The ICS-FORTH SWKM Model: A Main Memory RDF/S Management System

(see also "The ICS-FORTH SWKM (Semantic Web Knowledge Middleware)"
http://athena.ics.forth.gr:9090/SWKM/)

Overview

SWKM Model is a Main Memory RDF/S Management System. Unlike most known RDF parsers, it provides both an Object-based Model and a Triple-based Model for Main Memory RDF manipulation. The Model is not intended to be either a persistence storage model or a model that can be updated programmatically in main memory. For this purpose one should use the SWKM Services .

An important feature of the SWKM Model is it capability of handling both namespaces and graphspaces. For every namespace of the model, the triples in it are saved and as a result, they can easily be retrieved. A namespace is used programmatically as a container where all the relevant information to an actual RDF namespace is kept. Namespaces hold information (references) to the namespaces they depend on. A graphspace is defined as an arbitrary but defined collection of triples. Again, graphspaces hold information (references) to the namespaces or graphspaces they depend on. This provides the ability to save information either at the model, the namespace or the graphspace level, depending on its nature.

In SWKM Model a node is either a literal or a resource. A Hash Map that holds all the literals existing in the model is used. If the node is a resource it can be one of class, property, class instance or property instance. Depending on its type the node is saved in separate Hash Maps at the model level (class instances and property instances) or at the namespace level (classes and properties), with each Hash Map holding one type of resources and using as key its (unique) URI and as value the corresponding object.

After the nodes are saved, the triple is created as an object holding references to the nodes involved and to the namespace and/or graphspace that it might belong (if the same triple is found belonging to another graphspace then only a reference is added) and is saved either in the model or in the namespace it belongs. Triples are again typed against the RDF/S basic properties:

- Subclass
- Subproperty
- Domain
- Range
- Comment
- Label
- Seals

- Isdefinedby
- Member
- Type

If the predicate belongs in none of these categories, it is considered to be a property instance. All of these are typed as subclasses of the class Triple and their typing information can be exploited programmatically. For each of the mentioned types, there are six hash maps that contain the corresponding triples as values, having as keys the subject, the object or a multi key of all their possible combinations respectively. This has as a result larger memory usage but faster retrieval times for the triple view. Moreover separating the triples and providing object representations for each predicate type facilitates work on diff and evolution (and other algorithms that use the triple based representation) of RDF models, since it allows for easier identification and manipulation of the triples.

On the other hand, SWKM Model provides a complete object view on the RDF model at hand. This means that nodes can be retrieved fast (since they are saved in Hash Maps) and moreover all information available in the model can be retrieved by the API methods provided at the object level.

About API Access Methods

The SWKM Model API is a rich API. The developer has the possibility to choose between using the triple based API and the object based API to retrieve the required information. In both cases a wealth of access methods is offered.

In the triple based view a retrieval method for all triples of the model, as well as one for all triples with some subject, predicate and/or object are offered. To retrieve all triples of a model, all values of the hash maps used are returned. For this operation, all the maps hashed on the subjects are used. When searching for triples by subject, predicate and/or object, the respective hash maps are used. For example, when the predicate is given, the two hash maps of the corresponding property or property instance are used: the one hashed on the subjects if only the subject and the predicate are given and the one hashed on object if the predicate and the object are given. Finally, if the predicate is not given, the desired values of the hash maps of all the predicates are added (e.g. if only the subject is given as a parameter, the hash maps, hashed on the subjects will be used) and an iterator on them will be returned.

A rich set of variations of these methods are also provided. For instance, a user can choose not to include in the results of the "find all triples" method the triples contained in RDF/S. On another example one can retrieve triples searching for them based on their (string) URI and not the object representation of the node. In between the triple and the object based view methods one can retrieve all triples associated with a specific namespace or graphspace by giving its unique URI. This is extremely useful for applications manipulating namespaces or graphspaces as one object.

In the object based view one can retrieve all the necessary information starting at any of the objects without having to retrieve and process any triples in between. This provides, for

example, easy access to subsumption relationship information like subClassOf, superClassOf, subPropertyOf and superPropertyOf (both direct and all the descendants/ancestors). This methods work recursively by finding the direct descendants (or ancestors) and then their direct descendants or ancestors. The same way access information on in which triple this class or class instance serves as a subject (domain) or an object (range) can be provided.

The object based view offers unique capabilities to the programmer using the API that allow objectification of the RDF information in a consistent way providing objects that represent a full RDF Model. Since this can be combined when needed with information retrieved through the triple based access methods, the programmer has a wealth of methods and ways to manipulate the same RDF information. This combined capability of representing and manipulating RDF information in main memory both as triples and as full blown objects is to the best of our knowledge unique in SWKM.

SWKM Model API methods Description

A SWKM Model can be seen both as a collection of triples or objects and be manipulated accordingly. The model provides functions for both these views. For the Object based view several operations are provided for the Model, Namespace, Property and Class Instance level. The Triple Based View supports operations on a Model, Namespace, Graphspace, Class, Property and Class Instance level. For more information regarding the methods provided by the API, see the relevant API Reference & Javadocs (in folder docs).

Downloads & Installation

How to Use

System Requirements	
Operating System	Linux Tested on the following platforms: • Solaris 8 • RedHat Linux 7.3 - 9.0 • Fedora Core 1 - 5 • Debian Linux 3.x • OpenSUSE Linux 10.x • Ubuntu Linux 6.6 Windows: • Windows 2000/XP
JDK	1.5 or higher

To use the client simply put all the required libraries included in the installation bundle (every file in the lib folder including the swkm-model.jar in the dist folder) into the java class path and compile the example given in our Tutorials Section. The required jars contained in the lib folder are the following:

- commons-collections-3.2.jar
- java_cup.jar

- JFlex.jar log4j-1.2.14.jar
- ng4j.jar
- postgresql-8.1-407.jdbc3.jar
- relaxngDatatype.jar
- TaskMonitor.jar
- trig-v1.0.jar
- vrp3.0.jar
- swkm-services-api.jar

Downloads

Download Name	Release Date	File Type
The SWKM MODEL 3.0.0 bundle (includes source code,binary,required libraries and javadocs)	4/12/2012	zip file