`owl2onto.cls`: Marking up OWL2 Ontologies in sTeX.[*]

Michael Kohlhase
Jacobs University, Bremen
`http://kwarc.info/kohlhase`

August 15, 2010

Abstract

The `owl2onto` class allows mark up OWL2 Ontologies in sTeX and generate OWL2-XML from them via the LaTeXML system.

Contents

[*]Version ? (last revised ?)

Experimental! do not use!

1 The User Interface

EdNote(1) The owl2onto package allows mark up ontology-based Metadata in LaTeX documents that can be harvested by automated tools or exported to PDF.[1]

The main idea behind the owl2onto class and package is to think of (documented) ontologies as documents, which present the knowledge behind ontology informally to the (human) reader and at the same time contain enough (hidden) information so that a formal ontology can be generated from them.

1.1 Package Options

showmeta The owl2onto package takes a single option: showmeta. If this is set, then the metadata keys are shown (see [**Kohlhase:metakeys:ctan**] for details and customization options).

1.2 Ontologies

ontology The owl2onto class provides the ontology environment environment to declare OWL classes; it takes the place of the usual document environment[1], but its optional argument provides KeyVal attributes that allow to mark up the semantic aspects of the class, see Figure 1. In this example, we only have a single class declaration.

```
\documentclass{owl2onto}
\begin{ontology}[id=foaf,baseURI=http://xmlns.com/foaf/0.1/]
\declaration[type=Class,name=Agent,cseq=agent]
  {An agent (eg. person, group, software or physical artifact).}
\end{ontology}
```

Class: Agent
An agent (eg. person, group, software or physical artifact).

Example 1: A simple Ontology in LaTeX and its presentation

1.3 Declarations

\declaration type In general the \declaration macro can be used to declare the objects of various types in an ontology. The type key is used to specify this, its values range over

[1] EdNote: continue

[1] Admittedly, it is somewhat unconventional to use the document environment for this, but this is the best way to ensure that we an OWL/XML document with a single document root.

EdNote(2) the set[2]. The name attribute specifies the name of the declared object. This information is used in the XML generation; see Figure ?? for the result of generating XML from Figure 1. Finally, the cseq key allows to specify a command sequence for the object, which an be used in properties.

name

cseq

```
<?xml version="1.0"?>
<!--This OWL2 ontology is generated from an sTeX-encoded one via LaTeXML,
    you may want to reconsider editing it.-->
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
         xmlns:stex="http://kwarc.info/ns/sTeX"
         xmlns:omdoc="http://omdoc.org/ns"
         ontologyIRI="http://xmlns.com/foaf/0.1/"
         xml:base="http://xmlns.com/foaf/0.1//"
         xml:id="foaf">
  <Import>owl2.omdoc#OWL2</Import>
  <Declaration stex:srcref="test.tex#textrange(from=5;0,to=5;64)">
    <Class IRI="Agent"/>
  </Declaration>
  <AnnotationAssertion>
    <AnnotationProperty IRI="http://www.w3.org/2000/01/rdf-schema#comment"/>
    <IRI>Agent</IRI>
    <Literal>An agent (eg. person, group, software or physical artifact).</Literal>
  </AnnotationAssertion>
</Ontology>
```

Example 2: The OWL/XML generated from Figure 1

1.4 Properties

\axiom

EdNote(3) The properties of the declared objects can be stated via the \axiom macro. Its first argument is an OWL formula marked up in prefix notation[3], and the second one a natural language explanation.

```
%% generated with the docstrip utility.
```

Axiom: *agent*$\|$*document*

Agents are not Documents

```
  <DisjointClasses>
    <Annotation>
      <AnnotationProperty IRI="http://www.w3.org/2000/01/rdf-schema"/>
      <Literal>Agents are not Documents</Literal>
    </Annotation>
    <Class IRI="Agent"/>
    <Class IRI="Document"/>
  </DisjointClasses>
```

Example 3: An Axiom

2 The Implementation

The functionality is spread over the owl2onto class and package. The class provides the document environment and the ontology element corresponds to it,

[2]EdNote: give a list of all of them.

[3]EdNote: explain this better

whereas the package provides the concrete functionality.

owl2onto.dtx generates four files: owl2onto.cls (all the code between ⟨*cls⟩ and ⟨/cls⟩), owl2onto.sty (between ⟨*package⟩ and ⟨/package⟩) and their LaTeXML bindings (between ⟨*ltxml.cls⟩ and ⟨/ltxml.cls⟩ and ⟨*ltxml.sty⟩ and ⟨/ltxml.sty⟩ respetively). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

2.1 The owl2onto Class

We first define the owl2onto class, which on the LaTeX side just calls the article class.

```
⟨*cls⟩
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
\RequirePackage{modules}
\RequirePackage{owl2onto}
⟨/cls⟩
⟨*cltxml⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Global;
use LaTeXML::Package;
DeclareOption(undef,sub {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))));
ProcessOptions();
LoadClass('article');
RequirePackage('modules');
RequirePackage('owl2onto');
⟨/cltxml⟩
```

Now, we also need to register the namespace prefixes for LaTeXML to use.

```
⟨*cltxml⟩
RegisterNamespace('owl'=>"http://www.w3.org/2002/07/owl#");
RegisterNamespace('stex'=>"http://kwarc.info/ns/sTeX");
RegisterNamespace('ltx'=>"http://dlmf.nist.gov/LaTeXML");
RegisterNamespace('omdoc'=>"http://omdoc.org/ns");
⟨/cltxml⟩
```

Since we are dealing with a class, we need to set up the document type in the LaTeXML bindings.

```
⟨*cltxml⟩
RelaxNGSchema('owl2+ltxml',
        '#default'=>"http://www.w3.org/2002/07/owl#",
        'omdoc'=>"http://omdoc.org/ns",
        'ltx'=>"http://dlmf.nist.gov/LaTeXML",
        'stex'=>"http://kwarc.info/ns/sTeX");
```

```
⟨/cltxml⟩
```

The owl2onto class also supplies the top-level `document` environment, which we redefined so that we can provide KeyVal arguments.

EdNote(4) `document` The `document` environment is redefined to allow a baseURI[4]

```
⟨*cls⟩
\srefaddidkey{ontology}
\addmetakey{ontology}{baseURI}
\newcommand{\ontology}[1][]{\document\metasetkeys{ontology}{#1}%
\importmodule[owl2]{OWL2}%
\ifx\sref@id\@empty\begin{module}\else\begin{module}[id=\sref@id]\fi}
\newcommand{\endontology}{\end{module}\enddocument}
⟨/cls⟩
⟨*cltxml⟩
DefEnvironment('{ontology} OptionalKeyVals:omdoc',
       "<owl:Ontology "
      .    "?&KeyVal(#1,'id')(xml:id='&KeyVal(#1,'id')')() "
              .    "?&KeyVal(#1,'baseURI')(xml:base='&KeyVal(#1,'baseURI')/')() "
              .    "?&KeyVal(#1,'baseURI')(ontologyIRI='&KeyVal(#1,'baseURI')')() "
      .    "?#locator(stex:srcref='#locator')()>"
              .   "#body"
      ."</owl:Ontology>",
       beforeDigest=> sub { AssignValue(inPreamble=>0); },
       afterDigest=> sub { $_[0]->getGullet->flush; return; });#$
⟨/cltxml⟩
```

2.2 Classes and Properties

Before we provide the core functionality, we need to ensure that the `modules` and `presentation` packages are loaded. For LaTeXML we also initialize the package inclusions.

```
⟨*package⟩
\RequirePackage{amstext}
\RequirePackage{presentation}
⟨/package⟩
⟨*sltxml⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Global;
use LaTeXML::Package;
⟨/sltxml⟩
```

We first ste up a utility macro that allows us to export values

`\exportvalue`

[4]EdNote: @Deyan, need to remember the baseURI in a keyword, so that we can use it in the class and property environments

```
⟨*package⟩
\def\exportvalue#1#2{%
\expandafter\def\csname\declaration@cseq #2\expandafter\endcsname\expandafter{#1}
\expandafter\g@addto@macro\csname module@defs@\mod@id\expandafter\endcsname\expandafter%
{\expandafter\def\csname\declaration@cseq #2\expandafter\endcsname\expandafter{#1}}}
⟨/package⟩
```

\declaration

```
⟨*package⟩
\addmetakey{declaration}{id}
\addmetakey{declaration}{cseq}
\addmetakey{declaration}{type}
\addmetakey{declaration}{name}
\newcommand{\declaration}[2][]{\metasetkeys{declaration}{#1}%
\ifx\declaration@cseq\@empty\else
\expandafter\exportvalue{\ontology@baseURI\declaration@name}{@URI}
\expandafter\exportvalue{\expandafter\text\declaration@name}{}
\fi
\noindent\textbf{\declaration@type: \declaration@name}\par\noindent #2\par\noindent}
⟨/package⟩
⟨*sltxml⟩
DefMacro('\declaration OptionalKeyVals:declaration {}', sub {
 my ($self,@args)=@_;
 my $name = ToString(KeyVal($args[0],'name')) if $args[0];
 my $cseq = ToString(KeyVal($args[0],'cseq')) if $args[0];
 DefConstructor("\\".$cseq,"<XMTok name='$name'/>",
properties=>{mode=>'math',requiremath=>1})
               if ($cseq && $name);
 (Invocation(T_CS('\@declaration'),@args)->unlist);
});
DefConstructor('\@declaration OptionalKeyVals:declaration {}', sub {
  my ($document, $keyval, $arg, %properties) = @_;
  my $type = ToString(KeyVal($keyval,'type'));
  my $name = ToString(KeyVal($keyval,'name'));
  $document->openElement('owl:Declaration','stex:srcref'=>$properties{'locator'});
  $document->insertElement('owl:'.$type,undef,(IRI=>$name)) if $type;
  $document->closeElement('owl:Declaration');
  if (defined $arg) {
    $document->openElement('owl:AnnotationAssertion');
    $document->insertElement('owl:AnnotationProperty',undef,(IRI=>'http://www.w3.org/2000/01/rd
    $document->insertElement('owl:IRI',$name);
    $document->insertElement('owl:Literal',$arg);
    $document->closeElement('owl:AnnotationAssertion');
  }
});#$
⟨/sltxml⟩
```

\axiom

```
⟨*package⟩
\addmetakey{axiom}{id}
```

```
\addmetakey{axiom}{cseq}
\addmetakey{axiom}{type}
\addmetakey{axiom}{name}
\newcommand{\axiom}[3][]{\metasetkeys{axiom}{#1}%
\noindent\textbf{Axiom:} #2\par\noindent #3\par\noindent}
⟨/package⟩
⟨*sltxml⟩
DefConstructor('\axiom OptionalKeyVals:axiom {}{}',
             "<owl:Axiom>#2"
            ."?#3(<owl:Annotation>"
            .      "<owl:AnnotationProperty IRI='http://www.w3.org/2000/01/rdf-schema#comment'/>"
            .      "<owl:Literal>#3</owl:Literal>"
            .    "</owl:Annotation>)()"
            ."</owl:Axiom>");
⟨/sltxml⟩
```

2.3 Using Ontologies

useontology

```
⟨*package⟩
\def\useontology#1#2{\input}
⟨/package⟩
```

2.4 Finale

Finally, we need to terminate the file with a success mark for perl.

```
⟨cltxml | sltxml⟩1;
```