Jena

CSC480: Semantic Web Technologies

Dr. Lisa Frye
frye@kutztown.edu
Kutztown University

Ontology Frameworks

Framework	Language	Semantic Level
Jena	Java	RDF to OWL2
Sesame	Java & RESTful web service	RDF
OWL API	Java	OWL2

Jena

- Programming framework
- Includes SPARQL implementation
- Not scalable

Jena's Major Java Classes

- Resource
- Statement
- Graph
- Model
- Query
- ResultSet
- Reasoner

Define Ontology Files

```
public static final String TRAFFICONT =
                    KButility.URL_PREFIX + "traffic";
public static final String TRAFFICONT_URL =
                    TRAFFICONT + ".owl";
public static final String TRAFFICONT_PREFIX =
                    TRAFFICONT_URL + "#";
```

Establish the KB (model)

- createModel()
- createOntologyModel()

```
public static OntModel traffModel;
traffModel = ModelFactory.createOntologyModel(
        OntModelSpec.OWL_MEM_RULE_INF, null);
```

Models

- OWL_MEM
- OWL_MEM_RDFS_INF
- OWL_MEM_RULE_INF
- OWL_MEM_TRANS_IF

- OWL_DL_
- OWL_LITE_

Connect To Reasoner

```
private static Reasoner owlReasoner;
owlReasoner =
        ReasonerRegistry.getOWLReasoner();
```

Populate Model

- File
- URL
- Programmatically

```
traffModel.read(
        KBconnect.TRAFFICONT_PREFIX);
```

Useful Java Methods

- *getResource* - retrieve a resource from the model or create if it doesn't exist
- *createResource* - retrieve a resource from the model or create if it doesn't exist
- *getOntClass* – return a resource for the specified class
- *createStatement* – create a statement instance in KB

Useful Java Methods (2)

- *createIndividual* – create new individual in KB
- *listStatements* – lists all statements in model
- *listSubjects* – returns an iterator over all resources that are the subject of a statement
- *listSubjectsWithProperty* – returns an iterator over all resources that have a specific value for a specific property

Useful Java Methods (3)

- *listProperties()* – returns an iterator to list all properties
- *getProperty* – access a property of a resource
- *addProperty* – add a property value to KB

Useful Java Methods (4)

- *getDatatypeProperty* – get a datatype property
- *getObjectProperty* – get an object property
- *createLiteral* – create an RDF literal
- *createTypedLiteral* – create a RDF typed literal
- *createLiteralStatement* – create a new statement with a value of a RDF typed literal

Code Sample

- Get the URI for a class

```
String classSt = TRAFFICONT_PREFIX + "Packet";
OntClass packet =
        KBconnect.traffModel.getOntClass(classSt);
```

Code Sample (2)

- Create an individual

```
Individual individual;
individual = KBconnect.traffModel.createIndividual(
                                TRAFFICONT_PREFIX +
                                 "packet" + i, tcppacket);
```

Code Sample (3)

```
propStr = TRAFFICONT_PREFIX + "packetID";
intValue = i + 1;
prop = Bconnect.traffModel.getDatatypeProperty(propStr);
if ((prop != null) && (intValue != null)) {
   propLiteral =
                    KBconnect.traffModel.createTypedLiteral(
                                        intValue, XSDDatatype.XSDint);
   statement =
                    KBconnect.traffModel.createLiteralStatement(
                                         individual, prop, propLiteral);
   KBconnect.traffModel.add(statement);
}  // end if
```

Code Sample (4)

```
// IPPacket class fields
propStr = TRAFFICONT_PREFIX + "hasSrcIP";
propValue = TRAFFICONT_PREFIX +
                        packetList.get(i).getSrcIP();
// Find the resource for the IP address
res = KBconnect.traffModel.getResource(propValue);
```

Sample cont.

```
if (res == null)
        System.out.println("No IP address resource found!");
else {
        oprop = KBconnect.traffModel.getObjectProperty(propStr);
        if (oprop != null) {
                statement =
                        KBconnect.traffModel.createStatement(
                                individual, oprop, res);
                KBconnect.traffModel.add(statement);
        }  // end if
}  // end else
```

Query KB using SPARQL

- Create query
- Execute
  - *executeQuery()*

Modify KB using SPARQL Update

- Create graph store
- Set default graph store
- Create query
- Execute query

SPARQL Update Example

```
GraphStore graphStore = GraphStoreFactory.create();
graphStore.setDefaultGraph(KBconnect.traffModel.getGraph());

//Create the query
UpdateRequest updateRequest = UpdateFactory.create(queryStr);
UpdateProcessor uExecFactory =
            UpdateExecutionFactory.create(
                                    updateRequest, graphStore);
//Execute the query
uExecFactory.execute();
```