Protégé-OWL and the Semantic Web

8th Int. Protégé Conference, Madrid, July 2005

Holger Knublauch
The Semantic Web

Vision

Standards

W3C

RDF

OWL

Technology

Jena

Racer

protégé
Example Scenario
Traditional Web Architecture

Cruise Operator

Boat Rental

Hotel

Submission Forms

Travel Planning Portal Database

Accommodation Portal Database

Query Interfaces

End User

Manual actions

Automated actions
Semantic Web Architecture

Cruise Operator

Service Metadata

Crawler Agent

Travel Services Database and Agent

Query Interface

User Profile

End User

Manual actions

Automated actions

Crawler Agent

Hotel

Boat Rental

OWL

OWL

OWL
Semantic Web Evolution

- Service Metadata
  - OWL
  - XHTML
  - Travelogue
  - Personal Profile
  - Friend-of-a-Friend
  - RDF
  - Owl
  - Pet Ontology
  - Animal Ontology
  - Food Ontology

- Cruise Operator
  - OWL

- Boat Rental
  - OWL

- Hotel
  - OWL
  - Geography
  - Image Annotation
  - Restaurant
  - Recommendations
  - Owl
  - Cooking Recipes
Software Architecture (1)

- **Semantic Web Layer**
  - HeliBungee.owl
  - ...
  - ActivityX.owl

- **Internal Layer**
  - Travel.owl
  - Customer.owl
  - Ontology representation as Java objects

- **Core Ontologies**
  - Reasoners (OWL DL, SWRL, ...)

- **Web Service Interface (WSDL)**
  - Web Service, Control Logic (Java Code)

- **End-User Interface (JSP)**
Protege-OWL
Protege-OWL Classification
Protege-OWL Individuals
Working with Multiple Files

What is the Active Sub-Ontology?

Your ontology can import other ontologies, and either of them can be edited and saved. If you want to make an imported ontology editable, a local file needs to be specified for it. Then, you need to declare that the import is editable. All editable files will be overwritten when you save your project. The active file is the one where the next changes are being added into. For example, if you add a new property value to a resource, then this new value will be added to the active file.
Protege-OWL for RDF(S)
Protege-OWL for SWRL

<table>
<thead>
<tr>
<th>Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Def-hasAunt</td>
<td>( \text{hasParent}(?x, ?y) \land \text{hasSister}(?y, ?z) \rightarrow \text{hasAunt}(?x, ?z) )</td>
</tr>
<tr>
<td>Def-hasBrother</td>
<td>( \text{hasSibling}(?x, ?y) \land \text{Man}(?y) \rightarrow \text{hasBrother}(?x, ?y) )</td>
</tr>
<tr>
<td>Def-hasDaughter</td>
<td>( \text{hasChild}(?x, ?y) \land \text{Woman}(?x) \rightarrow \text{hasDaughter}(?x, ?y) )</td>
</tr>
<tr>
<td>Def-hasFather</td>
<td>( \text{hasParent}(?x, ?y) \land \text{Man}(?y) \rightarrow \text{hasFather}(?x, ?y) )</td>
</tr>
<tr>
<td>Def-hasMother</td>
<td>( \text{hasParent}(?x, ?y) \land \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y) )</td>
</tr>
<tr>
<td>Def-hasNephew</td>
<td>( \text{hasSibling}(?x, ?y) \land \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z) )</td>
</tr>
<tr>
<td>Def-hasNiece</td>
<td>( \text{hasSibling}(?x, ?y) \land \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z) )</td>
</tr>
<tr>
<td>Def-hasParent</td>
<td>( \text{hasParent}(?x, ?y) \land \text{Man}(?y) \rightarrow \text{hasParent}(?x, ?y) )</td>
</tr>
<tr>
<td>Def-hasSibling</td>
<td>( \text{hasChild}(?x, ?y) \land \text{hasChild}(?y, ?z) \land \text{differentFrom}(?x, ?z) \rightarrow \text{hasSibling}(?x, ?z) )</td>
</tr>
<tr>
<td>Def-hasSister</td>
<td>( \text{hasSibling}(?x, ?y) \land \text{Woman}(?y) \rightarrow \text{hasSister}(?x, ?y) )</td>
</tr>
<tr>
<td>Def-hasSon</td>
<td>( \text{hasChild}(?x, ?y) \land \text{Man}(?y) \rightarrow \text{hasSon}(?x, ?y) )</td>
</tr>
<tr>
<td>Def-hasUncle</td>
<td>( \text{hasParent}(?x, ?y) \land \text{hasBrother}(?y, ?z) \rightarrow \text{hasUncle}(?x, ?z) )</td>
</tr>
</tbody>
</table>
Protege-OWL Plugins
JenaOWLMpdel owlModel = ProtegeOWL.createJenaOWLMpdel();

RDFSNamedClass personClass =
   owlModel.createRDFSNamedClass("Person");
RDFProperty ageProperty = owlModel.createRDFProperty("age");
ageProperty.setRange(owlModel.getXSDInt());
ageProperty.setDomain(personClass);

RDFIndividual individual = personClass.createRDFIndividual("Holger");
individual.setPropertyValue(ageProperty, new Integer(34));
Java Code Generation

RDFIndividual individual = personClass.createRDFIndividual("Holger");
individual.setPropertyValue(ageProperty, new Integer(34));

Person individual = factory.createPerson("Holger");
individual.setAge(34);
Software Architecture (2)
Configuring Protege
Protege-Driven Development

Domain Modeling

Incremental Code Generation

Model

View

Control

Programming
Further Reading (1)

http://www.co-ode.org
Further Reading (2)

The Protégé-OWL API - Programmer’s Guide

by Holger Knublauch (last change June 02, 2005)

Contents

- Overview
- Installation & Getting Started
- Basics
  - Working with OWL Models
  - Names, Namespace prefixes and URLs
  - Understanding the Model Interfaces
  - Creating Named Classes and Individuals
  - Using Datatype Properties and Datatype Values
  - Using Object Properties to Build Relationships between Resources
  - Working with References to External/Untyped Resources
  - Property Domains
- Advanced Class Definitions (OWL DL)
  - Restrictions
  - Logical Class Definitions (Unions, Intersections, Complements)
  - Enumerated Classes
  - Creating Defined Classes
- Advanced Topics
  - Querying the OWL Model
  - RDF(S) and OWL
  - Reacting on Changes using Listeners
  - Loading and Saving Files
  - Working with Multi-File Projects and TripleStores
  - Working with Terns Models
- Protégé Programming
  - Protégé-OWL and the Core Protégé API
  - Changes from the old Protégé (OWL) API
  - Protégé User Interface Programming
  - Protégé Plugin Development
- Building Semantic Web Applications