

# OWL Editor

## Contents


1. [The OWL Language](#)
2. [OWL 2 Language Features](#)
3. [User Interface Overview](#)
  - 3.1. [Ontology Navigator](#)
  - 3.2. [Individuals Panel](#)
  - 3.3. [Entity Properties Panel](#)
    - 3.3.1. [Ontology](#)
    - 3.3.2. [Classes](#)
    - 3.3.3. [Object Properties](#)
    - 3.3.4. [Datatype Properties](#)
    - 3.3.5. [Annotation Properties](#)
    - 3.3.6. [Datatypes](#)
    - 3.3.7. [Individuals](#)
  - 3.4. [Manchester Syntax](#)
  - 3.5. [Search](#)
  - 3.6. [Ontology Visualizer](#)
  - 3.7. [Preferences](#)
4. [How to use the OWL Editor?](#)
  - 4.1. [How to create an Ontology Project?](#)
  - 4.2. [How to create an Ontology?](#)
  - 4.3. [How to create a Class?](#)
  - 4.4. [How to create an Object Property?](#)
  - 4.5. [How to create a Datatype Property?](#)
  - 4.6. [How to reate an Annotation Property?](#)
  - 4.7. [How to reate an Annotation?](#)
  - 4.8. [How to load an Ontology?](#)
  - 4.9. [How to save an Ontology?](#)
  - 4.10. [How to delete an Entity?](#)
5. [Online References](#)

## OWL Editor

The OWL Editor of NeOn Toolkit is a modeling tool for the creation and maintenance of semantic models (often referred to as "ontologies") written in the Web Ontology Language OWL.

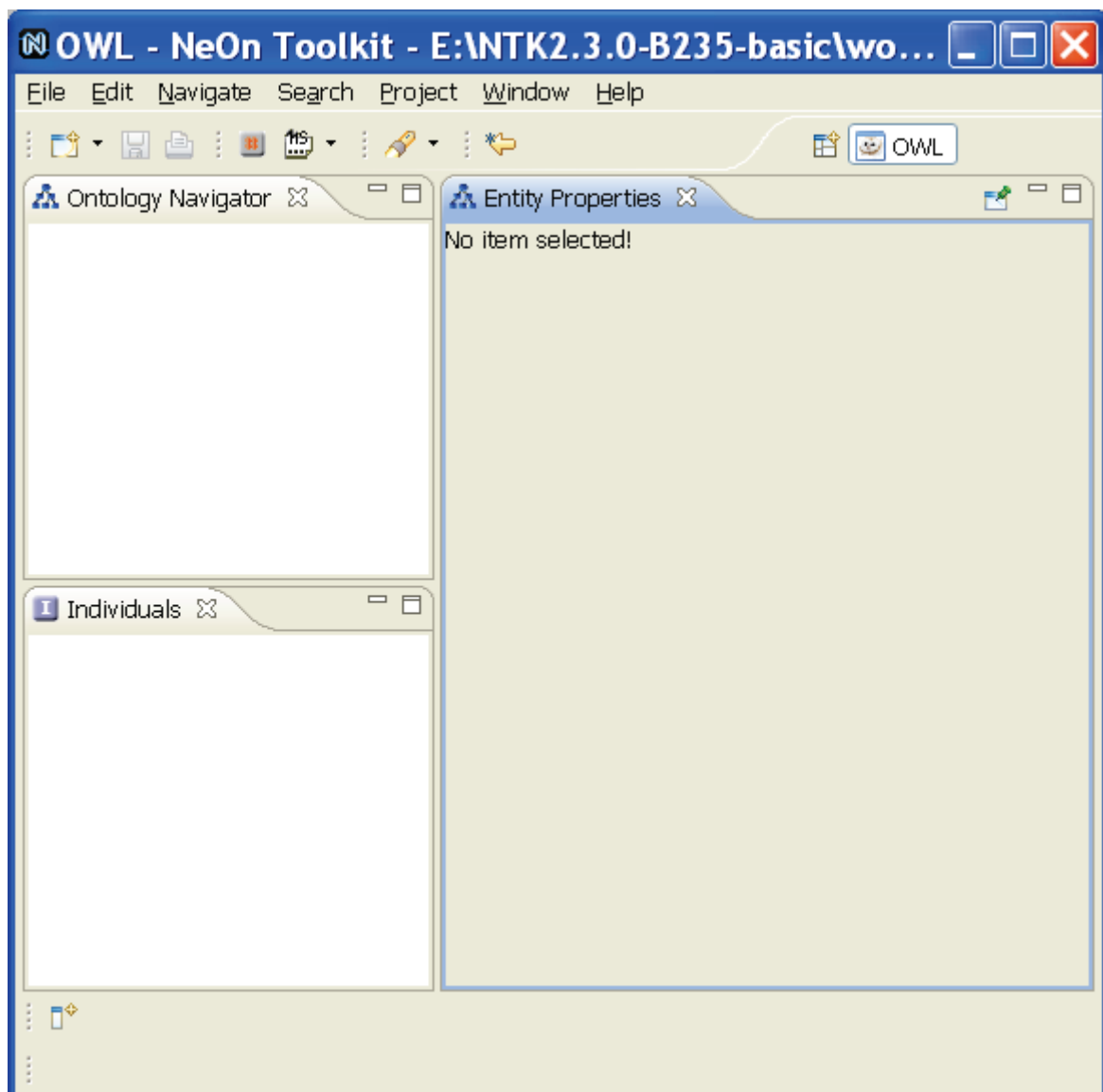
NeOn Toolkit is implemented as a set of Eclipse plugins, which means that it is built on a solid platform with a large user base. The foundation on the Eclipse plugin architecture also means that developers can build additional services on top of NeOn Toolkit.

For more information about the NeOn Toolkit, visit the [NeOn Toolkit Web site](#).

You can access the OWL Editor by selecting the OWL icon  in the upper right corner of NeOn Toolkit. Alternatively, go to Window >> Open Perspective >> Other and select OWL in the dialog box.

The OWL Editor consists of the following main panels:

- **Ontology Navigator.** The Ontology Navigator shows the ontology projects, their corresponding ontologies, and the hierarchies of the current ontology.
- **Individuals panel.** The Individuals panel displays all instances of the most recently selected class.
- **Entity Properties panel.** The Entity Properties panel is the main work area for defining and modifying resources of the selected ontology.



For more information on the user interface, see User Interface Overview.

## 1. OWL

The Web Ontology Language OWL is used for publishing and sharing ontologies on the World Wide Web, and is endorsed by the World Wide Web Consortium. OWL 2 was released as a Recommendation in October 2009. OWL is considered one of the fundamental technologies underpinning the Semantic Web and is designed for use by applications that need to process the content of information instead of just presenting information to humans. Among other things, OWL facilitates greater expressivity than XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics.

---

**Note** – Familiarity with the OWL language specification is recommended to use all features of the OWL Editor. For more information, see the [OWL 2 Web Ontology Language Primer](#). The full set of documents which represent the OWL 2 language definition can be found at the [OWL 2 Working Group page](#) the W3C.

---

This guide contains a brief and informal description of the OWL language constructs and is meant to serve as a reference for users who want to build ontologies using OWL.

The following page we give an overview over all available OWL 2 language features and indicate if and how you can find, access, and enter them in the NeOn Toolkit.

## 2. The OWL Modelling Facilities of the NeOn Toolkit

### Notational Conventions

The OWL2 Working group has published a quick reference guide [Bao et al. 2009] summarizing all language features<sup>1</sup>. In this reference guide the authors use a number of predefined namespaces and some place holders for certain kinds of entities or entity names. In the following table we replicate the used notational conventions:

Prefix Name	Expansion
rdf:	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>
rdfs:	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>
owl:	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>
xsd:	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>

**Prefixes for standard namespaces.**

Letters	Meaning
C	class expression
CN	class name
D	data range
DN	datatype name
P	object property expression
PN	object property name
R	data property
A	annotation property
a	individual
aN	individual name
_:a	anonymous individual (a blank node label)
v	literal

n	non-negative integer
f	facet
ON	ontology name
U	IRI
s	IRI or anonymous individual
t	IRI, anonymous individual, or literal
p	prefix name
_:x	blank node
(a1 ... an)	RDF list

**Variable names and their meaning**

## Supported and Unsupported OWL2 Features

In this section we will give an overview of the supported OWL2 features (including OWL1, cf. [Golbreich, Wallace 2009] for an overview and rationale for the new, additional OWL2 features). We will use the same structuring as chosen in the OWL2 reference guide [Bao et al. 2009]. The following tables present the following information:

- The actual OWL2 feature
- Functional style syntax of the feature according to the OWL2 recommendation
- Syntax for the feature as used in the NeOn Toolkit. This is based on the Manchester Syntax for OWL (if applicable)
- Is the feature supported by the GUI of the NeOn Toolkit? yes/no

### Class Expression

The first group of features represents the different ways to specify complex OWL classes.

	Functional Syntax	NeOn Toolkit Syntax	supported by GUI of NeOn Toolkit 2.3
<b>2.1 Class Expressions</b>			
<b>Predefined and named classes</b>			

named class	C	C	yes
universal class	owl:Thing	owl:Thing	yes
empty class	owl:Nothing	owl:Nothing	yes
<b>Boolean Connectives and Enumeration of Individuals</b>			
intersection	ObjectIntesectionOf(C1...Cn)	C1 and ... Cn	yes
union	ObjectUnionOf(C1...Cn)	C1 or ... Cn	yes
complement	ObjectComplementOf(C)	not C	yes
enumeration	ObjectOneOf(a1 ... an)	{a1, ... an}	yes
<b>Object Property Restrictions</b>			
universal	ObjectAllValuesFrom (P C)	P only C	yes
existential	ObjectSomeValuesFrom(P C)	P some C	yes
individual value	ObjectHasValue(P a)	P value a	yes
local reflexivity	ObjectHasSelf(P)	P Self	yes
exact cardinality	ObjectExactCardinality(n P)	P exactly n owl:Thing	yes
qualified exact cardinality	ObjectExactCardinality(n P C)	P exactly n C	yes
maximum cardinality	ObjectMaxCardinality(n P)	P max n owl:Thing	yes
qualified maximum cardinality	ObjectMaxCardinality(n P C)	P max n C	yes
minimum cardinality	ObjectMinCardinality(n P)	P min n owl:Thing	yes
qualified minimum cardinality	ObjectMinCardinality(n P C)	P min n C	yes
<b>Data Property Restrictions</b>			

universal	DataAllValuesFrom (P D)	P only D	yes
existential	DataSomeValuesFrom(P D)	P some D	yes
literal value	DataHasValue(P a)	P value a	yes
exact cardinality	DataExactCardinality(n P)	P exactly n	yes
qualified exact cardinality	DataExactCardinality(n P D)	P exactly n D	yes
maximum cardinality	DataMaxCardinality(n P)	P max n	yes
qualified maximum cardinality	DataMaxCardinality(n P D)	P max n D	yes
minimum cardinality	DataMinCardinality(n P)	P min n	yes
qualified minimum cardinality	DataMinCardinality(n P D)	P min n D	yes
<b>Restrictions Using n-ary Data Range</b>			
n-ary universal	DataAllValuesFrom (R1 ... Rn Dn)		no
n-ary existential	DataSomeValuesFrom (R1 ... Rn Dn)		no

As can be seen the NeOn Toolkit support nearly all complex class expressions, with the exception of n-ary data ranges. It also supports the new OWL2 expressions like qualified number restrictions, local reflexivity (Self) or the new built-in class owl:Nothing.

## Properties

In the second group we will all features representing property expressions.

	Functional Syntax	NeOn Toolkit Syntax	supported by GUI of NeOn Toolkit 2.3
<b>2.2 Properties</b>			
<b>Object Property Expressions</b>			
named object property	PN	PN	yes

universal object property	owl:topObjectProperty	owl:topObjectProperty	yes
empty object property	owl:bottomObjectProperty	owl:bottomObjectProperty	yes
inverse property	ObjectInverseOf(PN)		no
<b>Data Property Expressions</b>			
named Data property	R	R	yes
universal Data property	owl:topDataProperty	owl:topDataProperty	yes
empty Data property	owl:bottomDataProperty	owl:bottomDataProperty	yes

The NeOn Toolkit supports all property expressions, with the exception of inverse object properties. Users can specify inverse properties using axioms but currently cannot use them in class expressions. Nevertheless, loading and storing ontologies that make use of *ObjectInverseOf* expressions is supported by the NeOn toolkit via the OWL API datamodel, parsers and serializers. All new top and bottom properties introduced by OWL2 are supported by the NeOn Toolkit.

## Individuals and Literals

The third group describes basic individuals or datatype values (literals).

	Functional Syntax	NeOn Toolkit Syntax	supported by GUI of NeOn Toolkit 2.3
<b>2.3 Individuals and Literals</b>			
named individual	aN	aN	yes
anonymous individual	_:a		no
literal (datatype value)	"abc"^^DN	"abc"^^DN	yes

With the exception of the anonymous individuals we support all individuals and literals within the text fields of the NeOn Toolkit. The underlying datamodel does support anonymous individuals, thus the NeOn Toolkit can handle such ontologies.

## Data Ranges



Group number four describes (complex) datatypes, which are called data ranges in OWL2.

	Functional Syntax	NeOn Toolkit Syntax	supported by GUI of NeOn Toolkit 2.3
<b>2.4 Data Ranges</b>			
<b>Data Range Expressions</b>			
named datatype	DN	DN	Yes
data range complement	DataComplementOf (D)		No
data range intersection	DataIntersectionOf (D1...Dn)		No
data range union	DataUnionOf (D1...Dn)		No
literal enumeration	DataOneOf (v1 ... vn)	{v1, ... vn}	Yes
datatype restriction	DatatypeRestriction (DN f1 v1 ... fn vn)		No

With respect to complex data ranges the Neon Toolkit only supports named data types and enumerations. The new OWL2 expressions to construct new data ranges, e.g. via facets is not yet supported by the GUI.

## Class Axioms

The next group starts a long list of axioms which can be asserted in OWL2. In the NeOn toolkit axioms are accessible through different GUI mechanisms instead of being represented in pure textual form. Because of this, from now on, we leave out the NeOn Toolkit syntax column.

Class axioms make statements about the relationship between OWL classes.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI of NeOn Toolkit 2.3
<b>2.5 Axioms</b>			
<b>Class Expression Axioms</b>			
subclass	SubClassOf(C1 C2)	Taxonomy Tab of Class, resp. Restrictions Tab if C2 is a	Yes
equivalent classes	EquivalentClasses (C1 ... Cn)		Yes

		restriction	
disjoint classes	DisjointClasses(C1 C2)	Taxonomy Tab of Class	Yes
pairwise disjoint classes	DisjointClasses (C1 ... Cn)		Yes
disjoint union	DisjointUnionOf (CN C1 ... Cn)		No

Except for the new OWL2 feature of specifying a class as a disjoint union of its subclasses the NeOn Toolkit supports all class axioms. They can be entered via the entity property view for classes in separate sections or tabs (see below).

## Object Property Axioms

In the following group we list a rather extensive set of different axioms that specify the semantics of object properties OWL2.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>Object Property Axioms</b>			
subproperty	SubObjectPropertyOf (P1 P2)	Taxonomy tab of Object Property	Yes
property chain inclusion	SubObjectPropertyOf (ObjectPropertyChain (P1 ... Pn) P)		Yes
property domain	ObjectPropertyDomain (P C)	Domain/Range Tab of Object Property	Yes
property range	ObjectPropertyRange (P C)		Yes
equivalent properties	EquivalentObjectProperties (P1 ... Pn)	Taxonomy tab of Object Property	Yes
disjoint properties	DisjointObjectProperties (P1 P2)		No
pairwise disjoint properties	DisjointObjectProperties (P1 ... Pn)		No
inverse properties	InverseObjectProperties (P1 P2)	Taxonomy tab of Object Property	Yes
functional property	FunctionalObjectProperty (P)	Domain/Range Tab of Object Property	Yes
inverse functional property	InverseFunctionalObjectProperty (P)		Yes
reflexive property	ReflexiveObjectProperty (P)		Yes
irreflexive property	IrreflexiveObjectProperty (P)		Yes
symmetric property	SymmetricObjectProperty (P)		Yes
asymmetric property	AsymmetricObjectProperty (P)		Yes

transitive property	TransitiveObjectProperty (P)	Yes
------------------------	------------------------------	-----

With the sole exception of disjoint properties the NeOn toolkit supports all object property axioms, including the new OWL2 features such as property chains, reflexive, irreflexive or asymmetric properties.

## Data Property Axioms

The next group shows the data property axioms, which allow to specify the semantics of data properties OWL2.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>Data Property Axioms</b>			
subproperty	SubDataPropertyOf(R1 R2)	Taxonomy Tab of Data Property	Yes
property domain	DataPropertyDomain(R C)	Domain/Range Tab of Data Property	Yes
property range	DataPropertyRange(R D)		Yes
equivalent properties	EquivalentDataProperties (R1 ... Rn)	Taxonomy Tab of Data Property	Yes
disjoint properties	DisjointDataProperties (R1 R2)		No
pairwise disjoint properties	DisjointDataProperties (R1 ... Rn)		No
functional property	FunctionalDataProperty(R)	Domain/Range Tab of Data Property	Yes

Similarly to object properties above, there is only one OWL2 feature that users of the NeOn toolkit cannot immediately access via the GUI, namely disjoint data properties.

## Datatype Definitions

OWL2 allows now to define new datatypes by giving data range specifications a new name.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>Datatype Definitions</b>			
datatype definition	DatatypeDefinition(DN D)	Context menu of Datatype Folder of Ontology Navigator	Yes

This feature is not yet supported by the NeOn Toolkit because without being able to specify complex data ranges this would be meaningless.

## Assertions

In this group we collect all A-Box axioms making assertions about individuals.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>Assertions</b>			
individual equality	SameIndividual(a1 ... an)	Taxonomy tab of Individual	Yes
individual inequality	DifferentIndividuals(a1 a2)		Yes
pairwise individual inequality	DifferentIndividuals (a1 ... an)		Yes
class assertion	ClassAssertion(C a)		Yes
positive object property assertion	ObjectPropertyAssertion (PN a1 a2 )	Properties tab of Individual	Yes
positive data property assertion	DataPropertyAssertion (R a v )		Yes
negative object property assertion	NegativeObjectPropertyAssertion (P a1 a2 )		No
negative data property assertion	NegativeDataPropertyAssertion (R a v )		No

All essential types of assertions are supported by the NeOn toolkit. Only negative assertions are not yet implemented in the user interface.

## Keys

OWL2 now allows specifying key properties for classes, which formulates a rule that all individuals of that class with the same values for all key properties are identical.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>Keys</b>			
Key	HasKey(C (P1 ... Pm) (R1 ... Rn) )		no

This feature is not implemented in the NeOn Toolkit yet.





## Declarations

Entities of all kinds can be declared to exist in OWL2.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>2.6 Declarations</b>			
class	Declaration( Class( CN ) )	Context menu of folders and entites of Ontology Navigator	yes
datatype	Declaration( Datatype( DN ) )		no
object property	Declaration( ObjectProperty( PN ) )		yes
data property	Declaration( DataProperty( R ) )		yes
annotation property	Declaration ( AnnotationProperty( A ) )		yes
named individual	Declaration ( NamedIndividual( aN ) )		yes

With the exception of declaring datatypes users of the NeOn toolkit can introduce new instances all other entity types.

## Annotations

Annotations represent metadata about axioms or entities.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>2.7 Annotations</b>			
<b>Annotations</b>			
annotation assertion	AnnotationAssertion ( A s t )	Annotations tab of all entities	yes
annotation of an axiom	AXIOM(Annotation ( A t ) ...)		no
annotation of another annotation	Annotation(Annotation ( A t ) ... A1 t1)		no

The OWL2 specification supports extensive use of annotations. Users of the NeOn toolkit can read, create and modify annotations for all entity types and ontologies. Axiom annotations and annotations of annotations are not yet supported.

## Annotation Properties

OWL2 specifies a number pre-existing annotation properties.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>Annotation Properties</b>			
named annotation property	A		yes
human-readable name	rdfs:label		yes
human-readable comment	rdfs:comment		yes
additional information	rdfs:seeAlso		yes
defining agent	rdfs:isDefinedBy		yes
version information	owl:versionInfo		yes
deprecation	owl:deprecated		yes
backwards compatibility	owl:backwardCompatibleWith		yes
incompatibility	owl:incompatibleWith		yes
Prior version	owl:priorVersion		yes

All predefined annotation properties are supported by the NeOn toolkit. In addition, user defined annotation properties can be introduced and used.

## Annotation Axiom

OWL2 introduces the new features for modelling properties of annotation properties.

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>Annotation Axioms</b>			
annotation subproperties	SubAnnotationPropertyOf(A1 A2)		no
annotation property domain	AnnotationPropertyDomain(A U)		no

annotation property range	AnnotationPropertyRange(A U)		no
---------------------------	------------------------------	--	----

These new OWL2 features were not yet implemented in the NeOn Toolkit.

## Ontologies

In the final group we find ontology-level information

	Functional Syntax	NeOn Toolkit 2.3 GUI Element	supported by GUI NeOn Toolkit 2.3
<b>2.8 Ontologies</b>			
OWL ontology (importing)	Ontology([ON [U]] Import(ON1)... Annotation(A t) ... )	Imports and Namespace tab of Ontology	yes
prefix declaration	Prefix(p=U)		yes

In the NeOn Toolkit we support ontology imports and the management of prefixes.

## Reference

**Bao et al 2009:** Jie Bao, Elisa F. Kendall, Deborah L. McGuinness, Peter F. Patel-Schneider: OWL 2 Web Ontology Language Quick Reference Guide. W3C Recommendation 27 October 2009.  
<http://www.w3.org/TR/2009/REC-owl2-quick-reference-20091027/>

**Golbreich, Wallace 2009:** Christine Golbreich, Evan K. Wallace (ed.): OWL 2 Web Ontology Language. New Features and Rationale. W3C Recommendation 27 October 2009.  
<http://www.w3.org/TR/2009/REC-owl2-new-features-20091027/>

1cf. <http://www.w3.org/TR/owl2-quick-reference/>

### 3. User Interface Overview

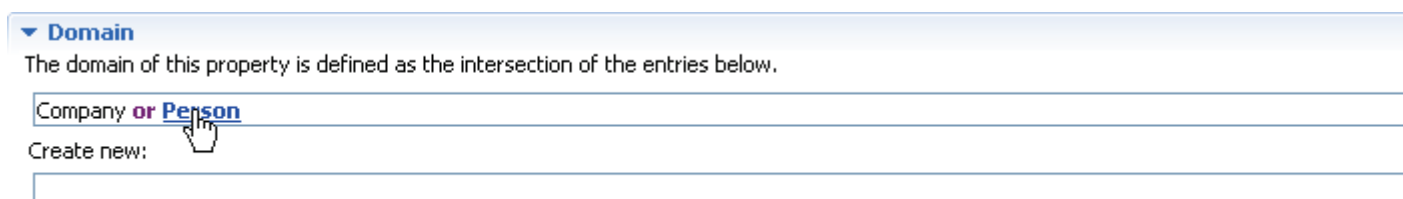
The OWL Editor consists of the following main panels:

- **Ontology Navigator.** The Ontology Navigator shows the ontology projects, their corresponding ontologies, and the hierarchies of the current ontology.
- **Individuals panel.** The Individuals panel displays all instances of the most recently selected class.
- **Entity Properties panel.** The Entity Properties panel is the main work area for defining and modifying resources of the selected ontology.

In addition, the OWL editor is characterized by many useful user-interaction features that are described below.

#### Navigation

If a resource is selected in either the Ontology Navigator or the Individuals panel, it will be loaded in the Entity Properties panel. From there you can navigate to other resources by pressing the Control (Ctrl) key (the entity becomes highlighted in blue and gets underlined to indicate that it is a clickable hyper link) and clicking its name.



#### Autocomplete

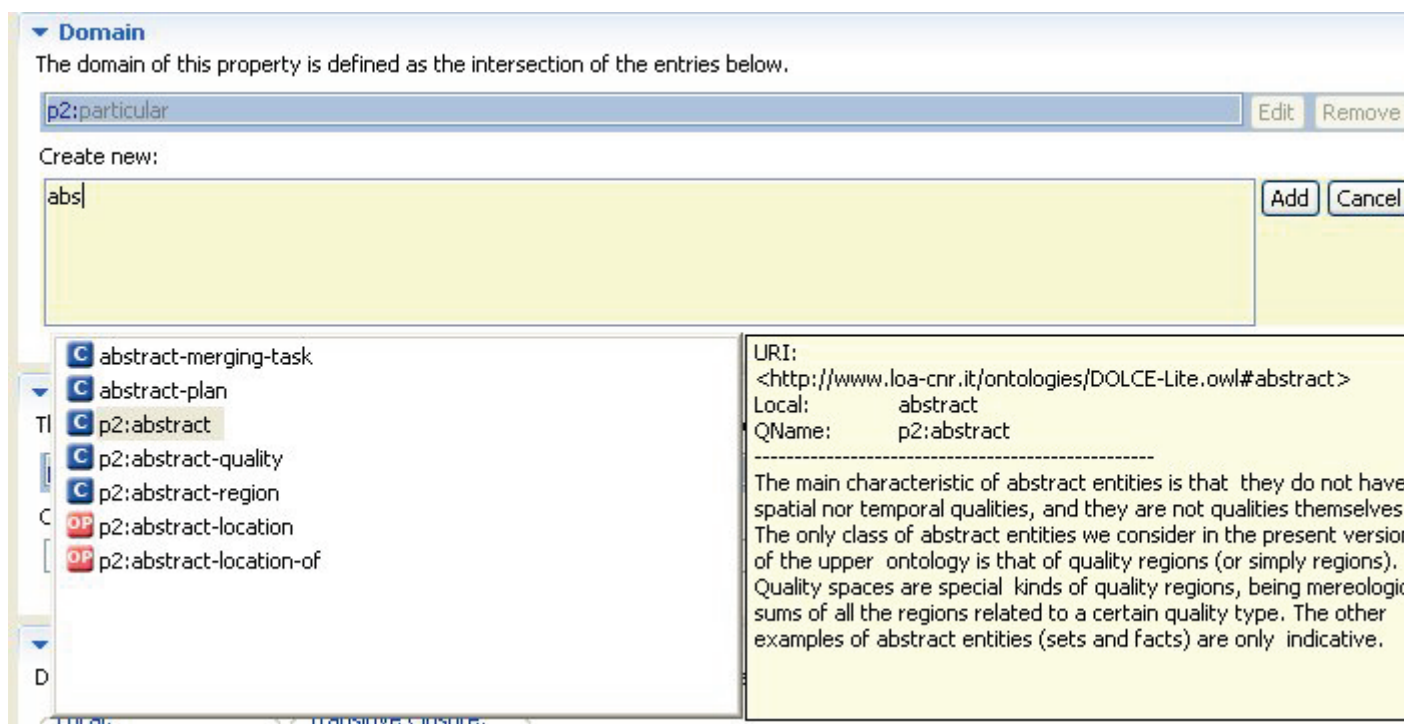
The OWL Editor also features a useful autocomplete function. Autocomplete involves the OWL Editor predicting a word or phrase that you want to type without you actually typing it completely. It is available in all text boxes while in edit mode. You can start autocomplete by clicking `Ctrl+spacebar` on your keyboard or by simply starting to type; autocomplete starts within a second.

In the example below, the user typed "abs" and the toolkit proposes a list of classes that contain this substring. On the right, it also shows a tooltip with the complete URI, and also the `rdfs:comment` for the selected entities, in this case "p2:abstract".

---

**Note** – Notice that autocomplete has an adaptive suggestion functionality, that is, it only shows suggestions that match your input.

---



## Entity Label Mode

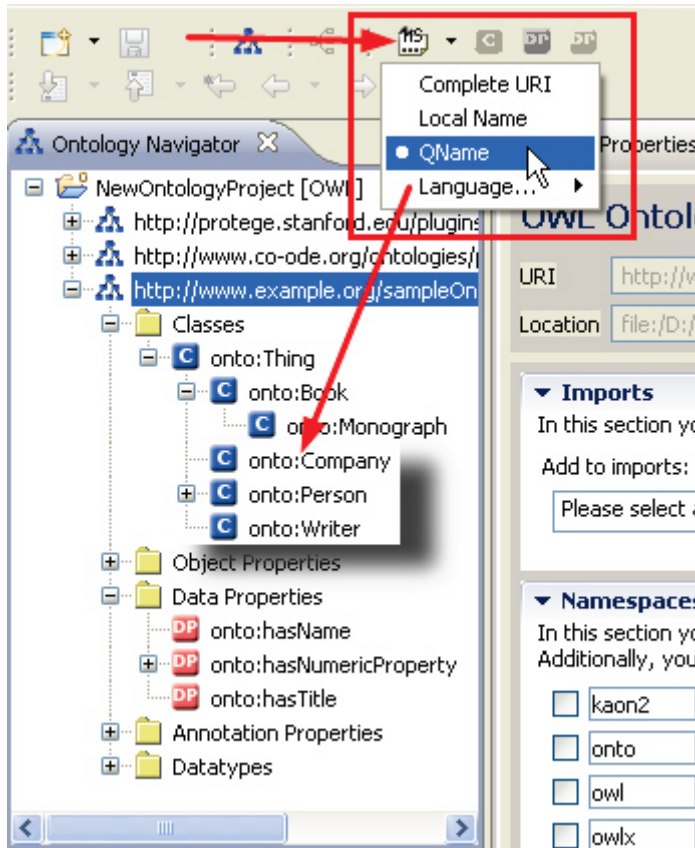
The labels that are displayed for entities in the different fields of the user interface depend on the selected *Entity Label Mode*. The "ns"-toolbar icon (cf. screen shot below) lets you choose between four different modes:

- **Complete URI.** Displays the complete URI of an entity (e.g., "http://www.example.org/sampleOntology#Company")
- **Local Name.** Displays only the local name of an entity (e.g., "Company"). This makes the ontology a lot more readable.
- **QName.** Displays the namespace prefix and local name of an entity (e.g., "onto:Company"). This is the recommended setting and is especially useful when multiple ontologies are used or different namespaces are in use.
- **Language.** If the ontology contains labels (`rdfs:label`) for entities, this option can displays the human readable labels in a specific language, thus providing multi-language support.

---

**Note** – Notice that only those languages are available which are specified in the Window >> Preferences >> Language Preferences dialog box.

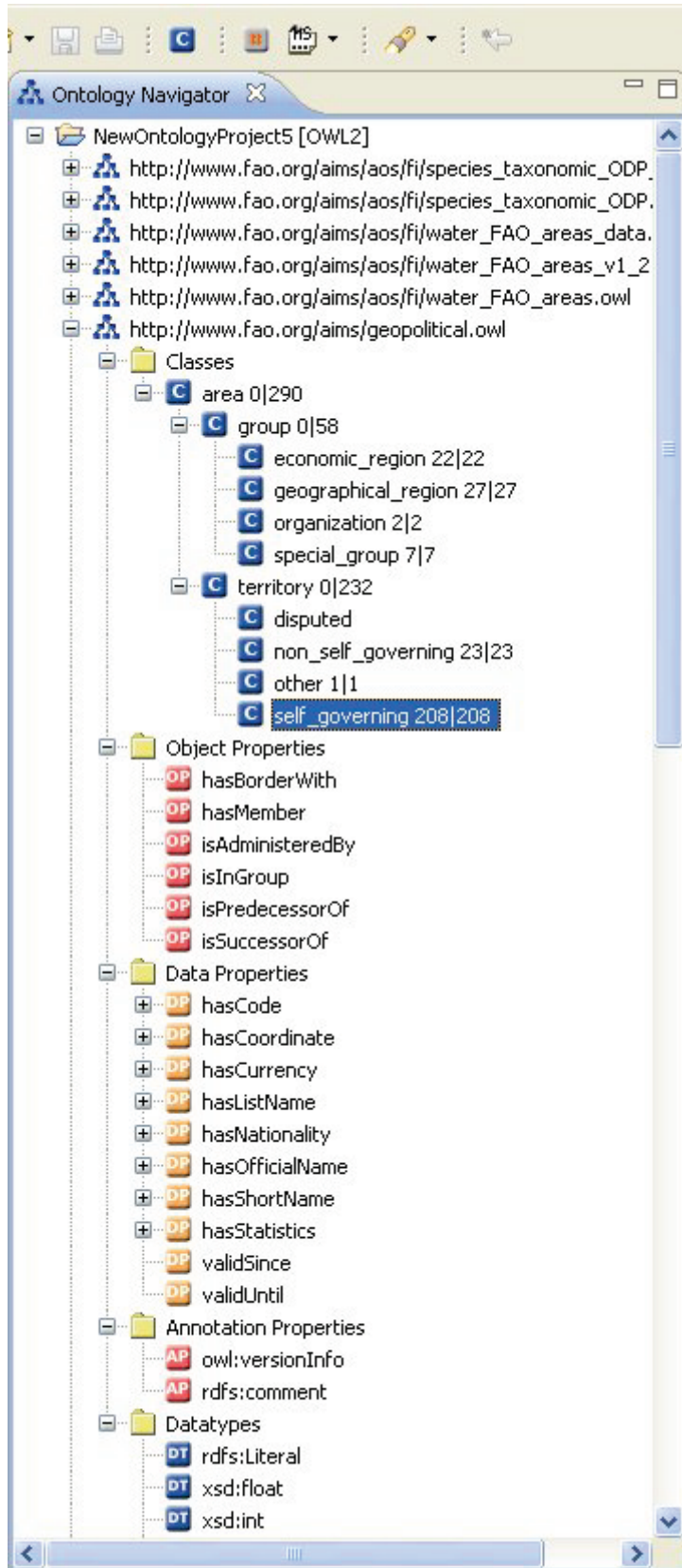
---



### 3.1. Ontology Navigator

The Ontology Navigator displays the different OWL ontology projects and their corresponding ontologies. The navigator is a hierarchical structure where each **ontology project** can hold multiple ontologies, which can refer to each other (via import statements). Each **ontology** contains some folders for the different entity types that exist in OWL. Within these folders the entities are arranged hierarchically. You can create, change, move, copy, and delete ontology elements in the navigator. The different actions possible are listed in the appropriate context menus.





To open the context menu of an element, click on the element using the right mouse button. If an element contains further sub-elements, then a plus sign is displayed on the left of the element. A click on this plus sign expands the sub-elements. In order to close the view of the sub-elements, click on the minus sign which shows besides the parent element.

Many nodes that are displayed in this tree can be dragged-and-dropped in order to modify the hierarchical structure of the ontology. For example, a class can be moved to a new superclass by simply dragging it to the new location. When holding the Ctrl key and dragging a node then, you will see the mouse cursor change to contain a little "+" icon. When dropping an entity it will *not be moved* to the new location but it will be *copied* there, resulting in additional `subClassOf` or `subPropertyOf` axioms.

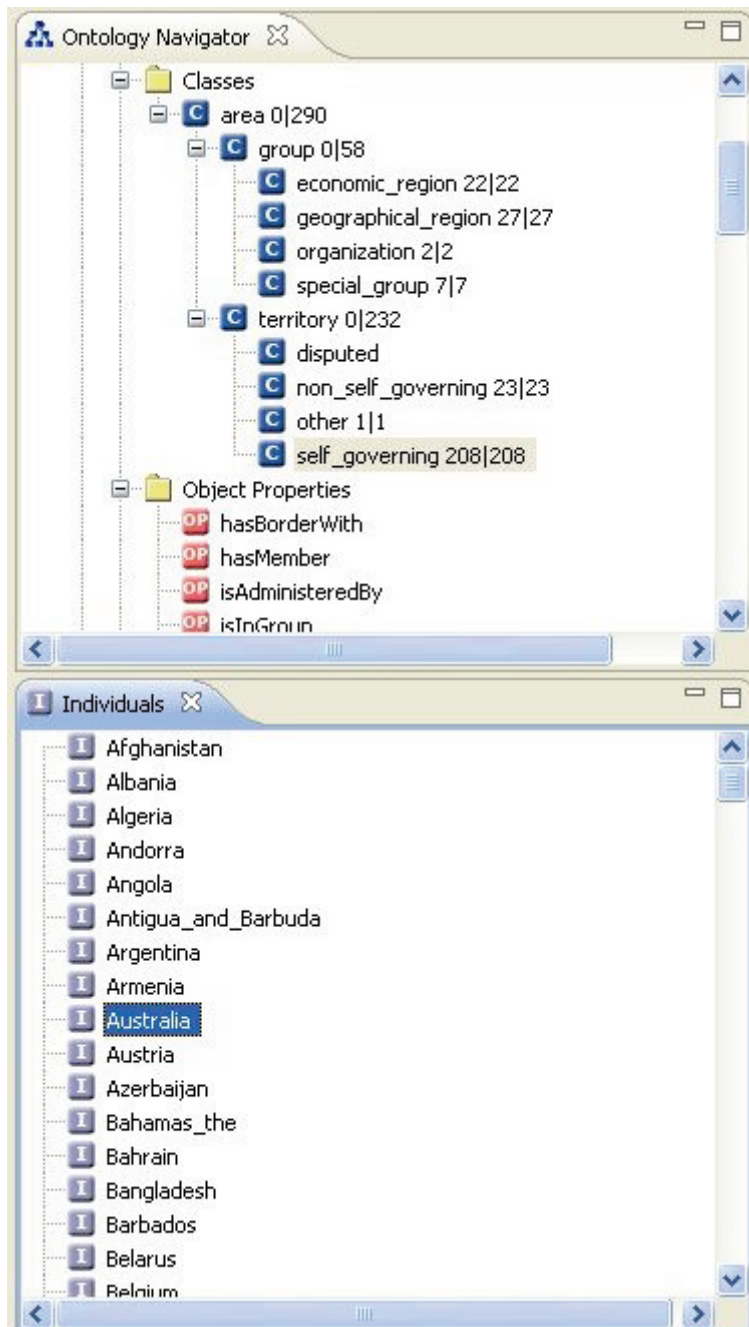
The Ontology Navigator can also give some information about the number of individuals which are contained in the ontology for each class. This helps in the task of locating instances. In the screen shot we see that there are 208 instances of the class "self governing countries". Its super class "territory" has 232 instances. All of them are inherited from sub-classes and no direct instances. With the button with the red "#" icon in the toolbar you can switch this instance-counting on and off, since, for big ontologies this can be time consuming.

Within an OWL project the Ontology Navigator hosts different types of entities. When selecting one of the nodes, the corresponding Entity Property panel (in the right part of the screen) will display the details of the recently selected entity. The Ontology Navigator hosts the following types of entities:

- **OWL Ontology**
- **OWL Class**
- **Object Property**
- **Datatype Property**
- **Annotation Property**
- **Datatype**

### **3.2. Individuals Panel**

The Individuals panel displays individuals (often referred to as "instances") of the class currently selected in the Ontology Navigator.



In the screen shot the class "self governing country" is selected and the Individuals panel lists all 208 instances of this class. Selecting an instance from the list will display its details in the Entity Properties panel on the right part of the screen.

New individuals of a class can be created either using the context menu of the class in the Ontology Navigator or by using the context menu of the Individuals panel and selecting "new individual" here. This option will only be accessible if a class is selected in the Ontology Navigator. The Individual Panel's context menu also allows to delete and rename individuals. Via drag-and-drop individuals can be moved or copied to other classes in their respective ontologies.

### 3.3. Entity Properties

The Entity Properties panel is the main work area for viewing and editing classes, properties, and individuals. There are seven different Entity Properties panels for the different entity types relevant for OWL ontologies

When selecting one of the nodes in the Ontology Navigator, the appropriate Entity Properties View will display the details for this entity.

- **OWL Ontology**
- **OWL Class**
- **Object Property**
- **Datatype Property**
- **Annotation Property**
- **Datatype**
- **Individual**

### 3.3.1. OWL Ontology

This Entity Properties panel displays general properties of an OWL ontology itself. Once you have selected an ontology in the Ontology Navigator, the Entity Properties panel will show import and namespace information of the currently selected ontology.

The screenshot shows the 'OWL Ontology' panel in the Eclipse IDE. The panel has a title bar with 'Ontology Visualizer', 'Search', and 'Entity Properties' tabs. The main content area is titled 'OWL Ontology' and contains the following fields and sections:

- URI:**
- Location:**
- Imports:** A section with a dropdown arrow and a text field containing 'http://www.ontoprise.com/ontologies/ontowebtravel'. Below it, a label 'Add to imports:' is followed by a button 'Please select an ontology to import'.
- Namespaces:** A section with a dropdown arrow and a table of namespaces. The table has two columns: a checkbox and a text field. The namespaces listed are:

Checkbox	Namespace
<input type="checkbox"/>	rdf http://www.w3.org/1999/02/22-rdf-syntax-ns#
<input type="checkbox"/>	owl http://www.w3.org/2002/07/owl#
<input checked="" type="checkbox"/>	onto http://www.example.org/sample#
<input type="checkbox"/>	xsd http://www.w3.org/2001/XMLSchema#
<input type="checkbox"/>	rdfs http://www.w3.org/2000/01/rdf-schema#
- Create new:** A section with a label 'Create new:' and a text field.

At the bottom of the panel, there are three tabs: 'Imports and Namespaces' (selected), 'Annotations', and 'Ontology Imports Graph'.

## Imports

In the Imports section, ontologies imported by the currently selected ontology are shown. Importing another ontology brings the entire set of classes, properties, and individuals provided by that ontology into the current ontology. Importing another ontology will also import all of the ontologies that the initial ontology imports.

---

**Note** – Only ontologies located in the current ontology project can be imported.

---

---

**Tip** – The OWL preferences dialog lets you decide whether you want to see imported axioms (and entities) in the user interface or not.

---

## Namespaces

Usually, each ontology also has a number of namespace declaration. The namespace declaration provides convenient means to reference names defined in this or other OWL ontologies. A namespace is a string of characters that prefixes the identifiers of classes, properties, and individuals in an ontology. By maintaining shortcuts (prefixes) for the usually long namespaces it is possible to make the display of entities, expressions and axioms much easier to read. The prefixes will be used when the QName option is chosen for the Entity Label Mode

Additionally, every ontology can specify a default namespace, stating that unprefix names refer to this namespace. You can specify the default namespace by selecting the check box in front of the namespace. Newly generated resources will then get this namespace by default.

---

**Note** – If you did not select a default namespace, the following namespace will be used by default: `http://example.org/ontology/`.

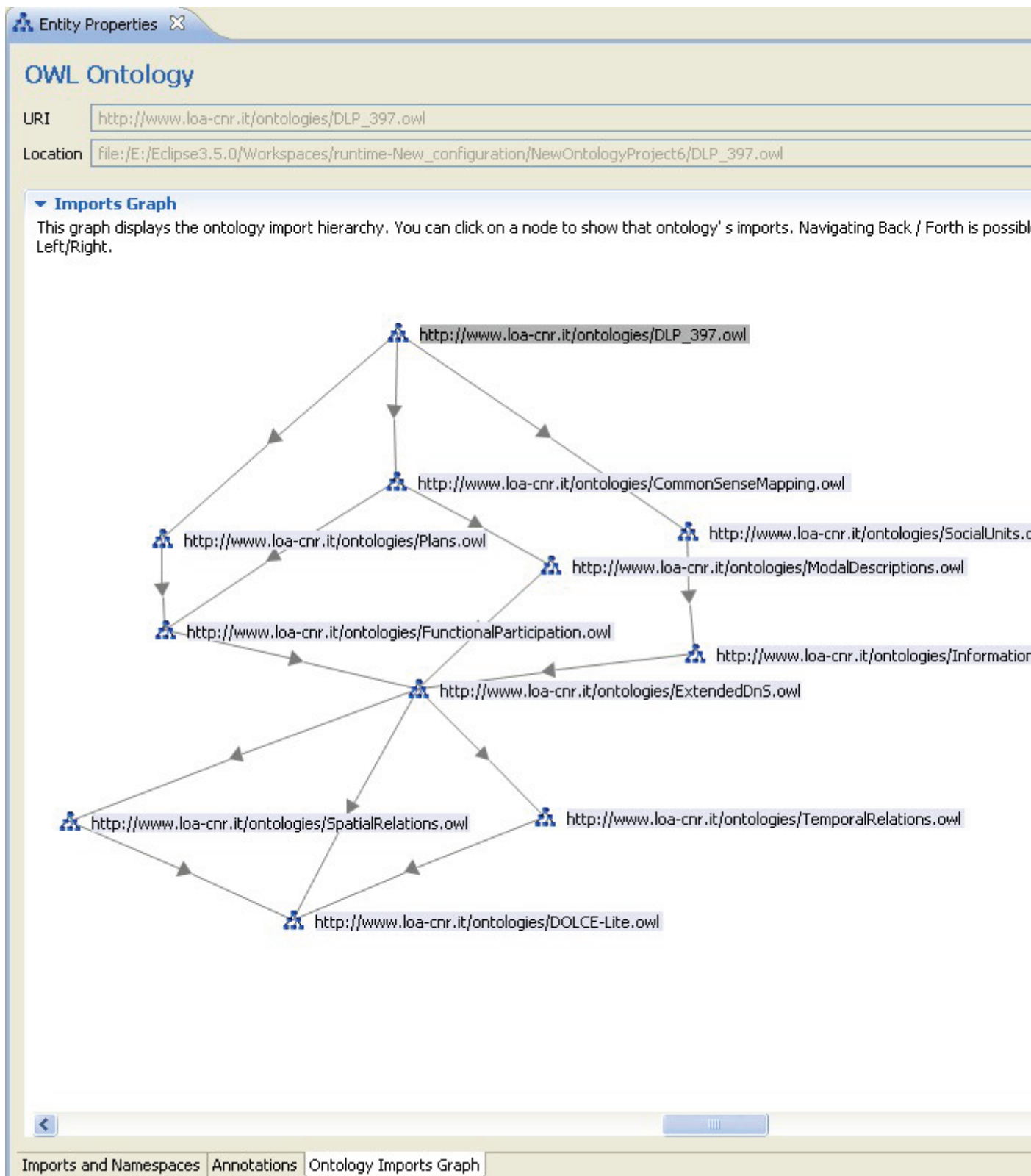
---

## Annotations

For a description of the Annotations section please refer to How to reate an Annotation?.

## Ontology Imports Graph

In this view all ontologies that are directly or indirectly visible (via the imports relationship) from this ontology are displayed as a graph. Each ontology is represented by a node and each "import" is represented by a directed link between them. The importing ontology points to the imported one.



### 3.3.2. OWL Classes

Generally speaking, OWL classes can be considered as sets of individuals that share similar characteristics. These classes are organized in hierarchies, of which `owl:Thing` is the root class. Every individual in the ontology is a member of the class `owl:Thing`. Thus, each class

you create is implicitly a subclass of `owl:Thing`. A class may be a subclass of more than one parent class. Likewise, an individual may belong to more than one class.

OWL classes are described through so-called class descriptions. A class description describes an OWL class, either by a class name or by specifying the complex class expression, representing an unnamed anonymous class. OWL distinguishes six types of class descriptions:

- Class identifiers (a URI reference)
- Exhaustive enumeration of individuals that together form the instances of a class
- Restrictions
- Intersection of two or more class descriptions
- Union of two or more class descriptions
- Complement of a class description

These class descriptions are distributed over the following tabs in the Entity Properties panel:

- **Class Restrictions.** Unnamed classes that are defined based on restrictions on the values for particular properties of the class.
- **Taxonomy.** All super classes, subclasses, equivalent classes and disjoint classes of the selected class.
- **Annotations.** OWL allows classes, properties, individuals, and the ontology itself to be annotated with readable labels, comments, or other useful information.

For instructions on how to create a class, see [Create a Class](#).

## Online References

Further information is available at the following Web sites:

- [The OWL 2 primer about basic classes](#)
- [The OWL 2 primer about advanced class relationships \(esp. Restrictions\)](#)

## Class Restrictions

OWL can be used to model situations in which the members of a particular class must have certain properties. You can use various restriction statements to indicate that something should apply to all values of a property versus only for some members. In addition, you can even specify a particular value for a property of all members of a class.

OWL distinguishes two kinds of restrictions: value restrictions and cardinality restrictions. A value restriction puts constraints on the range of the property when applied to this particular class. A cardinality restrictions puts constraints on the number of values the property can take, in the context of this particular class.

OWL provides the following value restrictions:

- `owl:allValuesFrom (ALL)`
- `owl:someValuesFrom (SOME)`
- `owl:hasValue (HAS_VALUE)`
- `owl:hasSelf (HAS_SELF)`

In addition, OWL provides the following cardinality restrictions:

- owl:maxCardinality (AT\_MOST/MAX)
- owl:minCardinality (AT\_LEAST/MIN)
- owl:cardinality (EXACTLY/CARD)

The screenshot shows the 'Entity Properties' panel in the OWL Editor. The 'Identifier' section at the top shows a URI: <http://www.example.org/sample#Person>. Below this, there are two main sections: 'Super Restrictions' and 'Equivalent Restrictions'. Each section contains a table with columns for 'Property', 'Quantifier', 'Min/Max', and 'Range'. In the 'Super Restrictions' section, there are two rows with 'hasRelative' as the property, 'SOME' and 'ALL' as quantifiers, and 'Person' as the range. Below these is a 'Create new:' section with a text input, a 'SOME' quantifier dropdown, and an 'Add' button. The 'Equivalent Restrictions' section has a similar 'Create new:' section. At the bottom of the panel, there are tabs for 'Class Restrictions', 'Taxonomy', and 'Annotations', with 'Class Restrictions' being the active tab.

Class restrictions can either be specified as Super Restrictions or Equivalent Restrictions. If you add the class restriction in the Super Restriction section, the class under consideration will become a subclass of the restriction class (i.e. you define a *necessary condition* for the class). On the other hand, if you add the class restriction in the Equivalent Restrictions section, both, the restriction class and the restricted class, will be interpreted as equivalent, which means that they always have exactly the same members (i.e. you define a *necessary and sufficient* condition for the class).

For instructions on how to create a class, see Create a Class. To create a class restriction, proceed as follows.

1. **Select a class in the Ontology Navigator.**  
The class will be displayed in the Entity Properties panel.
2. **Enter a property.**  
Enter the property used to restrict the class. You can also use the auto-complete feature of the OWL Editor.



3. **Select a quantifier.**  
Select the appropriate value constraint or cardinality constraint.
4. **(Optional) Select a number.**  
This only applies if you have selected a cardinality constraint.
5. **Select a range.**  
Select the class for which the property under consideration only should have values.
6. **Click Add.**  
The class restriction will be created. To discard your changes, click Cancel.

---

**Note** – The ranges of restrictions can be entered using the Manchester OWL Syntax. For more information, see Manchester Syntax.

---

### **Taxonomy**

On this tab you can define classes through class axioms. The simplest form of a class axiom is a class description of type 1 above. It just states the existence of a class, using `owl:Class` with a class identifier. Class axioms typically contain additional components that state necessary and/or sufficient characteristics of a class. OWL contains three language constructs for combining class descriptions into class axioms:

- **Subclasses.** The OWL language construct `rdfs:subClassOf` allows you to say that a class is a subset of another class.
- **Equivalent Classes.** The OWL language construct `owl:equivalentClass` allows you to say that a class has exactly the same members as another class.
- **Disjoint Classes.** The OWL language construct `owl:disjointWith` allows you to say that a class has no members in common with another class.

In addition to these language constructs, the OWL Editor allows you to specify the **super classes** of a class, which is the same as making the class under consideration a subclass of another (parent) class. For example, you will notice that `owl:Thing` is the super class of all other classes because every user-defined class is implicitly a subclass of `owl:Thing`.

Entity Properties

Identifier

URI: <http://www.example.org/sample#Person>

Super Classes

hasRelative **only** Person

hasRelative **some** Person

MyThing

Create new:

Sub Classes

Student

Create new:

Equivalent Classes

Create new:

Disjoint Classes

Book

Company

Create new:

Class RestrictionsTaxonomyAnnotations

To create a class axiom, proceed as follows:

- Select a class in the Ontology Navigator.**  
The class will be displayed in the Entity Properties panel.
- Go to the Taxonomy tab.**  
The class axioms will be displayed.
- Create a class axiom.**  
In the Create New text box, enter a class name. You can also use the auto-complete feature of the OWL Editor. In addition to simple classes you can also enter complex class descriptions using the Manchester Syntax.

#### 4. Click Add.

The class axiom will be created. To discard your changes, click Cancel.

### Annotations

Usually, annotations are used to add information to resources of the ontology. They contain useful information such as who is the creator or what are the readable labels and comments for single elements of the ontology.

For information on how to create an annotation, see [Create an Annotation](#).

The screenshot shows a web-based interface for editing an ontology. The main window is titled 'Entity Properties' with a close button. It has a tabbed interface with three tabs: 'Class Restrictions', 'Taxonomy', and 'Annotations', with 'Annotations' currently selected. The 'Annotations' tab contains a section for adding annotations to a resource. At the top, under the 'Identifier' section, the URI is set to '<http://www.example.org/sample#Person>'. Below this is a table for annotations with columns: 'Annotation Property', 'Value', 'Type', and 'Language'. The first row contains 'rdfs:label', 'personne', 'rdfs:Literal', and 'fr'. Below the table is a 'Create new:' section with three empty input fields. At the bottom of the dialog, there are three tabs: 'Class Restrictions', 'Taxonomy', and 'Annotations'.

Annotation Property	Value	Type	Language
rdfs:label	personne	rdfs:Literal	fr

Create new:

			--
--	--	--	----

### 3.3.3. Object Properties

An object property is a binary relation between two individuals that lets you assert general facts about the members of classes and specific facts about individuals. Characteristics of a property are defined by so-called property axioms. In its simplest form, a property axiom just defines the existence of a property. However, property axioms often define additional characteristics of properties. OWL supports the following constructs for property axioms:

- **RDF Schema constructs.** Represented in OWL by the language constructs `rdfs:subPropertyOf`, `rdfs:domain` and `rdfs:range`.
- **Relations to other properties.** Represented in OWL by the language constructs `owl:equivalentProperty` and `owl:inverseOf`.
- **Global cardinality constraints.** Represented in OWL by the language constructs `owl:FunctionalProperty` and `owl:InverseFunctionalProperty`.

- **Logical property characteristics.** Represented in OWL by the language constructs `owl:SymmetricProperty`, `owl:TransitiveProperty` and others.

These language constructs are distributed to the following tabs in the Entity Properties panel:

- **Domain and Range.** Properties may have a domain and a range specified, that is, they link individuals from the domain to individuals from the range. In addition, you can add global cardinality constraints as well as logical property characteristics here.
- **Taxonomy.** Object properties can be organized into a superproperty-subproperty hierarchy, also known as a taxonomy.
- **Annotations.** OWL allows properties to be annotated with readable labels, comments, or other useful information.

For instructions on how to create an object property, see [Create an Object Property](#).

## Online References

Further information is available at the following Web sites:

- [OWL 2 Primer on Object Properties](#)
- [OWL 2 Primer on Object Property Characteristics](#)

## Domain and Range

The notions *domain* and *range* have meanings inspired by the mathematical uses of these terms. In mathematics, they are used to refer to how a function can be used. The domain of a function is the set of values for which it is defined, while the range is the set of values it can take. Similarly in OWL, if you specify a class *D* as the domain of the property *P* and class *R* as the range of *P*, the informal interpretation of this is that the Property *P* relates values from the class *D* to values from the class *R*.

---

**Note** – Use domain and range declarations with care. It is important to realize that in OWL domain and range declarations do not act as constraints to restrict the relation. It is tempting but not correct to think of `rdfs:domain` as the class of objects to which a property can be applied. Instead, it is used as an axiom for reasoning purposes. For example, any individual that uses a property with a given domain can be inferred by the reasoner to be a member of the domain class, for the statement *P* `rdfs:domain` *D* just means that all things to which *P* is applied can be inferred to be a member of class *D*. This, on the other hand, can lead to unexpected classification results and side effects, especially in situations when multiple ontologies may be merged. Therefore, consider not setting any domains and ranges and using local class restrictions instead.

---

In addition to simple classes you can also enter complex classes using the Manchester Syntax. Note that selecting multiple domains means that the domain of the property is the intersection of the classes (and similarly for range).

## Characteristics

Additionally, OWL allows the meaning of properties to be enriched through the use of property characteristics.

- **Functional.** If a property is functional, for a given individual, there can be at most one individual that is related to the individual via the property.
- **Inverse Functional.** If a property is inverse functional, then it means that the inverse property is functional. For a given individual, there can be at most one individual related to that individual via the property.
- **Reflexive.** If a property  $P$  is reflexive, then we can derive that each individual  $a$  is related to itself via  $P$ .
- **Irreflexive.** If a property  $P$  is irreflexive, then we can derive that no individual  $a$  is related to itself via  $P$ .
- **Symmetric.** If a property  $P$  is symmetric, and the property relates individual  $a$  to individual  $b$ , then individual  $b$  is also related to individual  $a$  via property  $P$ .
- **Asymmetric.** If a property  $P$  is asymmetric, and the property relates individual  $a$  to individual  $b$ , then individual  $b$  cannot be related to individual  $a$  via property  $P$ .
- **Transitive.** If a property  $P$  is transitive, and the property relates individual  $a$  to individual  $b$ , and also individual  $b$  to individual  $c$ , then we can infer that individual  $a$  is related to individual  $c$  via property  $P$ .

The Characteristics section contains two sets of check boxes. The left set is changeable and represents the assertions of the current ontology. The set on the right hand side, labeled transitive closure, cannot be changed and represents also assertions that come from imported ontologies.

Entity Properties

Identifier

URI: <http://www.example.org/sample#hasSoleAuthor>

Domain

The domain of this property is defined as the intersection of the entries below.

Book

Create new:

Range

The range of this property is defined as the intersection of the entries below.

Person

Create new:

Characteristics

Define if this property is functional, inverse functional, transitive or symmetric

Local:

☒ Functional
 ☐ Inverse functional
 ☐ Reflexive
 ☐ Irreflexive
 ☐ Symmetric
 ☐ Asymmetric
 ☐ Transitive

Transitive Closure:

☒ Functional
 ☐ Inverse functional
 ☐ Reflexive
 ☐ Irreflexive
 ☐ Symmetric
 ☐ Asymmetric
 ☐ Transitive

Domain and Range

Taxonomy

Annotations

## Taxonomy

On this tab you can specify relations to other properties. This includes the following settings:

- **Super Properties/Sub Properties.** Object properties can be defined in a hierarchy in the same way as classes can be organized in a hierarchy. For example, a hierarchy of the property `hasParent` may have a subproperty `hasMother` and `hasFather`. When querying the ontology, you can now ask for individuals that are linked by the `hasParent` property, and return all results that are linked by its subproperties.
- **Equivalent Properties.** The `owl:equivalentProperty` construct can be used to state that two properties have the same property extension.
- **Inverse Properties.** Properties usually have a direction, from domain to range. In practice, it is often useful to define relations in both directions: authors write books, books are written by authors. The `owl:inverseOf` construct can be used to define such an inverse relation between properties.
- **Sub property Chains.** OWL 2 introduces the new feature of property chaining, which lets you define that the the composition of a sequence of properties is a subproperty of

a named property, eg. the famous uncle example `hasParent` o `hasBrother` is a subproperty of `hasUncle`.

Entity Properties

Identifier

URI:

▼ Super properties

Create new:

▼ Sub properties

Create new:

▼ Equivalent properties

Create new:

▼ Inverse properties

Create new:

▼ Sub property chains

Create new:

Domain and Range Taxonomy Annotations

## Annotations

Usually, annotations are used to add information to resources of the ontology. They contain useful information such as who is the creator or what are the readable labels and comments for single elements of the ontology.

For information on how to create an annotation, see [Create an Annotation](#).

Entity Properties

Identifier

URI: <http://www.example.org/sample#hasAuthor>

Annotations

Annotation Property	Value	Type	Label
label	autor	Literal	

Create new:

Domain and Range Taxonomy Annotations

### 3.3.4. Data Properties

Datatype properties relate individuals to RDF literals or simple types defined in accordance with XML Schema datatypes. For example, the datatype property `hasName` might link individuals from the class `Person` to the XML Schema datatype `xsd:string`.

Characteristics of datatype properties can be configured on the following tabs in the Entity Properties panel:

- **Domain and Range.** Datatype properties may have a domain and a range specified, that is, they link individuals from the domain to RDF literals or XML Schema datatypes.
- **Taxonomy.** Datatype properties can be organized into a superproperty-subproperty hierarchy.
- **Annotations.** OWL allows properties to be annotated with readable labels, comments, or other useful information.

For instructions on how to create a datatype property, see [Create a Datatype Property](#).

### Online References

Further information is available at the following Web sites:

- [XML Schema Part 2: Datatypes Second Edition](#)



- OWL 2 Recommendation

### Domain and Range

The notions *domain* and *range* have meanings inspired by the mathematical uses of these terms. In mathematics, they are used to refer to how a function can be used. The domain of a function is the set of values for which it is defined, while the range is the set of values it can take. Similarly in OWL, if you specify a class `D` as the domain of the datatype property `P` and `xsd:string` as the range of `P`, the informal interpretation of this is that the property `P` relates values from the class `D` to values from the XML Schema datatype `xsd:string`.

In addition to simple classes you can also enter complex classes in the domain text box using the Manchester Syntax. Note that selecting multiple domains means that the domain of the property is the intersection of the classes.

### Characteristics

Just as object properties, datatype properties can be declared as "functional". For this purpose, OWL defines the built-in class `owl:FunctionalProperty`. A functional property is a property that can have only one (unique) value `y` for each instance `x`, that is, there cannot be two distinct values `y1` and `y2` such that the pairs `(x, y1)` and `(x, y2)` are both instances of this property.

The Characteristics section contains two "Functional" check boxes. The left one is changeable and represents the assertions of the current ontology. The one on the right hand side, labeled transitive closure, cannot be changed and represents also assertions that come from imported ontologies.

The screenshot shows a web-based interface for defining an entity property. The title bar reads 'Entity Properties'. The 'Identifier' section contains a URI field with the value '<http://www.example.org/sample#hasName>'. The 'Domain' section has a description 'The domain of this property is defined as the intersection of the entries below.', a text field containing 'Company or Person', and a 'Create new:' section with an empty text field and an 'Add' button. The 'Range' section has a description 'The range of this property is defined as the intersection of the entries below.', a text field containing 'xsd:string', and a 'Create new:' section with an empty text field and an 'Add' button. The 'Characteristics' section has a description 'Define if this property is functional' and two sub-sections: 'Local:' with a 'Functional' checkbox, and 'Transitive Closure:' with a 'Functional' checkbox. At the bottom, there are three tabs: 'Domain and Range' (selected), 'Taxonomy', and 'Annotations'.

## Taxonomy

On this tab you can specify the characteristics of the datatype property regarding its hierarchy. This includes the following settings:

- **Super Properties/Sub Properties.** Datatype properties can be defined in a hierarchy in the same way as classes can be organized in a hierarchy. For example, a hierarchy of the property `hasName` may have a subproperty `hasFirstName` and `hasLastName`. When querying the ontology, you can now ask for individuals that are linked by the `hasName` property, and return all results that are linked by its subproperties.
- **Equivalent Properties.** The `owl:equivalentProperty` construct can be used to state that two properties have the same property extension.

Entity Properties

Identifier

URI:

▼ Super properties

Create new:

▼ Sub properties

Create new:

▼ Equivalent properties

Create new:

Domain and Range Taxonomy Annotations

### Annotations

Usually, annotations are used to add information to resources of the ontology. They contain useful information such as who is the creator or what are the readable labels and comments for single elements of the ontology.

For information on how to create an annotation, see [Create an Annotation](#).

Entity Properties

Identifier

URI:

Annotations

Annotation Property	Value	Type	Language
<input type="text" value="rdfs:label"/>	<input type="text" value="Name"/>	<input type="text" value="rdfs:Literal"/>	<input type="text" value="de"/>

Create new:

Domain and Range Taxonomy Annotations

### 3.3.5. Annotation Properties

OWL allows classes, properties, individuals, and the ontology header to be annotated with useful information such as labels, comments, creation date, author, or references to web pages. An OWL annotation simply associates property-value pairs for resources of the ontology or the entire ontology itself. However, this information is not really part of the semantics of the ontology and will thus be ignored by reasoning engines.

The selection of meaningful names and the use of annotations is especially important for documentation, maintenance, and traceability purposes, e.g. if the ontology is to be read or reused by someone else. You can enter annotations when you first create a resource or when you modify it later on.

- **label.** An `rdfs:label` statement supports a natural language label for the resource.
- **comment.** An `rdfs:comment` statement supports a natural language comment about a resource for future readers to see.
- **seeAlso.** An `rdfs:seeAlso` statement provides a way to specify the web location of supplementary information about the resource.
- **isDefinedBy** An `rdfs:isDefinedBy` statement provides a link to the source of information about the resource.
- **versionInfo.** An `owl:versionInfo` statement is intended to provide hooks for version control systems working with ontologies.

There are a number of built-in annotation properties already pre-specified by RDF(S) and OWL2, which can be used immediately, e.g.:

But you can also introduce your own new annotation properties. For instructions on how to do that, see [Create an Annotation Property](#).

Entity Properties

Identifier

URI:

Annotations

Annotation Property	Value	Type
<input type="text" value="myAnnotationProperty"/>	<input type="text" value="Can this work??? --- Yes, it can!!!"/>	<input type="text" value="string"/>

Create new:

<input type="text"/>	<input type="text"/>	<input type="text"/>
----------------------	----------------------	----------------------

## Online References

Further information is available at the following Web sites:

- [OWL2 Primer about Annotation Properties](#)

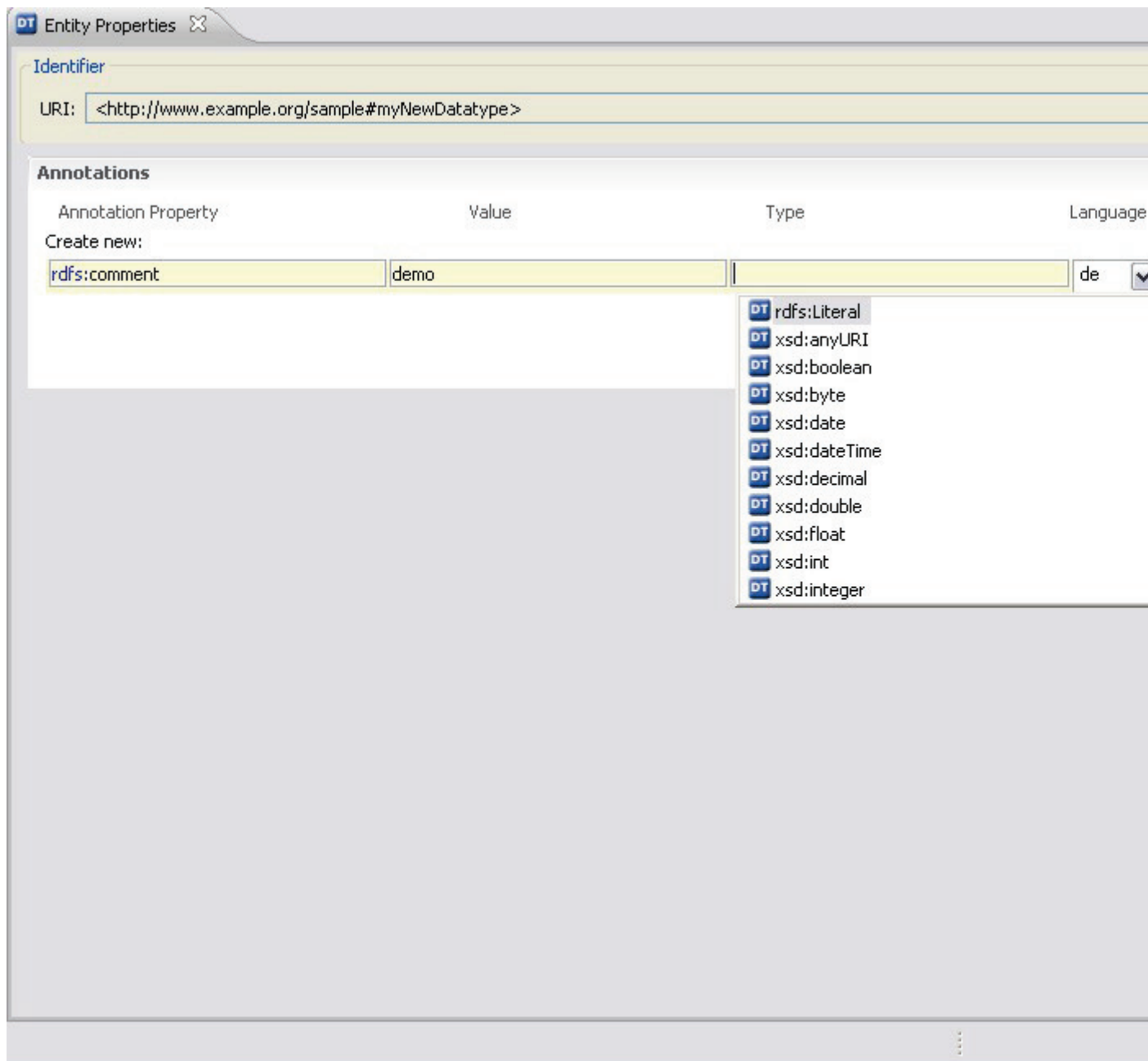
### 3.3.6. Datatypes

All datatypes used in the ontology are listed in the Datatypes folder. These datatypes can be annotated (see below). For an overview of all XML Schema datatypes, please refer to [XML Schema Part 2: Datatypes Second Edition](#) at W3C.org.

#### Annotations

Usually, annotations are used to add information to resources of the ontology. They contain useful information such as who is the creator or what are the readable labels and comments for single elements of the ontology.

For information on how to create an annotation, see [Create an Annotation](#).



### 3.3.7. OWL Individuals

The Entity Properties View for individuals contains the property values of the instance. The various configuration options are located on the following tabs:

- Properties
- Taxonomy
- Annotations

#### Properties

On the Properties tab you can edit the the object and datatype property values of the selected instance. Object properties link an individual to an individual. Datatype properties link an individual to an XML Schema datatype value or an RDF literal. For Data Properties you can

also define a Type, which is either an XML Schema datatype or RDFS Literal, and a language code.

**Entity Properties** X

Identifier

URI: <http://www.example.org/sample#peter>

▼ Object Property Values

Property	Value
hasNephew	walter

Create new:

--	--

▼ Data Property Values

Property	Value	Type	Language
----------	-------	------	----------

Create new:

--	--	--	--

Properties Taxonomy Annotations

## Taxonomy

On the Taxonomy tab you can edit the classes of which the individual is member of. In addition, you can specify whether two individuals are identical or different. Note that two different names of individuals can actually refer to the same individual. For the ontology language OWL does not make the so-called Unique Name Assumption (UNA), which means that that two different names of individuals in the ontology necessarily refer to different entities in the world. Instead, OWL provides explicit constructs to express that two names denote distinct entities.

- `owl:sameAs` is the OWL property that states that two given named individuals have the same identity.
- `owl:differentFrom` is the OWL property that states that two given named individuals have different identities.

Entity Properties

Identifier

URI:

▼ Same as

Create new:

▼ Different from

Create new:

▼ Classes

Create new:

Properties Taxonomy Annotations

### Annotations

Usually, annotations are used to add information to resources of the ontology. They contain useful information such as who is the creator or what are the readable labels and comments for single elements of the ontology.

For information on how to create an annotation, see [Create an Annotation](#).



**Entity Properties** X

Identifier

URI:

**Annotations**

Annotation Property	Value	Type	Label
Create new:			
<input type="text"/>	<input type="text"/>	<input type="text"/>	

Properties Taxonomy Annotations

### 3.4. The Manchester Syntax

The Manchester Syntax is a user-friendly compact syntax for the ontology language OWL, especially for writing OWL class expressions. Although the syntax borrows ideas from the OWL Abstract Syntax, it is much less verbose, meaning that it is quicker to write and easier to read. While following the compactness of the DL Syntax, special mathematical symbols such as the universal quantifier ( $\forall$ ) or the existential quantifier ( $\exists$ ) have been replaced by keywords such as "only" and "some".

The Manchester Syntax for OWL2 is not strictly a part of the OWL2 recommendation by the W3C but it was developed by members of the W3C OWL2 working group and the syntax is published as a W3C Note. In this Note at <http://www.w3.org/TR/2009/NOTE-owl2-manchester-syntax-20091027/> you can find the complete language specification and also examples for using it.

In the NeOn Toolkit the Manchester Syntax is applicable whenever class expressions are expected, i.e. when complex classes are allowed as part of certain slots of the GUI. To become familiar with the Manchester Syntax, the OWL Editor also provides a special toolbar that helps with the Manchester Syntax keywords. To activate the Manchester Syntax Toolbar, proceed as follows:

- 1. Open the preferences.**

Go to Window >> Preferences >> OWL Preferences.



	Keyword	
owl:intersectionOf	<b>and</b>	Writer <b>and</b> Male
owl:unionOf	<b>or</b>	Male <b>or</b> Female
owl:complementOf	<b>not</b>	<b>not</b> Child

### Complex Class Expressions

Complex class expressions can be constructed using the above boolean constructors and restrictions. For example, the following string describes the set of people who have at least one child that has some children that are only men, that is, grandparents that only have grandsons:

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

---

**Note** – Please use parenthesis to disambiguate the meaning of expressions.

---

### 3.5. Search

Ontologies can become quite big in size. To facilitate the work with larger ontologies, NeOn Toolkit provides a helpful search dialog. The search dialog is very easy to use. Just open the Search dialog by pressing Ctrl+h (or by clicking the flash light icon in the tool bar), then go to the OWL Search tab and type in the keyword you want to search for. Additionally, you can also select various check boxes to control the scope of elements to be searched.

The result of your search is displayed in a special search panel. A double-click on a result opens the entry. For example, if you double-click on a search result of a class, the Entity Properties panel for this class will be displayed automatically.

---

**Note** – The search can also consider annotation values and data property values, not only entity names.

---

#### How to use the Search feature?

To use the search, proceed as follows:

- 1. Start the search dialog.**

Press Ctrl+h to start the search dialog.

- 2. Go to the OWL Search tab.**

- 3. Enter a keyword.**

Enter the keyword you want to search for in the text box.

- 4. (Optional) Select case sensitivity.**

By default, the search is case insensitive.

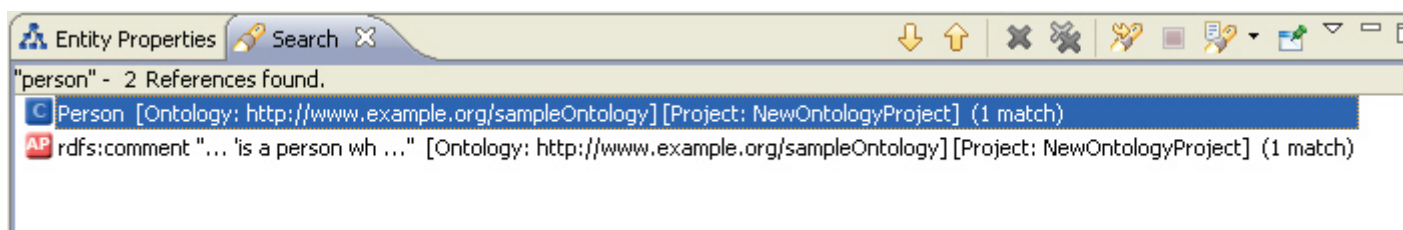
- 5. Specify search options.**

Select the check boxes to change the scope of elements to be searched.

- 6. Click Search.**

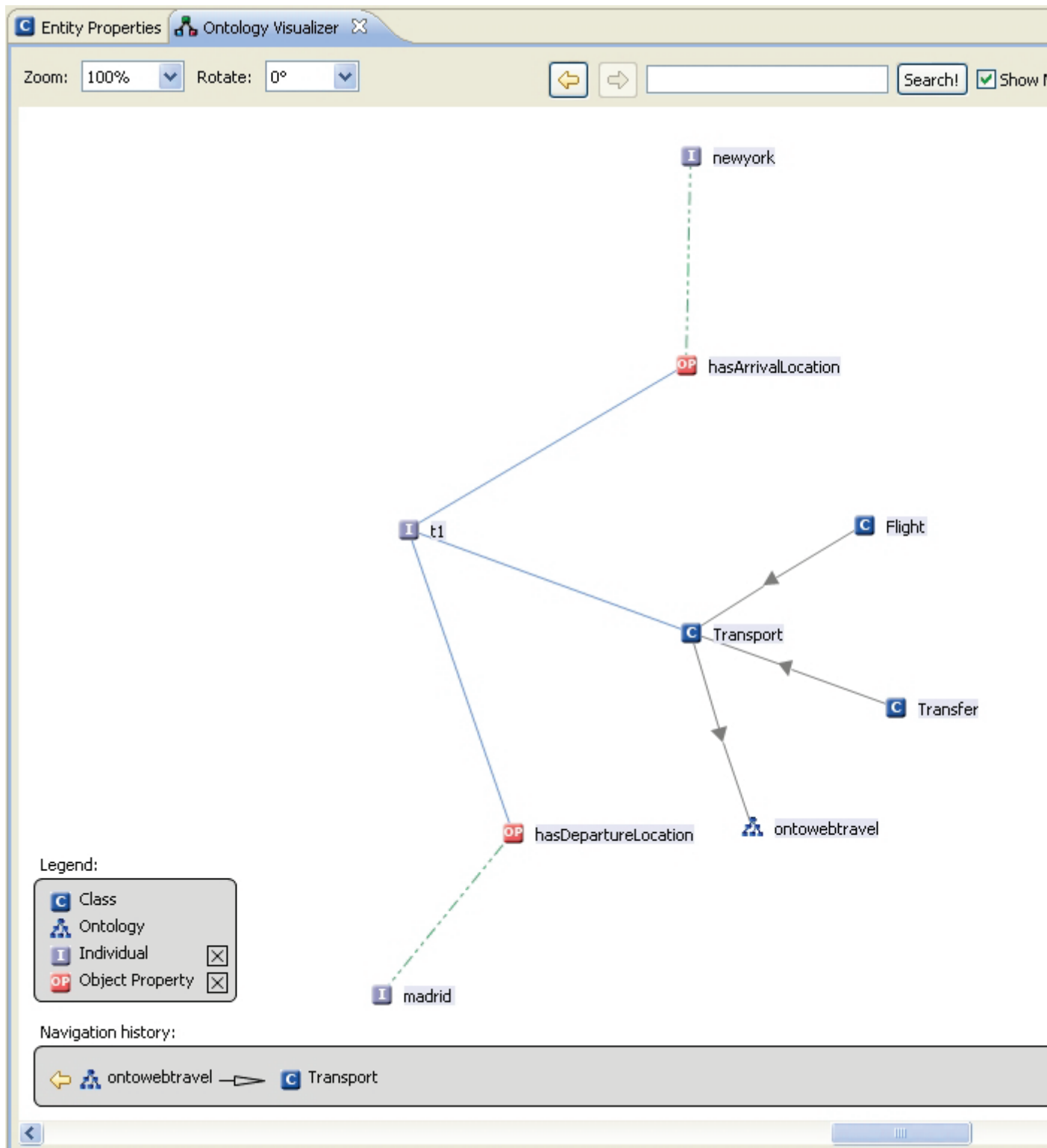
The search results will be displayed in the Search panel.

Double-click the entry to open the Entity Properties panel for the resource.



### 3.6. Ontology Visualizer

The Ontology Visualizer Panel displays the ontology and all elements of the hierarchy (classes, individuals, object properties) as a graph. You can open the view by selecting "Visualize Ontology" in the context menu of an ontology element or a class element in the Ontology Navigator.



### Functions:

- Display of classes, individuals and inheritance hierarchy by directed graphs
- Display of object properties, data properties and values by directed graphs
- Zoom function
- Rotation function
- Automatic layout for optimized view of the graph

### How to use the visualizer?

- Left-clicking an element in the Ontology Visualizer will refresh the visualization rendering with the currently selected element becoming its center.
- By right-clicking an element all of its connections will be added to the current visualization without shifting the focus to this element.
- In case there are many instances or classes displayed, the visualizer gets quite cluttered, i.e. you should avoid visualizing large ontologies with it.
- On the left side of the view a legend is displayed. Here you can activate or deactivate different ontology entities by selecting the check boxes.
- You can also enlarge or reduce the view of the graph. This can be achieved by changing the zoom value or by pressing Ctrl/+ to zoom in and Ctrl/- to zoom out.
- The context menu of the visualizer panel provides means for stopping and starting the layout engine, which helps when you want to manually arrange the nodes on the screen.
- Also, in the context menu you can find the functionality to rotate the whole graph, which sometimes improves the overall layout.

### Graph elements:

- Grey arrows:  
Represent the connection between super and subclasses or between a class and its ontology.
- Blue arrows:  
Represent the connection between classes and their individuals, and between individuals and their properties.
- Green arrows:  
The values of object properties are connected to the object property via green links.

## 3.7. Preferences

This page explains the preference settings of the OWL Editor. As for all other preferences you can access the preference dialog via Window >> Preferences. Here, in the "OWL Preferences" section you can customize the following features.

### OWL Preferences

Essentially, in the OWL Editor preferences you can configure three different things::

- Decide, whether the GUI should display also **imported axioms** or not
- Configure, whether the GUI should also display the complete **axioms** in Manchester Syntax in the GUI
- Request, to show a **toolbar for complex classes** in the Entity Properties panel for relevant text fields

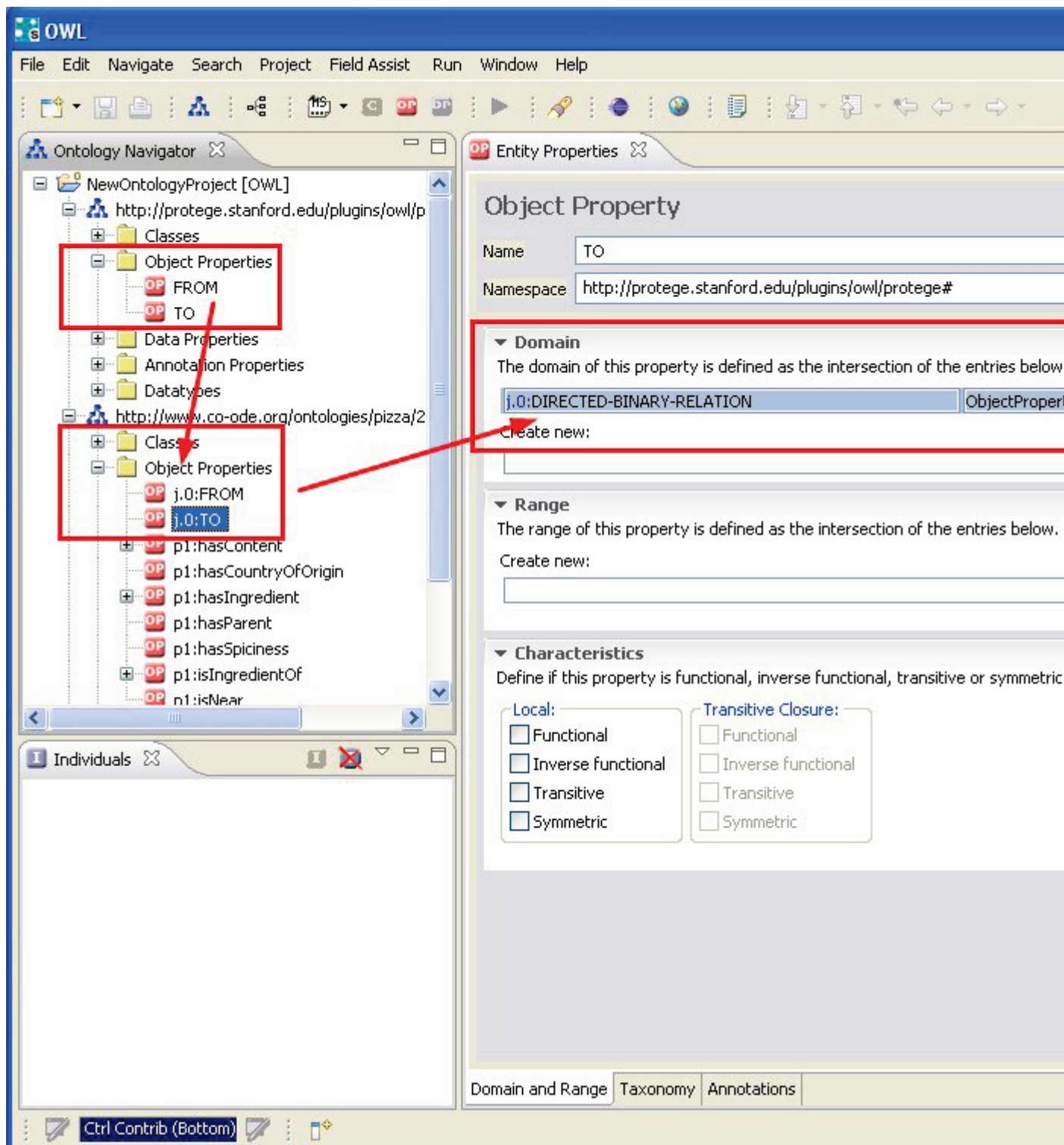
### Show Imported Axioms

Select the "Show Imported Axioms" check box to get the axioms displayed that are imported from other ontologies. In the Entity Properties Panel the rows containing such imported

axioms are highlighted with a light blue background. Additionally, a tool tip shows the source ontology the axiom was imported from.

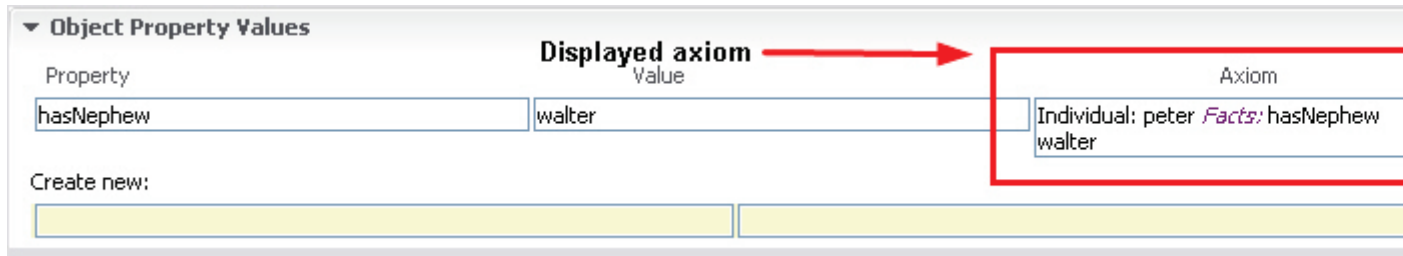
Note, that imported axioms that are highlighted by the GUI are immutable. They can only be changed at the ontology where they are defined.

Selecting the "Show Imported Axioms" option also shows all imported classes and properties in the Ontology Navigator. Sometimes it is useful to switch on the QNames or URI option for showing ontology labels to better distinguish between entities defined in different ontologies.



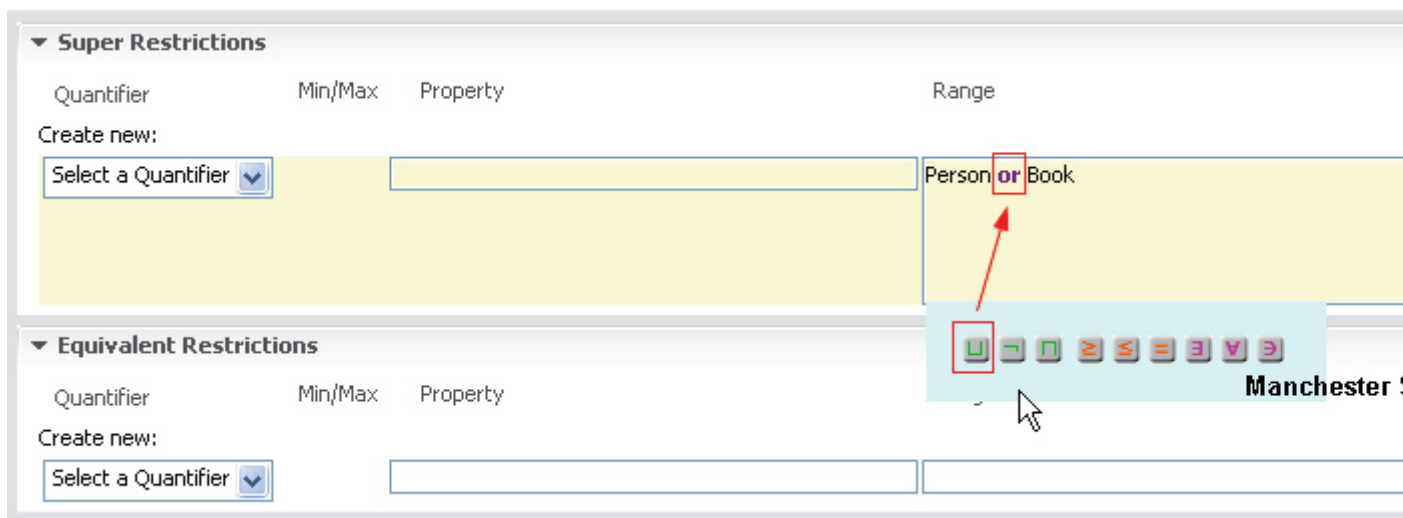
## Show Axioms in GUI

Select the check box "Show Axioms in GUI" to have the toolkit display the complete axioms in the Entity Properties panel. A new column will be added to the panel which contains the axioms in Manchester Syntax.



## Use Toolbar for Complex Class Editor

Selecting "Use Toolbar for Complex Class Editor" will enable the Manchester Syntax toolbar. This toolbar provides some buttons to enter keywords from the Manchester Syntax into text fields that can take complex class expressions. For more information on the Manchester Syntax and the toolbar, see Manchester Syntax.



## Language Settings

In addition to pure OWL settings you can specify the languages that can be selected in appropriate drop-down menus in the "Language Preferences" of the Preferences dialog.

OWL annotations such as `rdfs:label` or datatype property values can come in different languages, and thus are combined with a language attribute. The list of languages that is available when creating the annotations can be configured in this dialog box, i.e. you can add new languages or change their order. The default languages are:

- de
- en
- fr



This list is also used for selecting the languages for the entity labels that are displayed throughout the toolkit.

#### **4. How to use the OWL Editor?**

This section describes a number of frequently performed tasks using the OWL Editor. The following topics are included:

- **How to create an ontology project**
- **How to create an ontology?**
- **How to create a class?**
- **How to create an object property?**
- **How to create a datatype property**
- **How to create an annotation property?**
- **How to create an annotation?**
- **How to load an ontology?**
- **How to save an ontology?**
- **How to delete an Entity**

---

**Note** – If you want to use a special character in any Uniform Resource Identifier (URI) of your ontology, you must percent-encode it. Percent-encoding, also known as URL encoding, is a mechanism for encoding information in a Uniform Resource Identifier (URI). Percent-encoding a reserved character involves converting the character to its corresponding value in ASCII and then representing that value as a pair of hexadecimal digits. The digits, preceded by a percent sign ("%"), are then used in the URI in place of the reserved character. For example, if you want to use a whitespace in a resource, "%20" must be used instead of the raw space character (e.g., <http://example.org/sample#John%20Doe>).

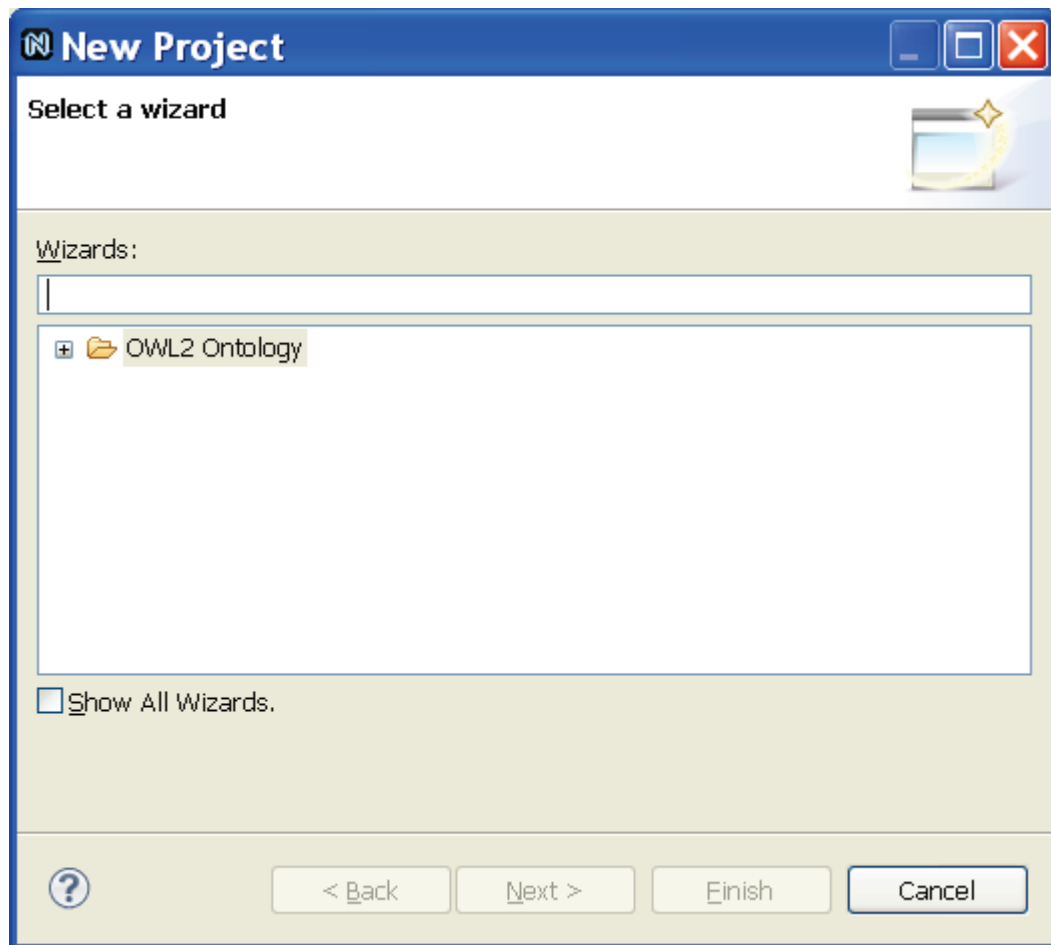
---

##### **4.1. How to create an OWL Project?**

Ontologies are managed within projects. To create an ontology project, proceed as follows:

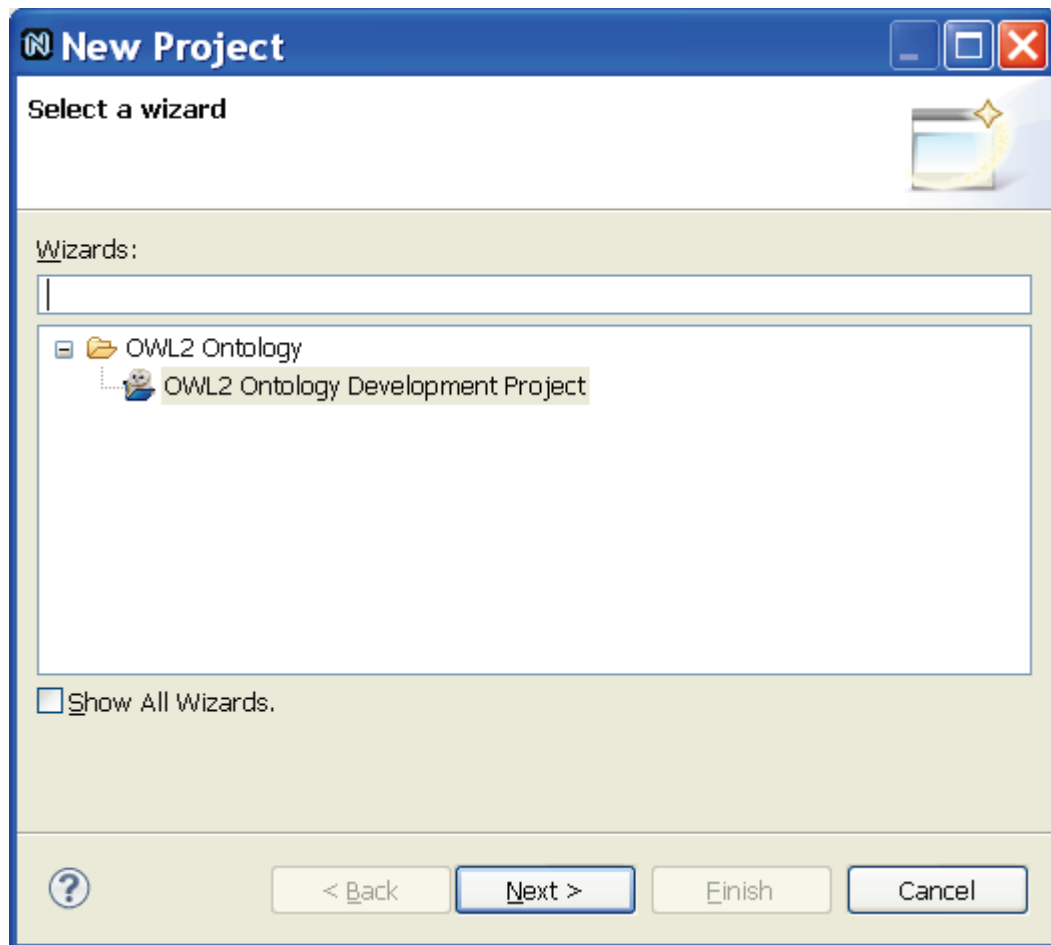
##### **1. Create an ontology project.**

Select File >> New >> Project. The New Project dialog opens.



## 2. Select Ontology Development Project.

Open the folder Ontology and select Ontology Development Project.



**3. Click Next.**

Enter the name of the project in the text box.

**4. Click Finish.**

The ontology project will be created. To discard your changes, click Cancel.

#### **4.2. How to create an Ontology?**

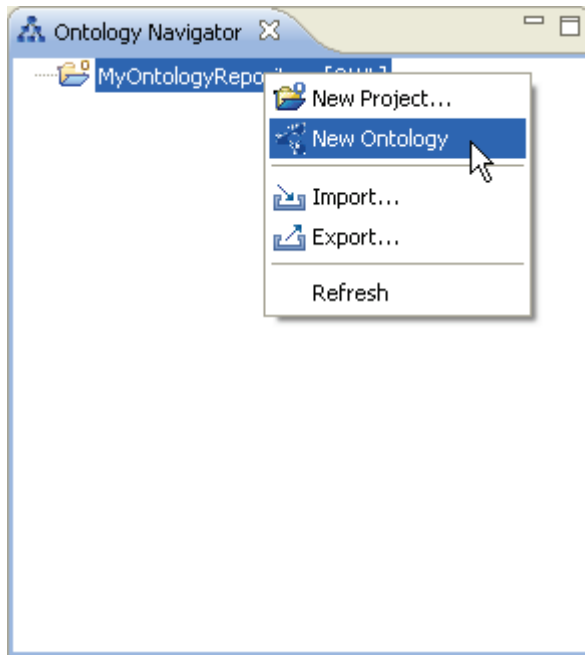
To create an OWL ontology, proceed as follows:

**1. Select an ontology project.**

Select an ontology project in the Ontology Navigator.

**2. Create an ontology.**

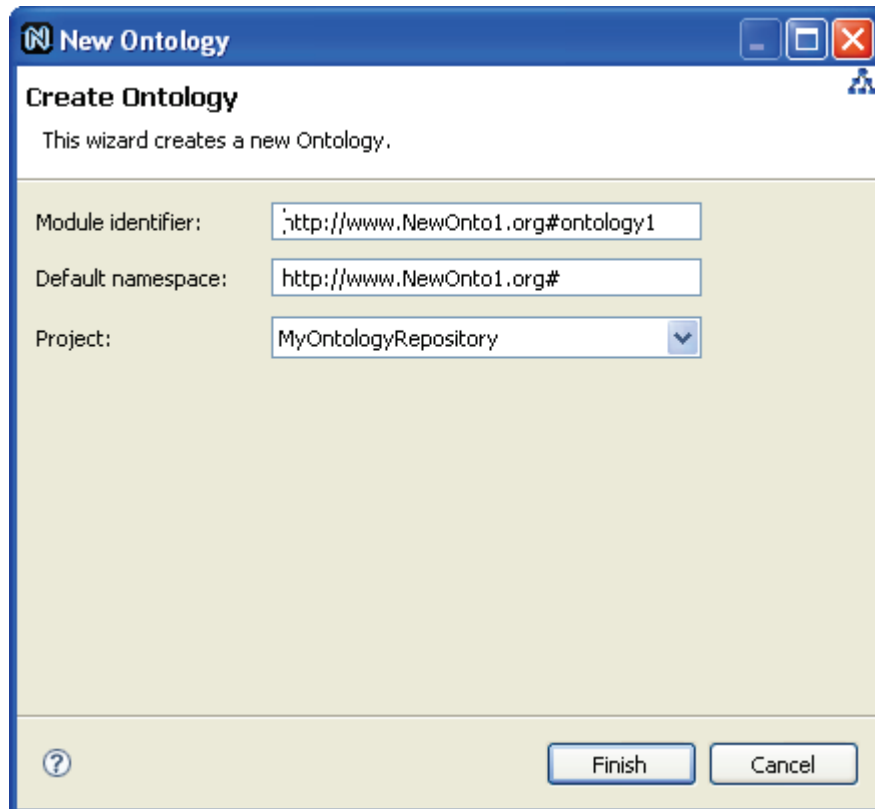
Right-click the ontology project and select New Ontology or press CTRL-N.



### 3. Enter appropriate information.

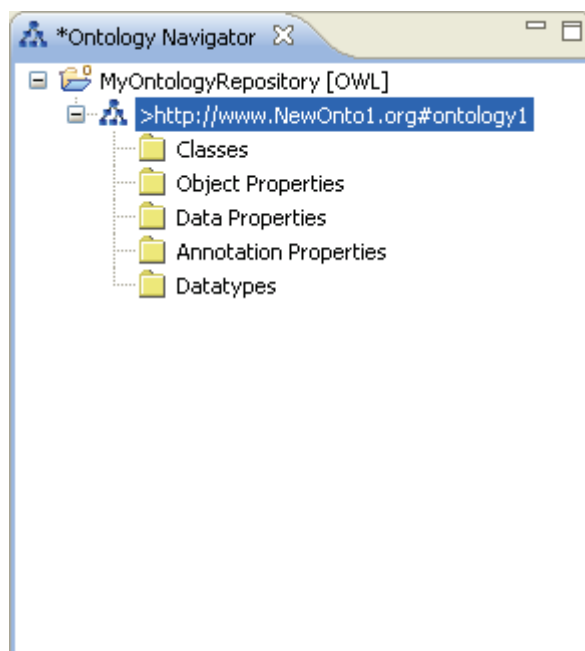
In the New Ontology dialog, enter the following information:

- **Ontology Identifier.** This ID is a URI which is used to uniquely denote the ontology.
- **Default Namespace.** The default namespace is used to bind unqualified names in the current ontology. It is good practice that the ontology ID and the default namespace are equivalent.
- **Project.** The project to which the ontology will be added.



#### 4. Click Finish.

The ontology will be created. To discard your changes, click Cancel.



### 4.3. How to create a Class?

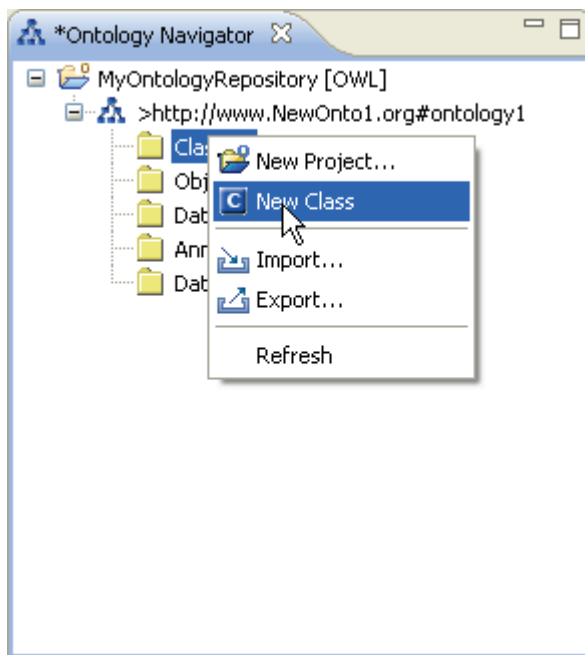
To create a new class, proceed as follows:

### 1. Select an ontology.

Select an ontology in the Ontology Navigator.

### 2. Select the Classes folder.

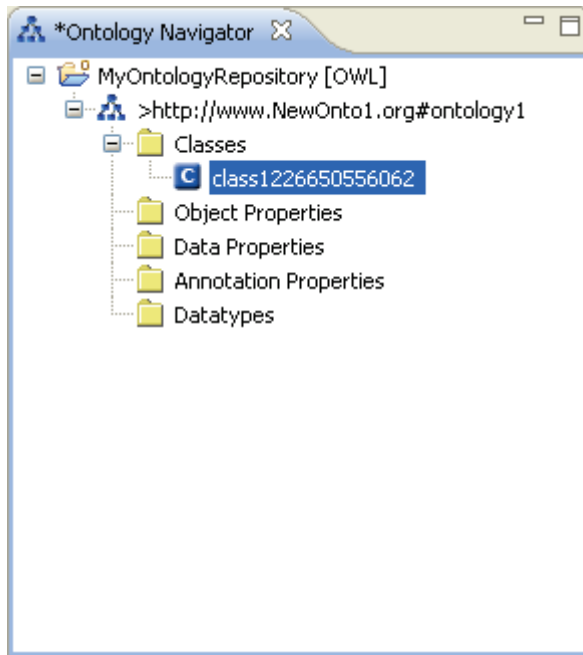
Right-click the Classes folder and select "New Class" or press CTRL-N.



Alternatively you can also do the same after selecting an already existing class. In this case the new class will be created as a subclass of the other one.

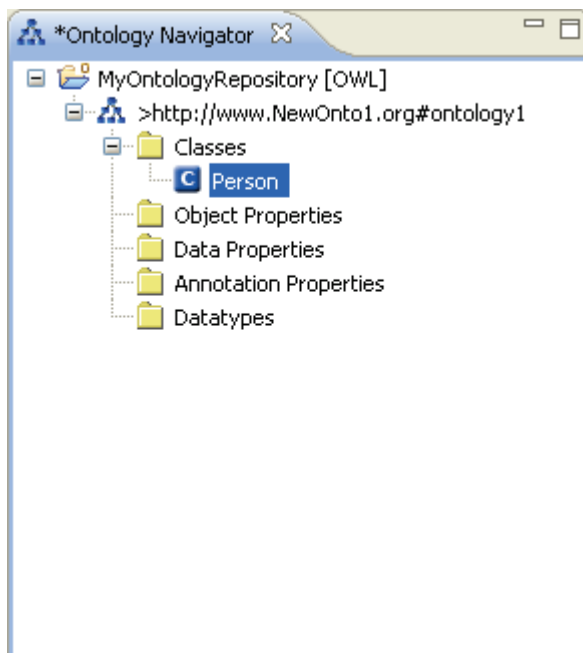
### 3. Rename the class.

The class will be created with an automatically generated name. Rename the class by changing its name in the Ontology Navigator.



#### 4. Press Return.

The class will be renamed.



#### 5. Edit the class.

The newly created class will be displayed in the Entity Properties panel. You can now continue to specify the characteristics of the class. For more information, see [OWL Classes](#).

### 4.4. How to create an Object Property?

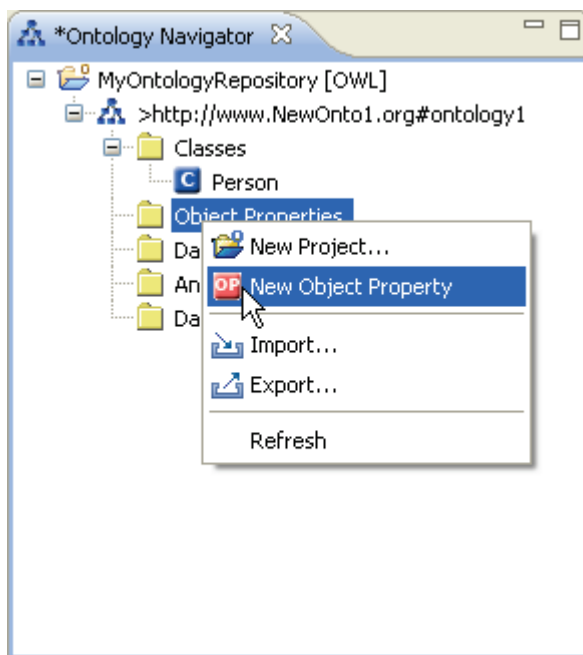
To create a new object property, proceed as follows:

- 1. Select an ontology.**

Select an ontology in the Ontology Navigator.

- 2. Select the Object Properties folder.**

Right-click the Object Properties folder and select "New Object Property" or press CTRL-N..

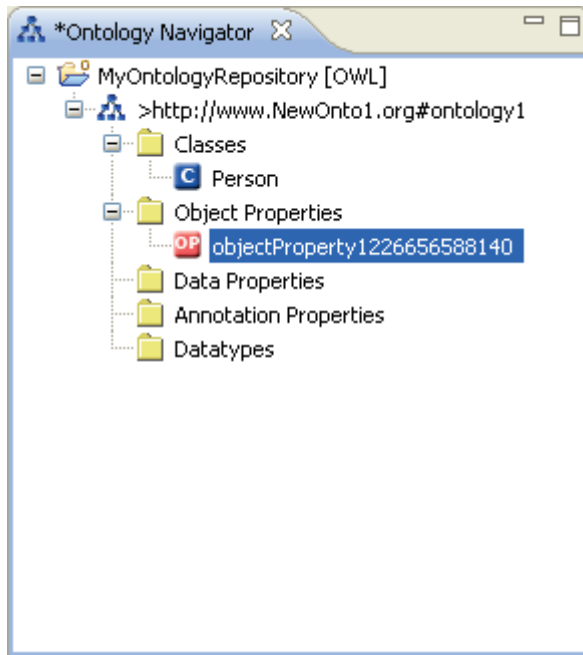


Alternatively you can also do the same after selecting an already existing object property. In this case the new property will be created as a sub-property of the other one.

- 3. Rename the object property.**

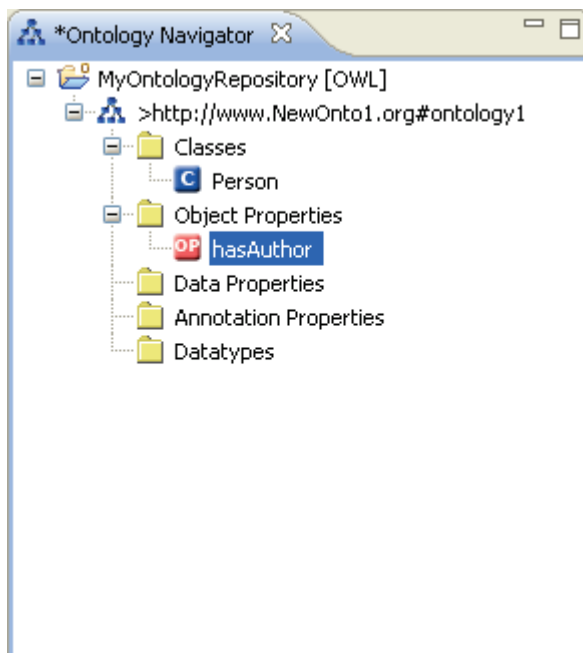
The object property will be created with an automatically generated name.  
Rename the object property by changing its name in the Ontology Navigator.





#### 4. Press Return.

The object property will be renamed.



#### 5. Edit the object property.

The newly created object property will be displayed in the Entity Properties panel. You can now continue to specify the characteristics of the object property. For more information, see OWL Object Properties.

### 4.5. How to create a Datatype Property?

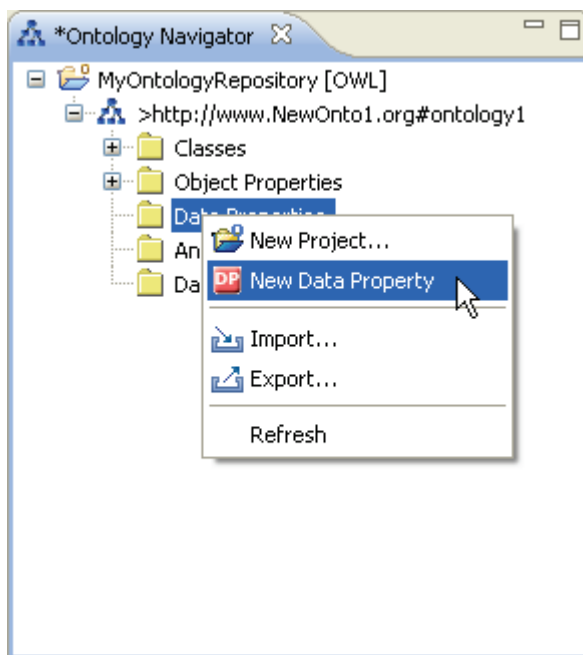
To create a new datatype property, proceed as follows:

- 1. Select an ontology.**

Select an ontology in the Ontology Navigator.

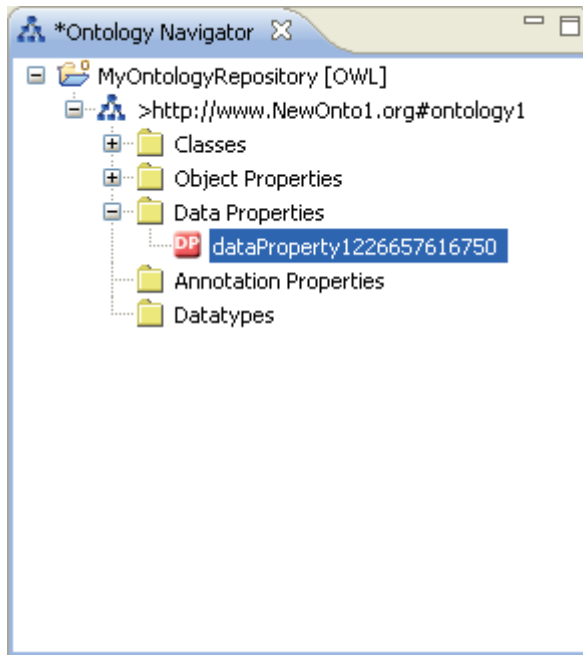
- 2. Select the Data Properties folder.**

Right-click the Data Properties folder and select "New Data Property" or press CTRL-N.



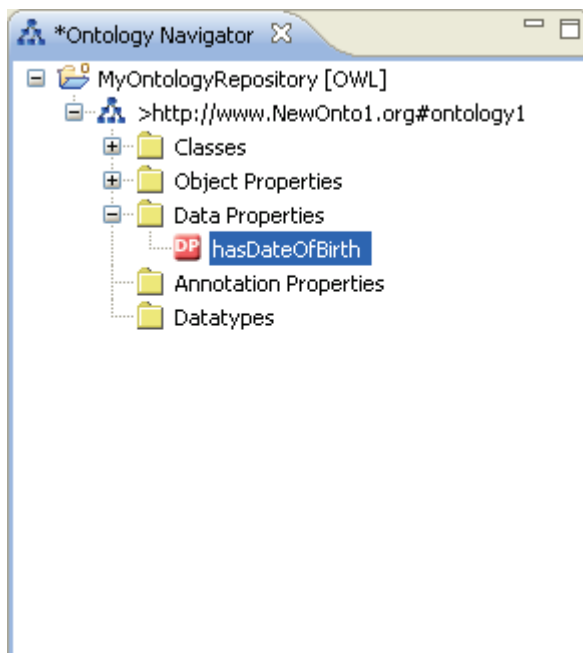
- 3. Rename the datatype property.**

The datatype property will be created with an automatically generated name. Rename the datatype property by changing its name in the Ontology Navigator.



#### 4. Press Return.

The datatype property will be renamed.



#### 5. Edit the datatype property.

The newly created datatype property will be displayed in the Entity Properties panel. You can now continue to specify the characteristics of the datatype property. For more information, see [OWL Datatypes Properties](#).

### 4.6. Create an Annotation Property

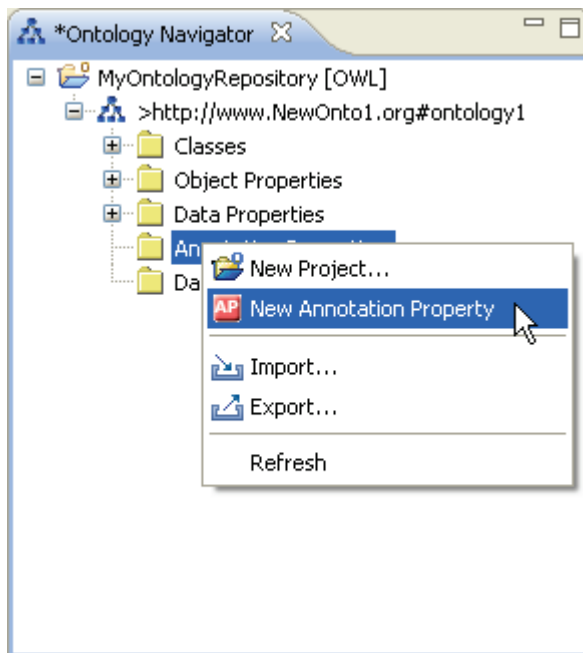
To create an annotation property, proceed as follows:

- 1. Select an ontology.**

Select an ontology in the Ontology Navigator.

- 2. Select the Annotation Properties folder.**

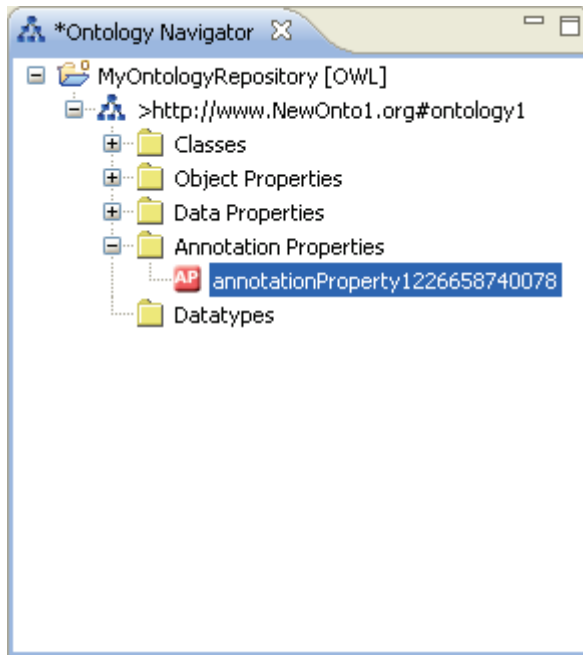
Right-click the Annotation Properties folder and select "New Annotation Property", hit CTRL-N, or select the "New Annotation Property" icon in the toolbar.



Alternatively you can also do the same after selecting an already existing annotation property. In this case the new property will be created as a sub-property of the other one.

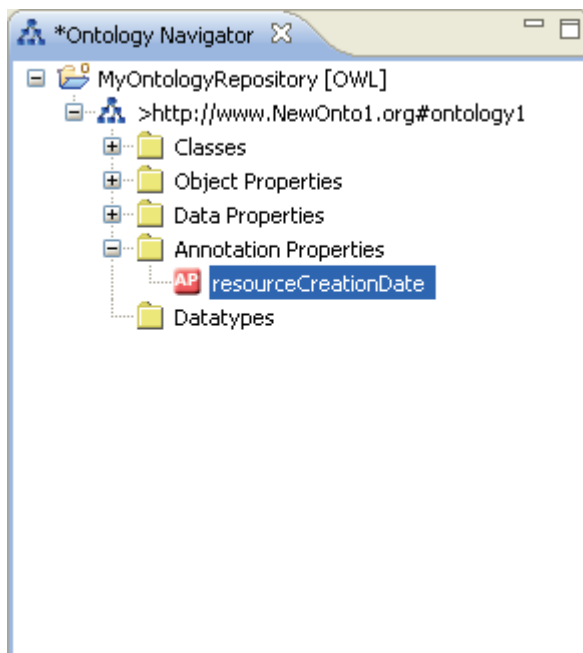
- 3. Rename the annotation property.**

The annotation property will be created with an automatically generated name. Rename the annotation property by changing its name in the Ontology Navigator.



**4. Press Return.**

The annotation property will be renamed.



**5. Edit the annotation property.**

The newly created annotation property will be displayed in the Entity Properties panel. You can now continue to specify the characteristics of the annotation property. For more information, see [OWL Annotation Properties](#).

**4.7. How to create an Annotation?**

Usually, annotations are used to add information to entities of the ontology. They contain useful information such as who is the creator or what are the readable labels and comments for single elements of the ontology.

Entity Properties

Identifier

URI: <http://www.example.org/sample#Person>

Annotations

Annotation Property	Value	Type	Language
rdfs:label	personne	rdfs:Literal	fr

Create new:

Class Restrictions Taxonomy Annotations

To create an annotation, proceed as follows:

- 1. Select an entity in the Ontology Navigator or the Individuals panel.**

The entity will be displayed in the Entity Properties panel.

- 2. Go to the Annotations tab.**

The annotations will be displayed.

- 3. Create an annotation.**

The bottom row of this view is labeled "Create New". Fill in appropriate information in these text boxes. You can also use the auto-complete feature of the OWL Editor.

- a. Enter an annotation property.**

OWL provides a number of built-in annotation properties, e.g.:

- `rdfs:label`
- `rdfs:comment`
- `rdfs:seeAlso`

- `rdfs:isDefinedBy`
- `owl:versionInfo`

In addition, you can also create your own annotation properties. For more information, see Annotation Properties.

- b. Enter a value for the annotation.**
- c. Enter a type for the annotation.**

The type is either an XML Schema datatype or an RDFS Literal.

- d. Select a language code for the annotation.**

The language code is only available for type `rdfs:Literal`. If you select another type, the language code will be removed. Likewise, the type will be reset to `rdfs:Literal` once you select a language code for the annotation.

#### **4. Click Add.**

The annotation will be created. To discard your changes, click Cancel.

### **4.8. How to load an ontology?**

To load an ontology into the NeOn Toolkit, proceed as follows:

#### **1. Select an ontology project.**

Select an ontology project in the Ontology Navigator.

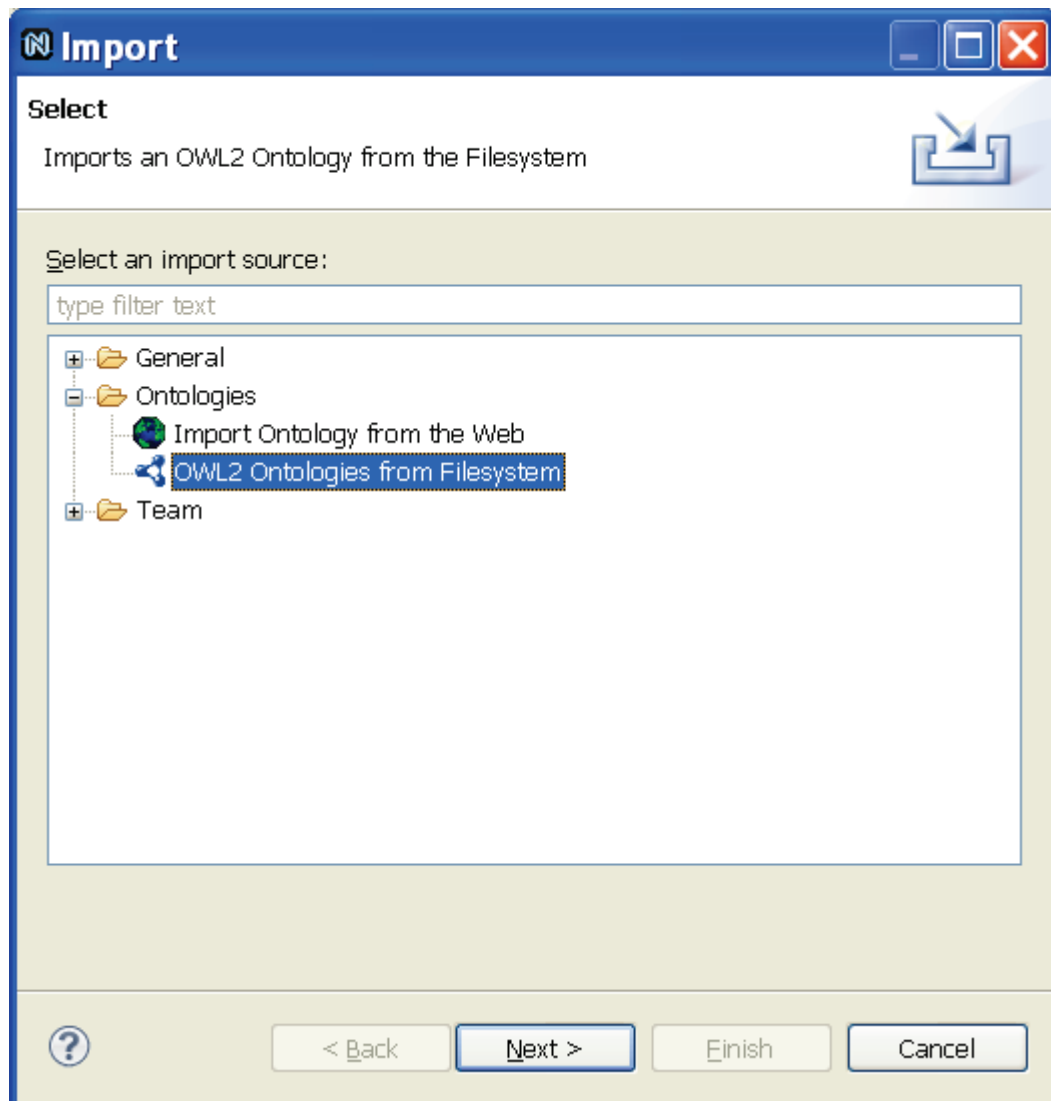
#### **2. Import an ontology.**

Right-click the ontology project and select Import. The New Ontology dialog opens.

#### **3. Select an import source.**

Locate the folder "Ontologies" and select an import source.  
You can choose between the following import methods:

- **Filesystem.** Import an OWL2 ontology from the filesystem.
- **The Web.** Import an ontology directly from the Web.



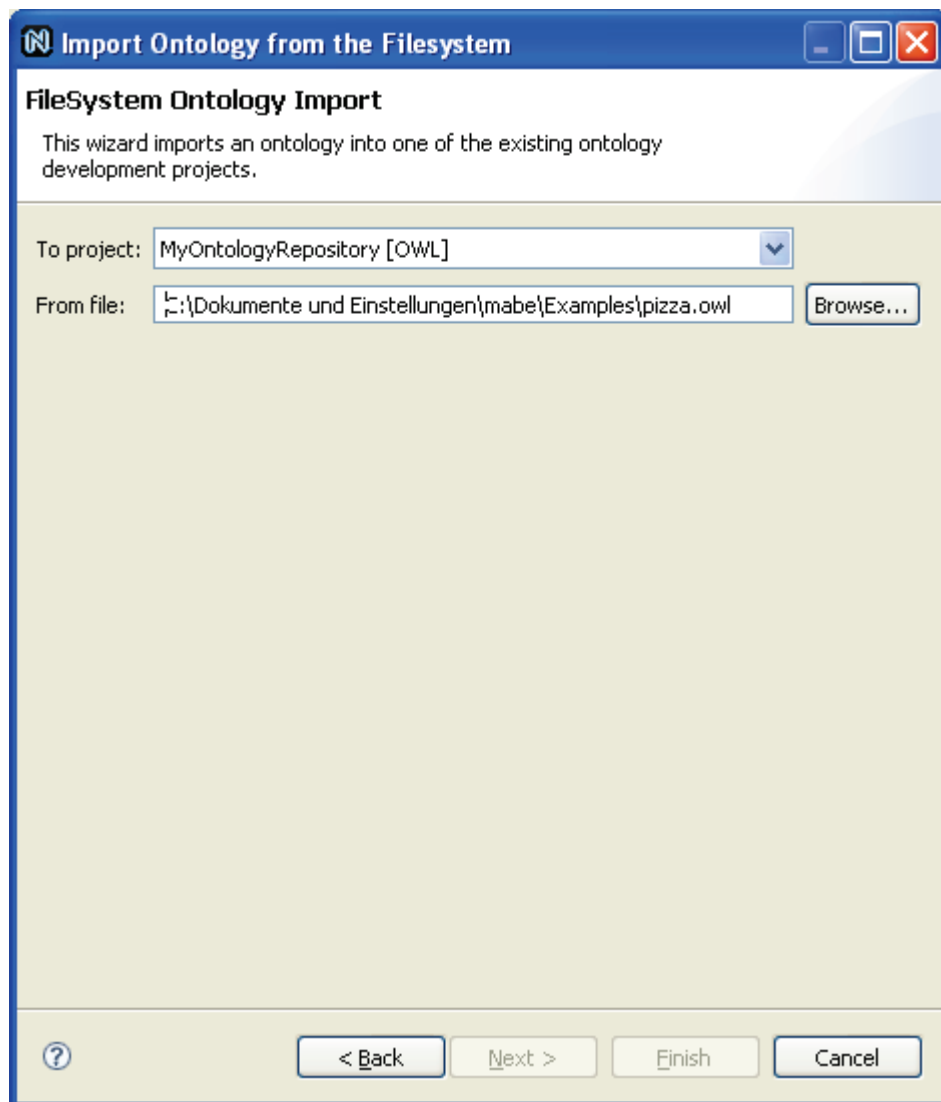
#### 4. Enter appropriate information.

In case you select the Web import a dialog appears which lets you choose an OWL project to import into and lets you enter the URL of the ontology to load.

In case you select the file system import another dialog appears which lets you enter the following information:

- **To Project.** Select the project to which you want to import the ontology.
- **From File.** Select the file you want to import. Its best to use the "Browse" button which lets you select the file from your file system with a file chooser dialog.

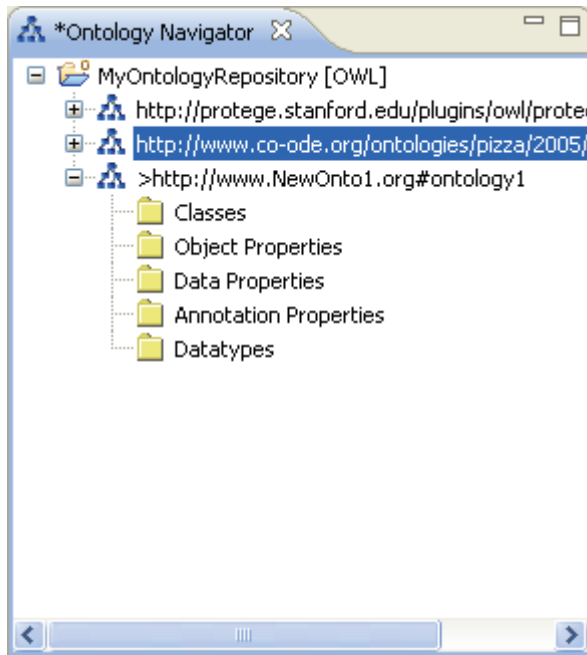




The file chooser also allows for selecting multiple ontologies at the same time using the Shift and CTRL keys. This is a handy mechanism to load ontologies into the NeOn Toolkit which import each other and should not be fetched from the web.

##### **5. Click Finish.**

After returning from the file chooser click on the "Finish" button to load the selected ontology. To not load it, click "Cancel".



---

**Note** – When the loaded ontology uses owl:import statements to import other ontologies, they will also be fetched by this mechanism and loaded into memory. If an ontology with the requested ID already exists in the project it will not be fetched again.

---

#### 4.9. How to save an Ontology?

The Export dialog allows you to save an ontology into a different file (and possibly in a different format).

To export an ontology, proceed as follows:

- 1. Select an ontology.**

Select an ontology in the Ontology Navigator.

- 2. Export an ontology.**

Right-click the ontology and select Export. The Export wizard opens. Choose the "OWL2 Ontology to Filesystem" option in the Ontologies section.

- 3. Specify the ontology to be exported.**

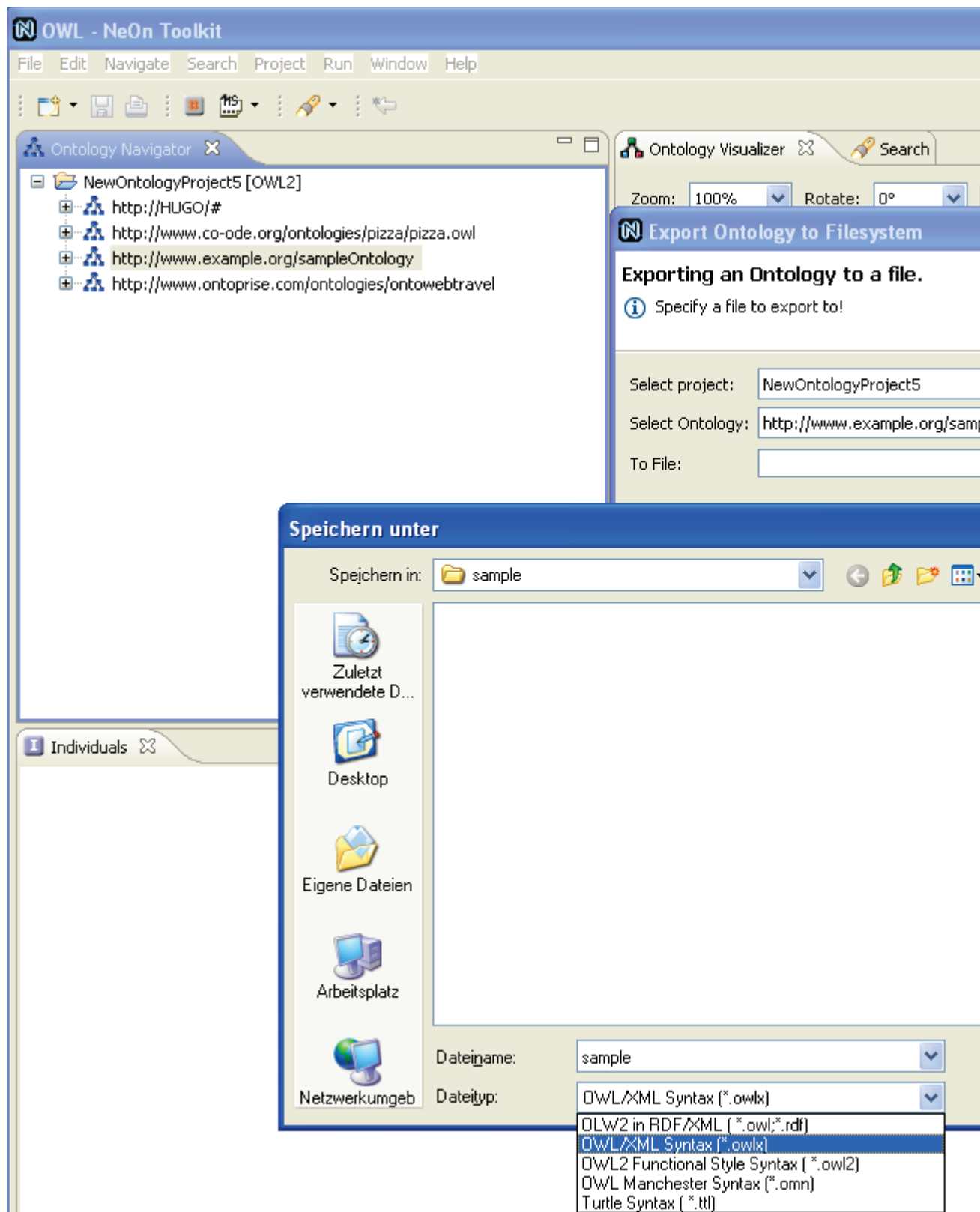
Enter the following information:

- **Select Project.** Select the project from which you want to export the ontology.
- **Select Ontology.** Select the ontology you want to export.

- **To File.** Select a file name and file type of the ontology to be exported. It is best to use the "Browse" button to open the file chooser.

In the file chooser a variety of file types are available:

- OWL/RDF – The official OWL2 W3C recommendation in RDF/XML.
- OWLX – An OWL2-XML presentation of the official OWL 2 W3C recommendation.
- OWL2 – The functional syntax of the official OWL 2 W3C recommendation.
- OMN – The Manchester syntax for OWL 2.
- TTL – A Turtle serialisation format for the RDF graph of the OWL 2 ontology.



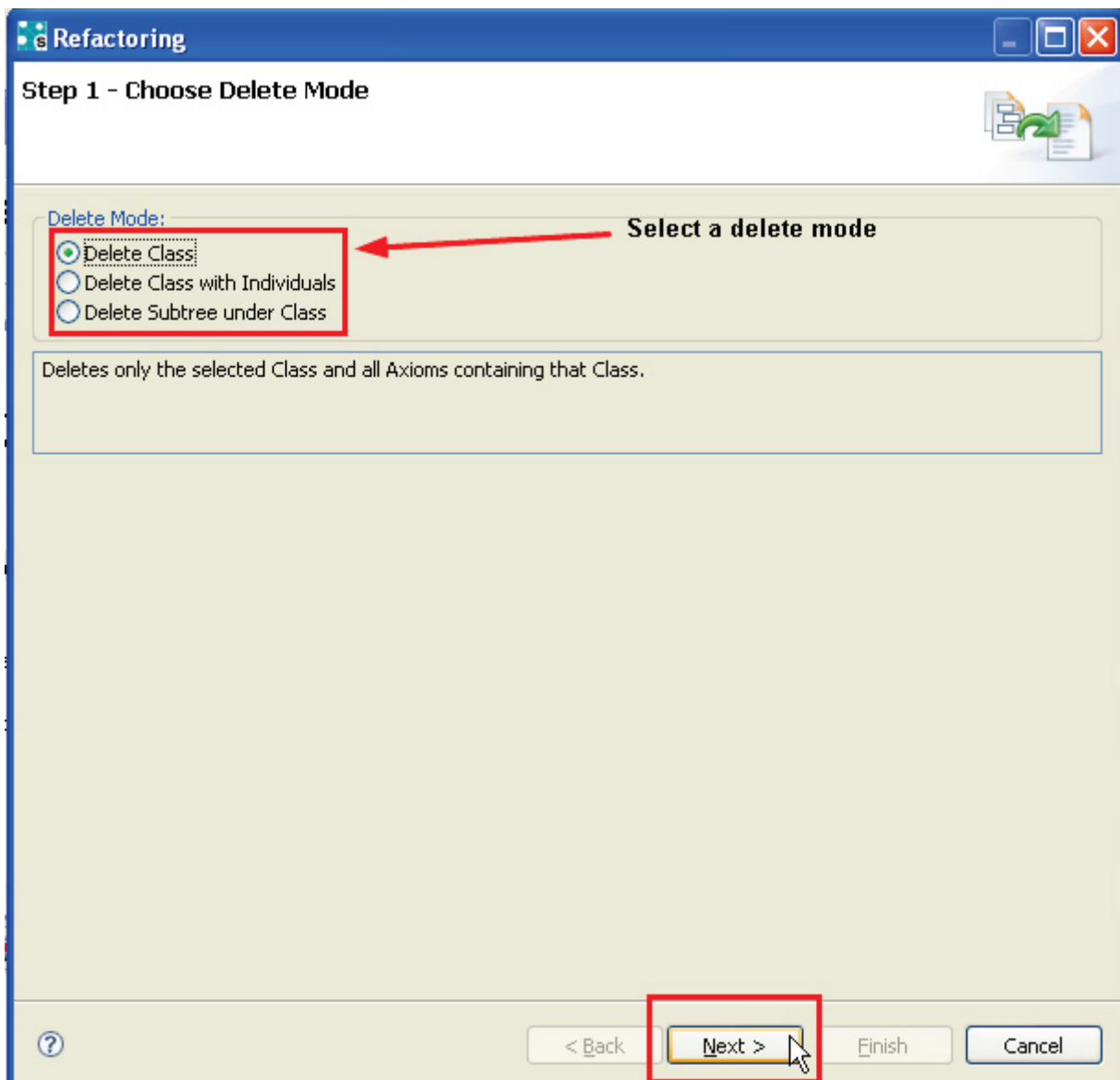
#### 4. Click Finish.

The ontology will be exported. To discard your changes, click Cancel.

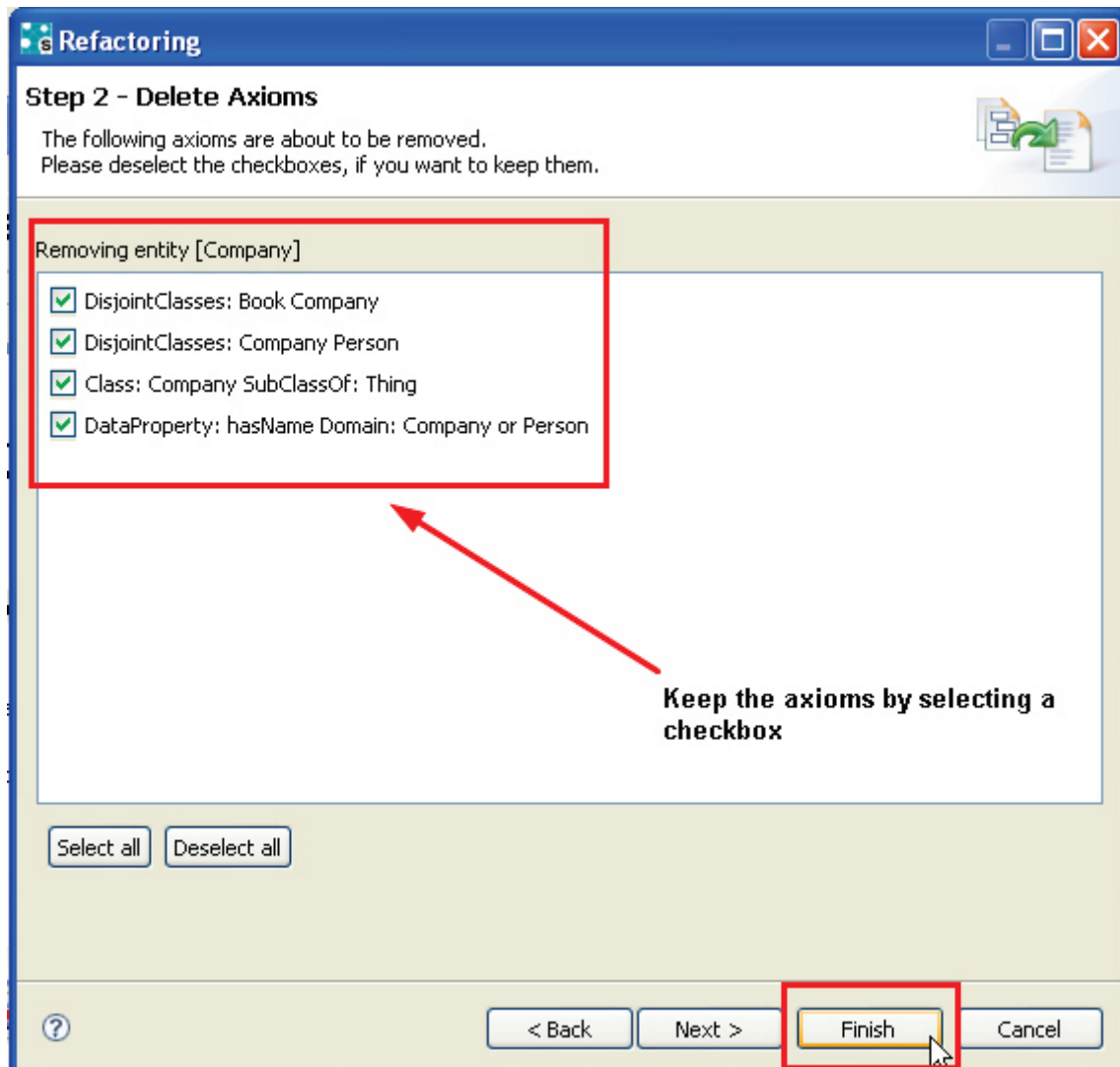
#### 4.10. How to delete an Entity?

If you want to delete a resource such as a class (object property, data property, etc.) from the ontology, you can use the Delete dialog which guides you through the process. For example, if you want to delete a class, three options are available:

- **Delete Class.** Deletes only the selected Class and all axioms containing this class.
- **Delete Class with Individuals.** Deletes the selected class, all individuals of that class and all axioms containing either the class or one of the individuals.
- **Delete Subtree under Classes.** Deletes the selected class with all of its Subclasses, as well as all individuals and all axioms containing one of these classes or individuals.



However, the axioms are not removed immediately. For every single axiom, you can still decide whether you want to remove or keep it. In the next step of the Delete dialog you are prompted to specify which axioms you want to keep.



Sometimes it is even necessary to add new axioms to not change the semantics of the ontology. For example, suppose you have an axiom stating that classes A, B and C are equivalent and you want to remove the information that A and B are equivalent. By simply removing the axiom you would also lose the information that A and C are equivalent as well as that B and C are equivalent. Therefore, you have to re-add those two axioms. This is realized in the third step of the Delete dialog.

The NeOn Toolkit discovers these cases and proposes axioms that could/should be added to the ontology again. Effectively this Wizard proposes not to remove complete axioms but rather to remove the occurrence of the entity-to-be-deleted from them.

## 5. Online References

This page contains useful online references with respect to the contents of the OWL Editor guide.

### OWL 2

- [Homepage of the W3C OWL2 Working Group](#)
- [Primer](#)

- [Quick reference Card for all OWL2 features](#)
- [Structural Specification](#)
- [New Features in OWL2 vs. OWL](#)
- [Manchester Syntax Web site.](#)

## **Datatypes**

- [XML Schema Part 2: Datatypes Second Edition](#)

## **NeOn Toolkit**

- [NeOn Toolkit portal](#)