Apache Jena API

• Developers: 2000-2009 HP Labs

2009-now  Apache Software Foundation (vote: 10+7-)

Vote: http://mail-archives.apache.org/mod_mbox/incubator-general/201011.mbox/%3C4CEC31E4.9080401@apache.org%3E

• Open source Semantic Web framework for Java.
• Read and write to RDF graphs.
• Graphs in Jena are represented as an abstract “model”.
• Model can be queried through SPARQL.
RDF and SPARQL

• “The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model”

• “SPARQL is an RDF query language -- that is, a semantic query language for databases.”

-------Wikipedia
Framework Architecture
Examples: creating a simple model

// some definitions
static String personURI = "http://somewhere/JohnSmith";
static String fullName = "John Smith";

// create an empty Model
Model model = ModelFactory.createDefaultModel();

// create the resource
Resource johnSmith = model.createResource(personURI);

// add the property
johnSmith.addProperty(VCARD.FN, fullName);

Resource johnSmith =
    model.createResource(personURI) 
    .addProperty(VCARD.FN, fullName);
Examples: creating a model

```java
// some definitions
String personURI = "http://somewhere/JohnSmith";
String givenName = "John";
String familyName = "Smith";
String fullName = givenName + " " + familyName;

// create an empty Model
Model model = ModelFactory.createDefaultModel();

// create the resource
// and add the properties cascading style
Resource johnSmith = model.createResource(personURI)
    .addProperty(VCARD.FN, fullName)
    .addProperty(VCARD.N, model.createResource()
        .addProperty(VCARD.Given, givenName)
        .addProperty(VCARD.Family, familyName));
```
Examples: reading & writing

```java
// create an empty model
Model model = ModelFactory.createDefaultModel();

// use the FileManager to find the input file
InputStream in = FileManager.get().open( inputFileName );
if (in == null) {
    throw new IllegalArgumentException(
        "File: " + inputFileName + " not found" );
}

// read the RDF/XML file
model.read(in, null);

// write it to standard out
model.write(System.out);
```
Examples: reading & writing

```
<rdf:RDF
    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
    xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'>
    <rdf:Description rdf:about='http://somewhere/JohnSmith'>
        <vcard:FN>John Smith</vcard:FN>
        <vcard:N rdf:nodeID="A0"/>
    </rdf:Description>
    <rdf:Description rdf:nodeID="A0">
        <vcard:Given>John</vcard:Given>
        <vcard:Family>Smith</vcard:Family>
    </rdf:Description>
</rdf:RDF>
```
Examples: operations on models

```java
// read the RDF/XML files
model1.read(new InputStreamReader(in1), "");
model2.read(new InputStreamReader(in2), "");

// merge the Models
Model model = model1.union(model2);

// print the Model as RDF/XML
model.write(system.out, "RDF/XML-ABBREV");
```
Examples: operations on models

// read the RDF/XML files
model1.read(new InputStreamReader(in1), "");
model2.read(new InputStreamReader(in2), "");

// merge the Models
Model model = model1.union(model2);

// print the Model as RDF/XML
model.write(System.out, "RDF/XML-ABBREV");
Statements (arc in RDF)

• Also called as triple.

• A statement has three parts:
  • Subject: the resource from which the arc leaves.
  • Predicate: the property that labels the arc.
  • Object: the resource or literal pointed to by the arc.

```java
// list the statements in the Model
StmtIterator iter = model.listStatements();

// print out the predicate, subject and object of each statement
while (iter.hasNext()) {
    Statement stmt = iter.nextStatement(); // get next statement
    Resource subject = stmt.getSubject(); // get the subject
    Property predicate = stmt.getPredicate(); // get the predicate
    RDFNode object = stmt.getObject(); // get the object
}
Examples: querying a model

```java
// select all the resources with a VCARD.FN property
ResIterator iter = model.listSubjectsWithProperty(VCARD.FN);
if (iter.hasNext()) {
    System.out.println("The database contains vcards for:");
    while (iter.hasNext()) {
        System.out.println(
            "   " + iter.nextResource()
            .getProperty(VCARD.FN)
            .getString());
    }
} else {
    System.out.println("No vcards were found in the database");
}
```

The database contains vcards for:
Sarah Jones
John Smith
Matt Jones
Becky Smith
Examples: querying a model

```java
// select all the resources with a VCARD.FN property
// whose value ends with "Smith"
StmtIterator iter = model.listStatements(
    new SimpleSelector(null, VCARD.FN, (RDFNode) null) {
        public boolean selects(Statement s)
        { return s.getString().endsWith("Smith"); }
    });
```

The database contains vcards for:
- John Smith
- Becky Smith
ARQ: A SPARQL processor for Jena that allows you to use SPARQL RDF query language.

```java
public ResultSet executeQuery(String queryString) throws Exception {
    Query query = QueryFactory.create(queryString);

    QueryEngineHTTP qexec = QueryExecutionFactory.createServiceRequest(this.service, query);
    qexec.addParam("apikey", this.apikey);
    ResultSet results = qexec.execSelect();
    return results;
}

String query = "PREFIX omv: <http://omv.ontoware.org/2005/05/ontology#> " +
    "SELECT ?ont ?name ?acr " +
    "WHERE { ?ont a omv:Ontology; " +
    "omv:acronym ?acr; " +
    "omv:name ?name . " +
    "}"
;

JenaARQTTest test = new JenaARQTTest(sparqlService, apikey);
ResultSet results = test.executeQuery(query);
```
Containers

• Collection of things are called containers.
• The members of a container can be either literals or resources.
• There are 3 types of containers:
  • BAG: unordered collection.
  • SEQ: ordered collection.
  • ALT: unordered collection intended to represent alternatives.
Camera Ontology:
protege.cim3.net/file/pub/ontologies/camera/camera.owl
```java
public class BFSInRDFWithJena {

    public static List<List<Resource>> BFS( final Model model, final Queue<List<Resource>> queue, final int depth ) {
        final List<List<Resource>> results = new ArrayList<>();
        while ( !queue.isEmpty() ) {
            final List<Resource> path = queue.poll();
            results.add( path );
            if ( path.size() < depth ) {
                final Resource last = path.get( path.size() - 1 );
                final StmtIterator stmt = model.listStatements( null, RDFS.subClassOf, last );
                while ( stmt.hasNext() ) {
                    final List<Resource> extPath = new ArrayList<>( path );
                    extPath.add( stmt.next().getSubject().asResource() );
                    queue.offer( extPath );
                }
            } else {
                // Get the paths, and display them
                final List<List<Resource>> paths = BFS( model, queue, 4 );
                for ( List<Resource> path : paths ) {
                    System.out.println( path );
                }
            }
        }
        return results;
    }

    public static void main( final String[] args ) throws IOException {
        final Model model = ModelFactory.createDefaultModel();
        try ( final InputStream in = BFSInRDFWithJena.class.getClassLoader().getResourceAsStream( "camera.owl" ) ) {
            model.read( in, null );
        }

        // setup the initial queue
        final Queue<List<Resource>> queue = new LinkedList<>();
        final List<Resource> thingPath = new ArrayList<>();
        thingPath.add( OWL.Thing );
        queue.offer( thingPath );

        // Get the paths, and display them
        final List<List<Resource>> paths = BFS( model, queue, 4 );
        for ( List<Resource> path : paths ) {
            System.out.println( path );
        }
    }
}
```
BFS in Jena API
Concurrency

```java
Model model = ...;
model.enterCriticalSection(Lock.READ);  // or Lock.WRITE
try {
    ... perform actions on the model ...
    ... obey contract - no update operations if a read lock
} finally {
    model.leaveCriticalSection();
}
```
Apache Jena Elephas

• A set of libraries that enable you to start writing Apache Hadoop based applications which work with RDF data.
• Under active development for about a year.
• Still in Beta stage.
• https://jena.apache.org/documentation/hadoop/index.html