FindRelatedNodesForTermAndAssoc Java

Java Code

```
/*
 * Copyright: (c) 2004-2009 Mayo Foundation for Medical Education and
 * Research (MFMER). All rights reserved. MAYO, MAYO CLINIC, and the
 * triple-shield Mayo logo are trademarks and service marks of MFMER.
 *
 * Except as contained in the copyright notice above, or as used to identify
 * MFMER as the author of this software, the trade names, trademarks, service
 * marks, or product names of the copyright holder shall not be used in
 * advertising, promotion or otherwise in connection with this software without
 * prior written authorization of the copyright holder.
 *
 * Licensed under the Eclipse Public License, Version 1.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *                 http://www.eclipse.org/legal/epl-v10.html
 *
 */
package org.LexGrid.LexBIG.example;

import org.LexGrid.LexBIG.DataModel.Collections.LocalNameList;
import org.LexGrid.LexBIG.DataModel.Collections.ResolvedConceptReferenceList;
import org.LexGrid.LexBIG.DataModel.Core.AssociatedConcept;
import org.LexGrid.LexBIG.DataModel.Core.Association;
import org.LexGrid.LexBIG.DataModel.Core.CodingSchemeSummary;
import org.LexGrid.LexBIG.DataModel.Core.CodingSchemeVersionOrTag;
import org.LexGrid.LexBIG.DataModel.Core.ResolvedConceptReference;
import org.LexGrid.LexBIG.Exceptions.LBException;
import org.LexGrid.LexBIG.Impl.LexBIGServiceImpl;
import org.LexGrid.LexBIG.LexBIGService.LexBIGService;
import org.LexGrid.LexBIG.LexBIGService.CodedNodeSet.PropertyType;
import org.LexGrid.LexBIG.LexBIGService.CodedNodeSet.SearchDesignationOption;
import org.LexGrid.LexBIG.Utility.ConvenienceMethods;
import org.LexGrid.LexBIG.Utility.LBConstants.SortableProperties;
import org.LexGrid.commonTypes.EntityDescription;

/**
 * Example showing how to find all endpoints of a named association for which
 * the given term matches as source or target.
 *
 * Note: the match algorithm applied to the term is the standard lucene query
 * syntax.
 */
public class FindRelatedNodesForTermAndAssoc {

    public FindRelatedNodesForTermAndAssoc() {
        super();
    }

    /**
     * Entry point for processing.
     *
     * @param args
     */
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out
                    .println("Example: FindRelatedNodesForTermAndAssoc \"lung\" \"
Anatomic_Structure_Is_Physical_Part_Of\"");
            return;
        }
        ;

        try {
```

```
            String term = args[0];
            String assoc = args[1];
            new FindRelatedNodesForTermAndAssoc().run(term, assoc);
        } catch (Exception e) {
            Util.displayAndLogError("REQUEST FAILED !!!", e);
        }
    }

    public void run(String term, String assoc) throws LBException {
        CodingSchemeSummary css = Util.promptForCodeSystem();
        if (css != null) {
            LexBIGService lbSvc = LexBIGServiceImpl.defaultInstance();

            String scheme = css.getCodingSchemeURI();
            CodingSchemeVersionOrTag csvt = new CodingSchemeVersionOrTag();
            csvt.setVersion(css.getRepresentsVersion());

            printSources(term, assoc, lbSvc, scheme, csvt);
            printTargets(term, assoc, lbSvc, scheme, csvt);
        }
    }

    /**
     * Display association sources for which the given term participates as
     * target.
     *
     * @param term
     * @param assoc
     * @param lbSvc
     * @param scheme
     * @param csvt
     * @throws LBException
     */
    protected void printSources(String term, String assoc, LexBIGService lbSvc, String scheme,
            CodingSchemeVersionOrTag csvt) throws LBException {
        // Find all nodes that the term matches.
        ResolvedConceptReferenceList nodeList = lbSvc.getNodeSet(scheme, csvt, null).
restrictToMatchingDesignations(
                term, SearchDesignationOption.PREFERRED_ONLY, "LuceneQuery", null).resolveToList(
                ConvenienceMethods.createSortOptionList(new String[] { SortableProperties.code.name() }), null,
                new PropertyType[] { PropertyType.PRESENTATION }, 1024);

        // For each node, find and print related sources ...
        int nodeCount = nodeList.getResolvedConceptReferenceCount();
        for (int i = 0; i < nodeCount; i++) {
            ResolvedConceptReference node = (ResolvedConceptReference) nodeList.getResolvedConceptReference(i);

            // Get a graph of relationships for which the node
            // participate as a target.
            ResolvedConceptReferenceList matches = lbSvc.getNodeGraph(scheme, csvt, null).restrictToAssociations
(
                    ConvenienceMethods.createNameAndValueList(assoc), null).resolveAsList(node, false, true, 1,
1,
                    new LocalNameList(), null, null, 1024);

            int matchCount = matches.getResolvedConceptReferenceCount();
            if (matchCount > 0) {
                Util.displayMessage('\n' + node.getConceptCode() + '/' + node.getEntityDescription().
getContent());
                for (int j = 0; j < matchCount; j++) {
                    ResolvedConceptReference match = (ResolvedConceptReference) matches.
getResolvedConceptReference(j);
                    Association a = match.getTargetOf().getAssociation(0);
                    AssociatedConcept[] acl = a.getAssociatedConcepts().getAssociatedConcept();
                    String aName = a.getDirectionalName();
                    for (int k = 0; k < acl.length; k++) {
                        AssociatedConcept ac = acl[k];
                        EntityDescription ed = ac.getEntityDescription();
                        Util.displayMessage('\t' + (aName == null ? "[R]" + a.getAssociationName() : aName) +
": "
                                + ac.getConceptCode() + '/' + (ed == null ? "**No Description**" : ed.
```

```
getContent()));
                    }
                }
            }
        }
    }

    /**
     * Display association targets for which the given term participates as
     * source.
     *
     * @param term
     * @param assoc
     * @param lbSvc
     * @param scheme
     * @param csvt
     * @throws LBException
     */
    protected void printTargets(String term, String assoc, LexBIGService lbSvc, String scheme,
            CodingSchemeVersionOrTag csvt) throws LBException {
        // Find all nodes that the term matches.
        ResolvedConceptReferenceList nodeList = lbSvc.getNodeSet(scheme, csvt, null).
restrictToMatchingDesignations(
                term, SearchDesignationOption.PREFERRED_ONLY, "LuceneQuery", null).resolveToList(
                ConvenienceMethods.createSortOptionList(new String[] { SortableProperties.code.name() }), null,
                new PropertyType[] { PropertyType.PRESENTATION }, 1024);

        // For each node, find and print related targets ...
        int nodeCount = nodeList.getResolvedConceptReferenceCount();
        for (int i = 0; i < nodeCount; i++) {
            ResolvedConceptReference node = (ResolvedConceptReference) nodeList.getResolvedConceptReference(i);

            // Get a graph of relationships for which the node
            // participate as a source.
            ResolvedConceptReferenceList matches = lbSvc.getNodeGraph(scheme, csvt, null).restrictToAssociations
(
                    ConvenienceMethods.createNameAndValueList(assoc), null).resolveAsList(node, true, false, 1,
1,
                    new LocalNameList(), null, null, 1024);

            int matchCount = matches.getResolvedConceptReferenceCount();
            if (matchCount > 0) {
                Util.displayMessage('\n' + node.getConceptCode() + '/' + node.getEntityDescription().
getContent());
                for (int j = 0; j < matchCount; j++) {
                    ResolvedConceptReference match = (ResolvedConceptReference) matches.
getResolvedConceptReference(j);
                    Association a = match.getSourceOf().getAssociation(0);
                    AssociatedConcept[] acl = a.getAssociatedConcepts().getAssociatedConcept();
                    String aName = a.getDirectionalName();
                    for (int k = 0; k < acl.length; k++) {
                        AssociatedConcept ac = acl[k];
                        EntityDescription ed = ac.getEntityDescription();
                        Util.displayMessage('\t' + (aName == null ? a.getAssociationName() : aName) + ": "
                                + ac.getConceptCode() + '/' + (ed == null ? "**No Description**" : ed.
getContent()));
                    }
                }
            }
        }
    }
}
```