasoner
- Add individuals & reasoner to check for problems
- Add comments and labels
- Export in desired formats, maybe upload to Web

# More workflow steps

- Use [OOPS](#) to find common ontology pitfalls
- Link concepts (and individuals) to common ontologies (e.g., DBpedia, Freebase, foaf)
  - Use owl:sameAs
- Generate visualizations
- Produce documentation
- Develop examples with your use case(s)
- Encode data, describe in [VoID](#) (Vocabulary of Interlinked Datasets), add to LOD cloud

# Demonstration/HW4

Use Protégé OWL (v5.2) to build a simple ontology for people based on the following

- People have just one sex that's either *male* or *female*, an integer age, and two parents, one male, one female
- A person's grandparent is the parent of their parent
- Every person is either a man or a woman but not both
- A man is defined as any person whose sex is male and a woman as any person whose sex is female
- A boy is defined as a person whose sex is male and whose age is less than 18, a girl is ...
- A person is either an adult or (age >18), minor (age <18), ...

# Test cases

## All Different people

Alice F

Bob M

Carol F

Don M

Edith F

Pat ?

## Other people

Frank M

Gwen F

## Some possible test cases

- Alice parent Bob . Bob parent Carol
  - Alice grandparent Carol
- Alice parent Bob . Alice parent Don.
  - Contradiction
- Alice parent Bob . Pat parent Bob
  - Pat a female
- Alice parent Bob . Gwen parent Bob .
  - Alice owl:sameAs Gwen