Skosmos

- Using Skosmos served vocabularies in VIVO
  - Creating a new Java class for a new vocabulary source
  - Adjusting the file ConceptSearchService.java
  - Defining the new vocabulary source in the data directory

Links and further information

- Project description and contact information: de/en
- Skosmos server: https://labs.tib.eu/skosmos/en/
- VIVO project: https://github.com/vivo-project/VIVO
- Integrating data from distributed sources via lookup services in EuropeanaTech Insight by Tatiana Walther
- and Christian Hauschke

This is an experimental service. It is not for productive use.

Using Skosmos served vocabularies in VIVO

New vocabularies, located on Skosmos server can be integrated in VIVO by means of a REST API or a SPARQL Endpoint of the Fuseki Server which holds the data in the background of Skosmos. The following documentation describes the implementation a new vocabulary service using the REST API of the Skosmos.

Creating a new Java class for a new vocabulary source

Within the following directory a new file with the Java class for the new vocabulary service has to be created **VIVO/api/src/main/java/edu/cornell/mannlib/semservices/service/impl/**

In the new file we declare the Java class for the new vocabulary source and we configure the connection between VIVO and the source:

FaechersystematikService.java

```
/* $This file is distributed under the terms of the license in LICENSE$ */
package edu.cornell.mannlib.semservices.service.impl;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringWriter;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.concurrent.TimeUnit;
import javax.xml.parsers.ParserConfigurationException;
import org.apache.commons.lang3.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.jena.query.Query;
import org.apache.jena.query.QueryExecution;
import org.apache.jena.query.QueryExecutionFactory;
import org.apache.jena.query.QueryFactory;
import org.apache.jena.query.QuerySolution;
import org.apache.jena.query.ResultSet;
import org.apache.jena.rdf.model.Literal;
import org.apache.jena.rdf.model.RDFNode;
import org.apache.jena.rdf.model.Resource;
import org.w3c.dom.Attr;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
```

```
import org.xml.sax.SAXException;
import com.fasterxml.jackson.databind.node.ArrayNode;
import com.fasterxml.jackson.databind.node.ObjectNode;
import edu.cornell.mannlib.semservices.bo.Concept;
import edu.cornell.mannlib.semservices.service.ExternalConceptService;
import edu.cornell.mannlib.semservices.util.SKOSUtils;
import edu.cornell.mannlib.semservices.util.XMLUtils;
import edu.cornell.mannlib.vitro.webapp.utils.json.JacksonUtils;
import edu.cornell.mannlib.vitro.webapp.web.URLEncoder;
//The definition of the new jawa class for the Faechersystematik as an external service
public class FaechersystematikService implements ExternalConceptService {
protected final Log logger = LogFactory.getLog(getClass());
private final String schemeUri = "https://onto.tib.eu/destf/cs/";
private final String ontologyName = "faechersystematik";
private final String format = "SKOS";
private final String lang = "de";
private final String searchMode = "starts with";//Used to be Exact Match, or exact word or starts with
protected final String dbpedia_endpoint = " http://dbpedia.org/sparql";
// URL to get all the information for a concept
// Exchange SKOSMOS_PATH with your local skosmos url.
protected final String conceptSkosMosBase = "SKOSMOS_PATH/skosmos/rest/v1/";
protected final String conceptsSkosMosSearch = conceptSkosMosBase + "search?";
protected final String conceptSkosMosURL = conceptSkosMosBase + "/faechersystematik/data?";


@Override
public List<Concept> getConcepts(String term) throws Exception {
List<Concept> conceptList = new ArrayList<Concept>();

//For the RDF webservices mechanism, utilize the following
/*
String result = getTermExpansion(this.ontologyName, term,
this.searchMode, this.format, this.lang);

// return empty conceptList if conceptUri is empty
if (StringUtils.isEmpty(result)) {
return conceptList;
}
// Get the list of the concept URIs in the RDF
List<String> conceptUris = getConceptURIsListFromRDF(result);
*/

//For the SKOSMos search mechanism, utilize this instead
String result = getSKOSMosSearchResults(term, this.lang);
List<String> conceptUris = getConceptURIsListFromSkosMosResult(result);

if (conceptUris.size() == 0)
return conceptList;
int conceptCounter = 0;

HashSet<String> encounteredURI = new HashSet<String>();

// Loop through each of these URIs and load using the SKOSManager
for (String conceptUri : conceptUris) {
conceptCounter++;
if (StringUtils.isEmpty(conceptUri)) {
// If the conceptURI is empty, keep going
continue;
}
if(encounteredURI.contains(conceptUri)) {
//If we have already encountered this concept URI, do not redisplay or reprocess
continue;
}
encounteredURI.add(conceptUri);
// Test and see if the URI is valid
URI uri = null;
try {
uri = new URI(conceptUri);
} catch (URISyntaxException e) {
logger.error("Error occurred with creating the URI ", e);
continue;
```

```
}
// Returns concept information in the format specified, which is
// currently XML
// Utilizing Agrovoc's getConceptInfo returns alternate and
// preferred labels but
// none of the exact match or close match descriptions
String bestMatch = "false";
//Assume the first result is considered the 'best match'
//Although that is not something we are actually retrieving from the service itself explicitly
if(conceptCounter == 1) {
bestMatch = "true";
}

Concept c = this.createConcept(bestMatch, conceptUri);
c.setDefinition("No definition found");
c.setLabel(getConceptPrefLabelFromSkosMosResult(result));
// c.setNotation(getConceptNotationFromSkosMosResult(result));
// c.setLabel(prefLabel);
conceptList.add(c);
// if (c != null) {
// // Get definition from dbpedia references stored in the close
// // Match list
// List<String> closeMatches = c.getCloseMatchURIList();
// for (String closeMatch : closeMatches) {
//
// if (closeMatch.startsWith("http://dbpedia.org")) {
// try {
// String description = getDbpediaDescription(closeMatch);
// c.setDefinition(description);
// } catch (Exception ex) {
// logger.error("An error occurred in the process of retrieving dbpedia description", ex);
// }
// }
// }
// conceptList.add(c);
// }
}

System.out.println(conceptList);
return conceptList;
}
public List<Concept> processResults(String term) throws Exception {
return getConcepts(term);
}
public Concept createConcept(String bestMatch, String skosConceptURI) {
Concept concept = new Concept();
concept.setUri(skosConceptURI);
concept.setConceptId(stripConceptId(skosConceptURI));
concept.setBestMatch(bestMatch);
concept.setDefinedBy(schemeUri);
concept.setSchemeURI(this.schemeUri);
concept.setType("");

concept.setUri(skosConceptURI);
concept.setConceptId(stripConceptId(skosConceptURI));
concept.setBestMatch(bestMatch);
concept.setDefinedBy(schemeUri);
concept.setSchemeURI(this.schemeUri);
concept.setType("");
concept.setLabel("test");
String encodedURI = URLEncoder.encode(skosConceptURI);
String encodedFormat = URLEncoder.encode("application/rdf+xml");
String url = conceptSkosMosURL + "uri=" + encodedURI + "&format="
+ encodedFormat;
// Utilize the XML directly instead of the SKOS API
try {
concept = SKOSUtils
.createConceptUsingXMLFromURL(concept, url, "en", false);
} catch (Exception ex) {
logger.debug("Error occurred for creating concept "
+ skosConceptURI, ex);
```

```
return null;
}

return concept;
}
public List<Concept> getConceptsByURIWithSparql(String uri)
throws Exception {
// deprecating this method...just return an empty list
List<Concept> conceptList = new ArrayList<Concept>();
return conceptList;
}
protected String getAgrovocTermCode(String rdf) throws Exception {
String termcode = new String();
try {
Document doc = XMLUtils.parse(rdf);
NodeList nodes = doc.getElementsByTagName("hasCodeAgrovoc");
if (nodes.item(0) != null) {
Node node = nodes.item(0);
termcode = node.getTextContent();
}
} catch (SAXException e) {
// e.printStackTrace();
throw e;
} catch (ParserConfigurationException e) {
// e.printStackTrace();
throw e;
} catch (IOException e) {
// e.printStackTrace();
throw e;
}
return termcode;
}
protected String getConceptURIFromRDF(String rdf) {
String conceptUri = new String();
try {
Document doc = XMLUtils.parse(rdf);
NodeList nodes = doc.getElementsByTagName("skos:Concept");
Node node = nodes.item(0);
NamedNodeMap attrs = node.getAttributes();
Attr idAttr = (Attr) attrs.getNamedItem("rdf:about");
conceptUri = idAttr.getTextContent();
} catch (IOException e) {
e.printStackTrace();
System.err.println("rdf: " + rdf);
} catch (SAXException e) {
e.printStackTrace();
System.err.println("rdf: " + rdf);
} catch (ParserConfigurationException e) {
e.printStackTrace();
System.err.println("rdf: " + rdf);
}
return conceptUri;
}
// When utilizing the getTermExpansion method, will get a list of URIs back
// and not just one URI
protected List<String> getConceptURIsListFromRDF(String rdf) {
List<String> conceptUris = new ArrayList<String>();
try {
Document doc = XMLUtils.parse(rdf);
NodeList nodes = doc.getElementsByTagName("skos:Concept");
int numberNodes = nodes.getLength();
int n;
for (n = 0; n < numberNodes; n++) {
Node node = nodes.item(n);
NamedNodeMap attrs = node.getAttributes();
Attr idAttr = (Attr) attrs.getNamedItem("rdf:about");
String conceptUri = idAttr.getTextContent();
conceptUris.add(conceptUri);
}

} catch (IOException e) {
```

```
e.printStackTrace();
System.err.println("rdf: " + rdf);
} catch (SAXException e) {
e.printStackTrace();
System.err.println("rdf: " + rdf);
} catch (ParserConfigurationException e) {
e.printStackTrace();
System.err.println("rdf: " + rdf);
}
return conceptUris;
}
protected String getDbpediaDescription(String uri) throws Exception {
String descriptionSource = " (Source: DBpedia)";
String description = new String();
String qs = ""
+ "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> \n"
+ "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> \n"
+ "PREFIX foaf: <http://xmlns.com/foaf/0.1/> \n"
+ "PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>\n"
+ "SELECT DISTINCT ?description WHERE { \n" + "<" + uri
+ "> rdfs:comment ?description . \n"
+ "FILTER (LANG(?description)='en' ) \n" + "}";
// System.out.println(qs);
List<HashMap> resultList = new ArrayList<HashMap>();
QueryExecution qexec = null;
try {
Query query = QueryFactory.create(qs);
qexec = QueryExecutionFactory.sparqlService(this.dbpedia_endpoint, query);
qexec.setTimeout(5000, TimeUnit.MILLISECONDS);
resultList = new ArrayList<HashMap>();
ResultSet resultSet = qexec.execSelect();
int resultSetSize = 0;
while (resultSet.hasNext()) {
resultSetSize++;
QuerySolution solution = resultSet.nextSolution();
Iterator varnames = solution.varNames();
HashMap<String, String> hm = new HashMap<String, String>();
while (varnames.hasNext()) {
String name = (String) varnames.next();
RDFNode rdfnode = solution.get(name);
// logger.info("rdf node name, type: "+ name
// +", "+getRDFNodeType(rdfnode));
if (rdfnode.isLiteral()) {
Literal literal = rdfnode.asLiteral();
String nodeval = literal.getString();
hm.put(name, nodeval);
} else if (rdfnode.isResource()) {
Resource resource = rdfnode.asResource();
String nodeval = resource.toString();
hm.put(name, nodeval);
}
}
resultList.add(hm);
}
description = "";
for (HashMap map : resultList) {
if (map.containsKey("description")) {
description = (String) map.get("description");
}
}
} catch (Exception ex) {
throw ex;
}
// Adding source so it is clear that this description comes from DBPedia
return description + descriptionSource;
}
/**
* @param uri The URI
*/
protected String stripConceptId(String uri) {
String conceptId = new String();
```

```
int lastslash = uri.lastIndexOf('/');
conceptId = uri.substring(lastslash + 1, uri.length());
return conceptId;
}
/**
* @param str The String
*/
protected String extractConceptId(String str) {
try {
return str.substring(1, str.length() - 1);
} catch (Exception ex) {
return "";
}
}
/**
* The code here utilizes the SKOSMOS REST API for Agrovoc
* This returns JSON LD so we would parse JSON instead of RDF
* The code above can still be utilized if we need to employ the web services directly
*/
//Get search results for a particular term and language code
private String getSKOSMosSearchResults(String term, String lang) {
String urlEncodedTerm = URLEncoder.encode(term);
//Utilize 'starts with' using the * operator at the end
String searchUrlString = this.conceptsSkosMosSearch + "query=" + urlEncodedTerm + "*" + "&vocab="+
ontologyName +"&lang=" + lang;
System.out.println(searchUrlString);
URL searchURL = null;
try {
searchURL = new URL(searchUrlString);
} catch (Exception e) {
logger.error("Exception occurred in instantiating URL for "
+ searchUrlString, e);
// If the url is having trouble, just return null for the concept
return null;
}


String results = null;
try {
StringWriter sw = new StringWriter();
BufferedReader in = new BufferedReader(new InputStreamReader(
searchURL.openStream()));
String inputLine;
while ((inputLine = in.readLine()) != null) {
sw.write(inputLine);
}
in.close();
results = sw.toString();
System.out.println(results);
logger.debug(results);
} catch (Exception ex) {
logger.error("Error occurred in getting concept from the URL "
+ searchUrlString, ex);
return null;
}
return results;

}

//JSON-LD array
private List<String> getConceptURIsListFromSkosMosResult(String results) {
List<String> conceptURIs = new ArrayList<String>();
ObjectNode json = (ObjectNode) JacksonUtils.parseJson(results);
//Format should be: { ..."results":["uri":uri...]
if (json.has("results")) {
ArrayNode jsonArray = (ArrayNode) json.get("results");
int numberResults = jsonArray.size();
int i;
for(i = 0; i < numberResults; i++) {
ObjectNode jsonObject = (ObjectNode) jsonArray.get(i);
if(jsonObject.has("uri")) {
```

```
conceptURIs.add(jsonObject.get("uri").asText());
}
}
}
return conceptURIs;
}

//Utilize this for getting preffered label of the concept
//JSON-LD array
private String getConceptPrefLabelFromSkosMosResult(String results) {
String getPrefLabel ="";

ObjectNode json = (ObjectNode) JacksonUtils.parseJson(results);
//Format should be: { ..."results":["uri":uri...]
if (json.has("results")) {
ArrayNode jsonArray = (ArrayNode) json.get("results");
int numberResults = jsonArray.size();
int i;
for(i = 0; i < numberResults; i++) {
ObjectNode jsonObject = (ObjectNode) jsonArray.get(i);

if(jsonObject.has("prefLabel")) {

System.out.println(jsonObject.get("prefLabel").asText());
getPrefLabel = jsonObject.get("prefLabel").asText();
}

}
}
return getPrefLabel;
}

//Utilize this for getting the notation of the concept, if given
private String getConceptNotationFromSkosMosResult(String results) {
String getNotation ="";

ObjectNode json = (ObjectNode) JacksonUtils.parseJson(results);
//Format should be: { ..."results":["uri":uri...]
if (json.has("results")) {
ArrayNode jsonArray = (ArrayNode) json.get("results");
int numberResults = jsonArray.size();
int i;
for(i = 0; i < numberResults; i++) {
ObjectNode jsonObject = (ObjectNode) jsonArray.get(i);

if(jsonObject.has("notation")) {

System.out.println(jsonObject.get("notation").asText());
getNotation = jsonObject.get("notation").asText();
}

}
}
return getNotation;
}


}
```

Adjusting the file ConceptSearchService.java

This file containes all of the vocabularies services defined **: VIVO/api/src/main/java/edu/cornell/mannlib/vitro/webapp/utils /ConceptSearchService .**
In order to add a new service, the file has to be adjusted as follows:

ConceptSearchService.java

```
/* $This file is distributed under the terms of the license in LICENSE$ */
```

```
package edu.cornell.mannlib.vitro.webapp.utils.ConceptSearchService;
import java.util.HashMap;
import java.util.List;
import javax.servlet.ServletContext;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import edu.cornell.mannlib.semservices.bo.Concept;
import edu.cornell.mannlib.semservices.service.ExternalConceptService;
import edu.cornell.mannlib.vitro.webapp.controller.VitroRequest;
/**
 * Utilities for search
 */
public class ConceptSearchServiceUtils {
 private static final Log log = LogFactory.getLog(ConceptSearchServiceUtils.class);
 //Get the appropriate search service class
 //TODO: Change this so retrieved from the system instead using a query

 private static final String UMLSVocabSource = "http://link.informatics.stonybrook.edu/umls";
 private static final String AgrovocVocabSource = "http://aims.fao.org/aos/agrovoc/agrovocScheme";
 private static final String GemetVocabSource = "http://www.eionet.europa.eu/gemet/gemetThesaurus";
 private static final String LCSHVocabSource = "http://id.loc.gov/authorities/subjects";
//We define the new vocabulary source here. Exchange SKOSMOS_PATH with your local skosmos url.
private static final String FaechersystematikVocabSource = "SKOSMOS_PATH/faechersystematik/data";
//Get the class that corresponds to the appropriate search
public static String getConceptSearchServiceClassName(String searchServiceName) {
 HashMap<String, String> map = getMapping();
 if(map.containsKey(searchServiceName)) {
 return map.get(searchServiceName);
 }
 return null;
 }

 //Get the URLS for the different services
 //URL to label
 public static HashMap<String, VocabSourceDescription> getVocabSources() {
 HashMap<String, VocabSourceDescription> map = new HashMap<String, VocabSourceDescription>();
 map.put(UMLSVocabSource, new VocabSourceDescription("UMLS", UMLSVocabSource, "http://www.nlm.nih.gov
/research/umls/", "Unified Medical Language System"));

 //Commenting out agrovoc for now until implementation is updated
 map.put(AgrovocVocabSource, new VocabSourceDescription("AGROVOC", AgrovocVocabSource, "http://www.fao.org
/agrovoc/", "Agricultural Vocabulary"));
//Define the abbreviation, the URL and the full title of the new source here
map.put(FaechersystematikVocabSource, new VocabSourceDescription("Faechersystematik",
FaechersystematikVocabSource, "https://labs.tib.eu/info/", "Faechersystematik of the German Federal Office
of Statistics, located at Skosmos by TIB"));
map.put(GemetVocabSource, new VocabSourceDescription("GEMET", GemetVocabSource, "http://www.eionet.europa.eu
/gemet", "General Multilingual Environmental Thesaurus"));
 map.put(LCSHVocabSource, new VocabSourceDescription("LCSH", LCSHVocabSource, "http://id.loc.gov/authorities
/subjects/", "Library of Congress Subject Headings"));


return map;
 }

 //Get additional vocab source info


 //Get the hashmap mapping service name to Service class
 private static HashMap<String, String> getMapping() {
 HashMap<String, String> map = new HashMap<String, String>();
map.put(UMLSVocabSource, "edu.cornell.mannlib.semservices.service.impl.UMLSService");
map.put(AgrovocVocabSource, "edu.cornell.mannlib.semservices.service.impl.AgrovocService");
//We assign the new java class FaechersystematikService to the new vocabulary source
 map.put(FaechersystematikVocabSource, "edu.cornell.mannlib.semservices.service.impl.
FaechersystematikService");
map.put(GemetVocabSource, "edu.cornell.mannlib.semservices.service.impl.GemetService");
 map.put(LCSHVocabSource, "edu.cornell.mannlib.semservices.service.impl.LCSHService");
return map;
 }
```

```
 public static List<Concept> getSearchResults(ServletContext context, VitroRequest vreq) throws Exception {
 String searchServiceName = getSearchServiceUri(vreq);
 System.out.println(searchServiceName);
 String searchServiceClassName = getConceptSearchServiceClassName(searchServiceName);
 System.out.println(searchServiceClassName);

 ExternalConceptService conceptServiceClass = null;

 Object object = null;
 try {
 Class classDefinition = Class.forName(searchServiceClassName);
 object = classDefinition.newInstance();
 conceptServiceClass = (ExternalConceptService) object;
 } catch (InstantiationException e) {
 System.out.println(e);
 } catch (IllegalAccessException e) {
 System.out.println(e);
 } catch (ClassNotFoundException e) {
 System.out.println(e);
 }

 if(conceptServiceClass == null){
 log.error("could not find Concept Search Class for " + searchServiceName);
 System.out.println("getSearchResults : searchServiceName | "+searchServiceName );
 return null;

 }

 //Get search
 String searchTerm = getSearchTerm(vreq);
 System.out.println("getSearchResults : getSearchResults | "+searchTerm );
 log.error(searchTerm);

 List<Concept> conceptResults = conceptServiceClass.getConcepts(searchTerm);



 return conceptResults;
 }

 private static String getSearchServiceUri(VitroRequest vreq) {
 System.out.println(vreq);

 return vreq.getParameter("source");

 }
private static String getSearchTerm(VitroRequest vreq) {
 return vreq.getParameter("searchTerm");
 }
}
```

After steps 1 and 2 the application must be rebuilt.

Defining the new vocabulary source in the data directory

In this file (**It vivo/data/.../rdf/abox/filegraph/vocabularySource.n3)** the new vocabulary service has to be be defined as an owl:Thing, and it receives a label**:**

vocabularySource.n3

```
<http://link.informatics.stonybrook.edu/umls> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.
w3.org/2002/07/owl#Thing> .
<http://aims.fao.org/aos/agrovoc/agrovocScheme> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2002/07/owl#Thing> .
<http://www.eionet.europa.eu/gemet/gemetThesaurus> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2002/07/owl#Thing> .
<http://id.loc.gov/authorities/subjects> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org
/2002/07/owl#Thing> .
//Defining the new vocabulary as an instance of owl:Thing
<https://labs.tib.eu/skosmos/faechersystematik/data> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2002/07/owl#Thing> .

<http://link.informatics.stonybrook.edu/umls> <http://www.w3.org/2000/01/rdf-schema#label> "UMLS"
^^<http://www.w3.org/2001/XMLSchema#string> .
<http://id.loc.gov/authorities/subjects> <http://www.w3.org/2000/01/rdf-schema#label> "LCSH"^^<http://www.w3.
org/2001/XMLSchema#string> .
<http://aims.fao.org/aos/agrovoc/agrovocScheme> <http://www.w3.org/2000/01/rdf-schema#label> "AGROVOC"
^^<http://www.w3.org/2001/XMLSchema#string> .
<http://www.eionet.europa.eu/gemet/gemetThesaurus> <http://www.w3.org/2000/01/rdf-schema#label> "GEMET"
^^<http://www.w3.org/2001/XMLSchema#string> .
//Providing a label
<https://labs.tib.eu/skosmos/faechersystematik/data> <http://www.w3.org/2000/01/rdf-schema#label>
"Faechersystematik"^^<http://www.w3.org/2001/XMLSchema#string> .
```

While rebuilding the application and restarting the Tomcat Server this file is going to be overwritten, so that the edits can get lost. Therefore make sure, that the edits are in the file after each rebuild and restart.