Diseño y Construcción de Ontologías

Acceso a las Entidades de las Ontologías

Dra. Maricela Bravo
mcbc@correo.azc.uam.mx

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

CONTENIDO

1. Interfaces de programación y Frameworks de Java para acceso a ontologías
2. Desarrollo de programas de acceso a las ontologías con OWL API
   a. Lectura y carga de modelos ontológicos
   b. Registro de individuos
   c. Listado de clases e individuos de la ontología
   d. Creación de relaciones semánticas entre individuos

Librerías de Java para el Manejo de Ontologías

- OWL API es un conjunto de librerías de clases en Java que ofrecen mecanismos para la manipulación y administración de ontologías OWL.
- Existen varias implementaciones de las librerías, entre éstas se encuentran las siguientes:
  - OWL API de Protege
  - OWL API de Jena
  - OWL API de Semantic Web
  - GATE Ontology Plugin

OWL API de Semantic Web

- La OWL API es una interface de Java y una implementación del lenguaje de ontologías OWL de la W3C, utilizado para representar ontologías en la Web Semántica.
- La API se enfoca en OWL DL y en la nueva versión de OWL 2.
- La API OWL también ofrece soporte para utilizar razonadores como FaCT++ y Pellet.

OWL API de Semantic Web

- La API OWL incluye los siguientes componentes:
  - Una API para OWL 2 y una implementación del modelo de referencia eficiente.
  - RDF/XML parser y writer
  - OWL/XML parser y writer
  - OWL Functional Syntax parser y writer
  - OBO (Open Bio Ontologies) Flat file format parser
  - Soporte para la integración con razonadores como Pellet y FaCT++

OWL API de Protege

- La API de Protégé-OWL API es una librería de Java de código abierto para OWL y RDF(S).
- Proporciona clases y métodos para cargar y salvar archivos OWL, para consultar y manipular modelos de datos en OWL, y para realizar razonamiento.
- Está diseñada para ser usada en dos contextos:
  - Para el desarrollo de componentes que son ejecutados dentro del editor de Protégé.
  - Para el desarrollo de aplicaciones de Java standalone.

OWL API de Protege

- Protege es una plataforma flexible y configurable para el desarrollo de aplicaciones y componentes.
- Protege tiene una arquitectura abierta que permite a los programadores integrar los plug-ins, que pueden aparecer como pestañas separadas y específicas de componentes de interfaz de usuario, o realizar cualquier otra tarea en el modelo actual.
- El editor de Protege-OWL proporciona muchas facilidades de edición y navegación de los modelos de OWL.

OWL API de Jena

- Jena es un marco de trabajo en Java para crear aplicaciones de Web Semántica.
- Jena ofrece una colección de herramientas y librerías de Java para ayudar a desarrollar aplicaciones, herramientas y servidores para la web semántica.

OWL API de Jena

- El marco de trabajo de Jena incluye:
  - una API para leer, procesar y escribir datos RDF en XML, N-triples y formato Turtle
  - una API de ontologías para el manejo de OWL y RDFS
  - un motor de inferencia basado en reglas para el razonamiento con RDF y OWL
  - almacenes para permitir que grandes cantidades de tripletas RDF sean eficientemente almacenadas en disco
  - un motor de consultas compatible con la última especificación de SPARQL

Tabla Comparativa de las APIs

Old Protégé-OWL API	RDF/OWL element	New Protégé-OWL API	Jena	Wonderweb API
OWLKnowledgeBase	(model)	OWLModel	OntModel	OWLOntology
OWLInstance	rdf:Resource	RDFResource	OntResource	OWLEntity
RDFSIot	rdf:Property	RDFProperty	OntProperty	OWLProperty (?)
OWLSlot	-	*OWLProperty*	OntProperty	OWLProperty
DatatypeSlot	owl:DatatypeProperty	OWLDatatypeProperty	DatatypeProperty	OWLDataProperty
ObjectSlot	owl:ObjectProperty	OWLObjectProperty	ObjectProperty	OWLObjectProperty
OWLCls	-	*RDFSClass*	OntClass	OWLDescription
RDFSCls	rdfs:Class	RDFSNamedClass	OntClass	?
NamedCls	owl:Class	OWLNamedClass	OntClass	OWLClass
AnonymousCls	owl:Class	*OWLAnonymousClass*	(OntClass.isAnon)	?
Restriction	owl:Restriction	*OWLRestriction*	Restriction	OWLRestriction
AllRestriction	owl:allValuesFrom	OWLAllValuesFrom	AllValuesFromRestriction	OWLxAllRestriction
SomeRestriction	owl:someValuesFrom	OWLSomeValuesFrom	SomeValuesFromRestriction	OWLxSomeRestriction
HasRestriction	owl:hasValue	OWLHasValue	HasValueRestriction	OWLxValueRestriction
CardiRestriction	owl:cardinality	OWLCardinality	CardinalityRestriction	OWLCardinalityRestriction
MaxCardinalityRestriction	owl:maxCardinality	OWLMaxCardinality	MaxCardinalityRestriction	OWLCardinalityRestriction
MinCardinalityRestriction	owl:minCardinality	OWLMinCardinality	MinCardinalityRestriction	OWLCardinalityRestriction
IntersectionCls	owl:intersectionOf	OWLIntersectionClass	IntersectionClass	OWLAnd
UnionCls	owl:unionOf	OWLUnionClass	UnionClass	OWLOr
ComplementCls	owl:complementOf	OWLComplementClass	ComplementClass	OWLNot
EnumerationCls	owl:oneOf	OWLEnumeratedClass	EnumeratedClass	OWLEnumeration
SimpleOWLInstance	(Individual of rdfs:Class)	RDFIndividual	Individual	OWLIndividual
SimpleOWLInstance	(Individual of owl:Class)	OWLIndividual	Individual	OWLIndividual
ListInstance	rdf:List	RDFList	RDFList	?
OntologyInstance	owl:Ontology	OWLOntology	Ontology	OWLOntology (?)

Tabla Comparativa de las APIs

Old Protégé-OWL API	RDF/OWL element	New Protégé-OWL API	Jena	Wonderweb API
AllDifferentInstance	owl:AllDifferent	OWLAllDifferent	AllDifferent	?
(ValueType SYMBOL)	owl:DataRange	OWLDataRange	DataRange	?
(primitive values)	rdf:Literal	RDFLiteral	Literal	?
ExternalResourceInstance	rdf:Resource (untyped)	RDFExternalResource	Resource	?
isAnnotationProperty()	owl:AnnotationProperty	isAnnotationProperty()	AnnotationProperty	OWLAnnotationProperty
isSymmetric()	owl:SymmetricProperty	isSymmetric()	SymmetricProperty	isSymmetric
isTransitive()	owl:TransitiveProperty	isTransitive()	TransitiveProperty	isTransitive()
isFunctional()	owl:FunctionalProperty	isFunctional()	FunctionalProperty	isOneToOne()
isInverseFunctional()	owl:InverseFunctionalProperty	isInverseFunctional()	InverseFunctionalProperty	isInverseFunctional()

Programas para el manejo de ontologías desde Java

1. Registro de individuos
2. Creación de relaciones dataType para los individuos
3. Creación de relaciones objectProperty entre individuos.

Registro de individuos

```
public void createClassIndividual(String ontoFile, String ontoIRI, String clase, String individual) throws
    OWLOntologyStorageException, OWLOntologyCreationException
  {
    try
    {
      OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
      IRI ontologyIRI = IRI.create(ontoIRI);
      IRI ontoIri = IRI.create(new File(ontoFile));
      OWLOntology ontology = manager.loadOntologyFromOntologyDocument(ontoIri);
      OWLDataFactory factory = manager.getOWLDataFactory();
      IRI namedClase = generateIRI(clase,ontologyIRI);
      IRI namedIndividual = generateIRI(individual,ontologyIRI);
      OWLClass clasePadre = factory.getOWLClass(namedClase);
      OWLNamedIndividual ind = factory.getOWLNamedIndividual(namedIndividual);
      OWLClassAssertionAxiom classAssertion = factory.getOWLClassAssertionAxiom(clasePadre, ind);
      manager.addAxiom(ontology, classAssertion);
      manager.saveOntology(ontology);
    }
    catch (OWLOntologyCreationException e)
    {       System.out.println("No se pudo cargar la ontología: " + e.getMessage());      }
    catch (OWLOntologyStorageException e)
    {        e.printStackTrace();          }
  }
```

Relaciones dataType

```
public void createDataPropertyAssertion(String ontoFile, String ontoIRI, String objName,
    String property, String value) throws OWLOntologyStorageException,
    OWLOntologyCreationException
  {
      OWLOntologyManager manager =  OWLManager.createOWLOntologyManager();
      IRI ontologyIRI = IRI.create(ontoIRI);
      IRI ontoIri = IRI.create(new File(ontoFile));
      OWLOntology ontology = manager.loadOntologyFromOntologyDocument(ontoIri);
      OWLDataFactory factory = manager.getOWLDataFactory();
      IRI dataPropertyName = generateIRI(property,ontologyIRI);
      OWLDataProperty owlDatatype =
        factory.getOWLDataProperty(dataPropertyName);
      IRI namedIndividual = generateIRI(objName,ontologyIRI);
      OWLNamedIndividual individual =
        factory.getOWLNamedIndividual(namedIndividual);
      OWLLiteral owlLiteral = factory.getOWLStringLiteral(value);
      OWLDataPropertyAssertionAxiom dataProperty =
        factory.getOWLDataPropertyAssertionAxiom(owlDatatype, individual, owlLiteral);
      manager.addAxiom( ontology, dataProperty);
      manager.saveOntology(ontology);
    }
```

Relaciones objectProperty

```
public void createObjectPropertyAssertion(String ontoFile, String ontoIRI, String relation, String
    obj1Name, String obj2Name) throws OWLOntologyStorageException,
    OWLOntologyCreationException
    {
     OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
     IRI ontologyIRI = IRI.create(ontoIRI);
     IRI ontoIri = IRI.create(new File(ontoFile));
     OWLOntology ontology = manager.loadOntologyFromOntologyDocument(ontoIri);
     OWLDataFactory factory = manager.getOWLDataFactory();
     IRI namedIndividual1 = generateIRI(obj1Name,ontologyIRI);
     IRI namedIndividual2 = generateIRI(obj2Name,ontologyIRI);
     IRI hasRelationIRI = generateIRI(relation,ontologyIRI);
     OWLNamedIndividual obj1 = factory.getOWLNamedIndividual(namedIndividual1);
     System.out.println(obj1.toString());
     OWLNamedIndividual obj2 = factory.getOWLNamedIndividual(namedIndividual2);
     System.out.println(obj2.toString());
     OWLObjectProperty objProperty = factory.getOWLObjectProperty(hasRelationIRI);
     System.out.println(objProperty.toString());


     OWLObjectPropertyAssertionAxiom propertyAssertion =
    factory.getOWLObjectPropertyAssertionAxiom(objProperty, obj1, obj2);
     manager.addAxiom(ontology, propertyAssertion);


     manager.saveOntology(ontology);
```