iQvoc – Open Source SKOS(XL) Maintenance and Publishing Tool

Thomas Bandholtz[1], Till Schulte-Coerne[1], Robert Glaser[1], Joachim Fock[2], Tim Keller[1]

[1] innoQ Deutschland GmbH, Halskestr. 17, 40880 Ratingen, Germany
{thomas.bandholtz|till.schulte-coerne|robert.glaser|tim.keller}@innoq.com
[2] Federal Environment Agency, Wörlitzer Platz 1, 06844 Dessau, Germany
joachim.fock@uba.de

Abstract. iQvoc is a new open source SKOS-XL vocabulary management tool developed by the Federal Environment Agency, Germany, and innoQ Deutschland GmbH. Its immediate purpose is maintaining and publishing reference vocabularies in the upcoming Linked Data cloud of environmental information, but it may be easily adapted to host any SKOS-XL compliant vocabulary. iQvoc is implemented as a Ruby on Rails application running on top of JRuby – the Java implementation of the Ruby Programming Language. To increase the user experience when editing content, iQvoc uses heavily the JavaScript library jQuery.

Keywords: SKOS, Thesauri, Cool URIs, Linked Data, Editorial, Open Source, Ruby, jQuery.

1 Introduction

When governmental authorities start publishing Linked Data[1], they need to draw some border lines within the open world. Authorities want to streamline legal publishing obligations, but they have to take care about provenance and trust in a way that every citizen can make sure that this data has been published by a governmental authority.

For such reasons, authorities publish controlled reference vocabularies themselves rather than linking their data to publicly maintained vocabularies such as dbPedia[2] or Geonames[3]. iQvoc has been built by the Federal Environment Agency, Germany[4], and innoQ Deutschland GmbH[5] to serve such controlled reference vocabularies through their life cycle (Clarify -> Agree -> Formalize -> Publish -> Reference).

[1] http://linkeddata.org/
[2] http://dbpedia.org
[3] http://www.geonames.org
[4] http://umweltbundesamt.de/
[5] http://innoq.com

In order to achieve the best development productivity in the iQvoc project, the decision was to use Ruby and jQuery as the basic development tools.

iQvoc is available with a European Public License (EUPL[6]).

2 Reference Vocabulary Management

iQvoc has been developed as a management tool for reference vocabularies in knowledge organization systems (KOS) [1] on the Web. Reference vocabularies provide definitions and structure for a set of preferred keywords designating what the knowledge contributions (documents or data) are talking about. The traditional pattern for the usage of such a vocabulary is thesaurus-based indexing in libraries. Today, KOS are typically accessible via the Web, although some contributions may still be available in paper form only. Since years, more and more content has been published in the Web in form of digital documents, Web pages, or Linked Data, so that the content items can be linked to the respective keywords by URI references.

In contrast to folksonomies, reference vocabularies are controlled by an editorial team. While any user may post her proposals, the editorial team will decide.

Usually, reference vocabularies are quite large and thoroughly organized in concept hierarchies and cross-references, and many of them add a multitude of synonyms and vernacular terms around the preferred concept labels.
Some established examples from the environment domain are:

- GEneral Multilingual Environment Thesaurus (GEMET)[7], and
- EUNIS biodiversity database[8], both maintained by the European Environment Agency;
- Environmental Applications Reference Thesaurus (EARTh)[9] from Italy;
- Environmental Thesaurus UMTHES, from Germany[10].

2.1 Requirements and Existing Tools

Important functional requirements of the Agency have been:

- SKOS-XL compliance of the model
- Web interface for browsing and navigation
- Multilingualism
- Comfortable editing features with validation
- Editorial team and workflow support
- Linked Data support

These requirements are discussed more closely in the following sub-sections.

---

[6] http://ec.europa.eu/idabc/eupl
[7] http://www.eionet.europa.eu/gemet
[8] http://eunis.eea.europa.eu/
[9] http://uta.iia.cnr.it/earth_eng.htm
[10] http://www.semantic-network.de

The most important non-functional requirement has been Open Source availability of the tool.

After investigating and comparing existing tools such as Pool Party[11], MMI Ontology Registry and Repository[12], Neologism[13], OntoWiki[14], and an early demo version of iQvoc[15], iQvoc turned out to be the closest match. None of these tools met all the requirements out of the box, but iQvoc represented the closest strategic consensus between the Agency and the potential development partner. iQvoc had already proved to work with the above mentioned GEMET multilingual GEMET vocabulary in SKOS on the Web, and there was a Linked Data commitment. Furthermore, innoQ shared the strategic focus on serving lexical complexity (SKOS-XL) and a controlled editorial workflow.

2.2 SKOS-XL Compliance

The *Simple Knowledge Organization System* (SKOS) W3C Recommendation [2] provides a RDFS/OWL schema for "thesauri, classification schemes, subject heading lists and taxonomies within the framework of the Semantic Web". The SKOS Core model defines basic classes such as Concept and ConceptScheme and simple properties (such as skos:prefLabel, skos:altLabel, skos:note) and semantic relations (broader, narrower, related). The recommendation also contains a *SKOS Extension for Labels* (SKOS-XL). XL is needed in order to express lexical complexity of the labels such as inflectional forms or term composition [3], which is a required feature for any kind of natural language processing and widely supported by the environmental thesaurus of the Agency.

While SKOS labels are simple annotation properties (sub-properties of rdfs:label) with a literal value, SKOS-XL labels are classes which can be linked to concepts and among each other using object properties. Consequently the SKOS-XL version of prefLabel has not a literal as its range, but an instance of the skosxl:Label class, like in the following example:

```
# SKOS
:4711 a skos:Concept;
  skos:prefLabel "waste"@en;
  skos:altLabel  "garbage"@en.

# corresponding statements in SKOSXL
:4711 a skos:Concept;
  skosxl:prefLabel :4712;
  skosxl:altLabel  :4713.
```

[11] http://www.enterprise-20.at/produkte/poolparty/
[12] http://mmisw.org/orr/
[13] http://neologism.deri.ie/
[14] http://ontowiki.net/Projects/OntoWiki
[15] http://apps.innoq.com/iqvoc/about.html

```
:4712 a skosxl:Label;
skosxl:literalForm "waste"@en.

:4713 a skosxl:Label;
skosxl:literalForm "garbage"@en.
```

In SKOS-XL, the literal value has moved to the skosxl:Label instance and appears as the value of its skosxl:literalForm property. In order to make this compatible with a simple SKOS representation of the label, the Recommendation defines property chains such as "The property chain (skosxl:prefLabel, skosxl:literalForm) is a sub-property of skos:prefLabel." [2, S55]. Based on this generic model, applications may create extensions to express inflectional forms of the literal form, term compositions, abbreviated forms, or spelling variants.

2.3 Web Interface for Browsing and Navigation

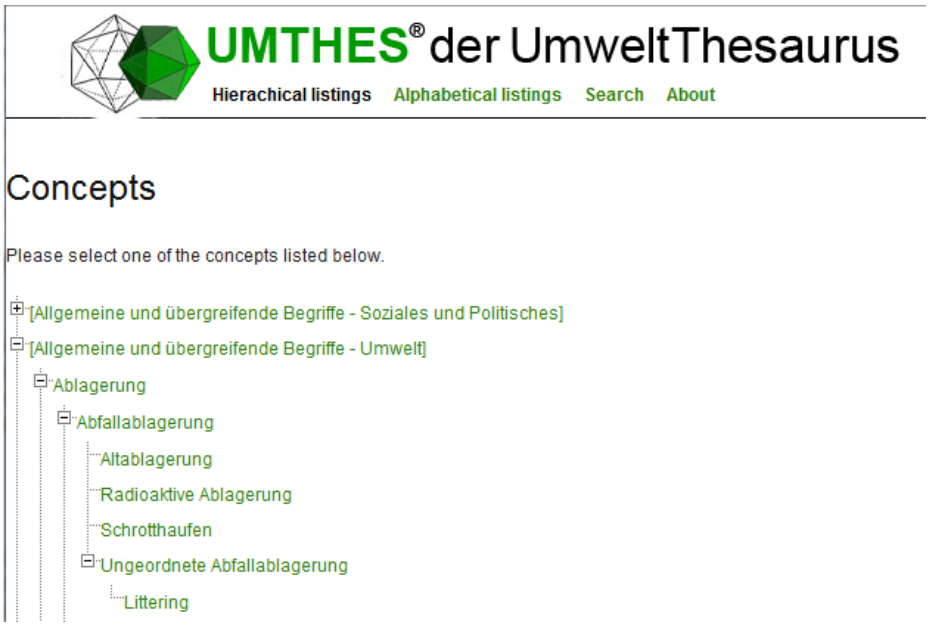

Fig. 1. Expandable hierarchical view

The vocabulary can be accessed and navigated in multiple forms:

- the hierarchy view starts with a list of top concepts from which the concept hierarchy may be explored level by level;
- the alphabetic view provides a sorted list of labels;
- the search dialog with comprehensive search options;
- resolving a URI reference pointing to a single concept or label.

- dedicated Web pages for concept and label instances in which cross-references are implemented as hyperlinks.

Most of these are rather simple features and implemented by many tools, apart from the property chain pattern (see previous subchapter). In the concept view, iQvoc displays the literalFom values of the related skosxl:Label instances, which is what human readers are expecting, but a generic RDF or specialized simple SKOS tool would never behave like that. They would treat skosxl:Label instances lust like any class instance which has no rdfs:label annotation property.

2.4 Multilingualism

Most of the reference vocabularies are multilingual (GEMET concepts have labels in 28 languages). The first demo version of iQvoc (2008) hosted GEMET as test data, supporting concept labels in all these 28 languages. UMTHES is German-centric with English translations.

iQvoc supports both symmetrical and asymmetrical multilingualism. In the symmetrical approach, each concept has preferred labels in any of the supported languages (like in GEMET), while the asymmetrical approach has language-specific concepts with translation relations. The German-centric approach of UMTHES implies that there must be exactly one German preferred label for each concept. English labels are added as altLabels only. Explicit translation relations may be asserted to Label instances. While these different approaches need no specialized implementation for browsing-only applications, support of the selected pattern is a requirement for the editing features (see below).

A different aspect is the internationalization of the user interface. Currently iQvoc is configured bilingual German/English, but more languages can be added simply by providing a YAML file with the specific interface terms.

2.5 Comfortable editing features with validation

The most challenging pattern of the editing features is managing valid references from huge lists of concepts or labels in many semantically different relations.

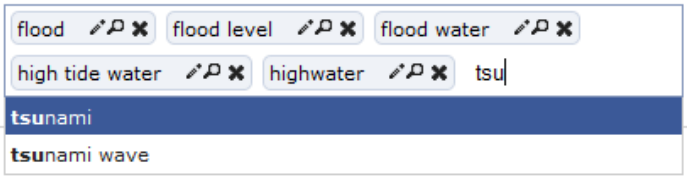

Fig. 2. Reference Widget

Fig. 2 shows an example dialog for the selection of English altLabels for the German concept “Hochwasser” (flood). Each box in this canvas represents a single

target of this relation (in SKOS-XL: the literalForm value of the related label). The icons in each box mean (from left to right):

- open the concept display of this target for viewing in a new browser tab;
- open the concept display of this target for editing in a new browser tab;
- remove this reference from the canvas.

At the end of the target list, you can see someone has started typing the label of a further target ("tsu …"). In this moment, a dynamic selection list opens showing all the available valid targets starting with the characters as typed. If the intended target ("tsunami") would not exist, it can be created on the fly in a second browser tab or window.

All target lists are pre-validated according to the semantics of the respective relation, so that it is not possible to make invalid selections in this widget. However, there might be collisions with cardinality restrictions. For example, each concept must have exactly one preferred label in German, and the composition of a compound must have at least two components. Such constraints are checked on demand at any time, and consistency check is forced before final release. However, for convenience of the editor, the state of work may be saved in any state of inconsistency or incompleteness but remaining invisible in the production.

2.6 Editorial team and workflow support

While the public user does not need any account for browsing or dereferencing the released data, there are four roles with ascending rights in the editorial team:

- *guest* – has read access to unreleased versions and internal editorial notes;
- *edit* – can edit concepts, labels, and their relations;
- *release* – can release new versions after they have been edited;
- *admin* – can create accounts and assign roles to users.

These roles already give an idea of the implemented workflow:

- *checked out* – create a copy of an item for editing. This copy will stay locked to the individual editor, but he (or the higher roles) can unlock the record so that someone else can take over;
- *submit* – from the point of view of the single editor, the copy may be released now.
- *release* – the new version passes into production after a consistency check. The copy can also be rejected or dismissed before release.

Intentionally, there is no way to delete a concept after it has been released. In order to provide sustainable references ("Cool URIs don't change"), deprecated concepts are annotated instead ("expired").

The overall state of editing is displayed in a dashboard from which the editorial team may pick up an item. This dashboard supports team planning in order to provide some coordination. In addition, when a dataset gets checked out, iQvoc looks up if any directly linked instance is currently being edited, and in this case displays a warning.

2.7 Linked Data support

As more and more data gets published as Linked Data, iQvoc needs "cool URIs" in order to be linked with this data. Currently, the Federal Environment Agency is moving towards Linked Environment Data[16], and iQvoc will be used to maintain the reference vocabularies. So iQvoc needs to implement Linked Data technology as described in [4]. Content negotiation follows the "303 URIs forwarding to Different Documents" [5] pattern. IQvoc renders concepts or labels in Turtle syntax[17] like in the following example.

```
:00028876 a skos:Concept;
          skosxl:prefLabel :Hochwasser;
          skosxl:altLabel :Flusshochwasser, :Flut--Hochwasser,
:Flutereignis, :Flutkatastrophe, :Fruehjahrshochwasser,
:HochwasserEinesFlusses, :Hochwasserereignis, :Hochwasserganglinie,
:Hochwasserganglinienvorhersage, :Hochwasserkatastrophe,
:Hochwasserrisiko, :Oderhochwasser, :Winterhochwasser, :flood,
:floodLevel, :floodWater, :highTideWater, :highwater;
          skos:broader :_00027345, :_00028887, :_00650524;
          skos:narrower :_00012775, :_00012789, :_00012791,
:_00651102;
          skos:related :_00012793, :_00029767, :_00655642;
          umt:exportNote [
    umt:source "<aDisBMS>";
    umt:thsisn "\"00028876\"";
    dct:date "\"2010-04-29\""
];
          umt:changeNote [
    umt:editor "<nn>";
    dct:modified "\"2008-11-21\""
];
          umt:changeNote [
    umt:editor "<nn>";
    dct:created "\"1991-01-15\""
];
          umt:sourceNote "GEMETID3298"@de;
          skos:closeMatch gemet:3298, gemet:3218;
          skos:classified "WA60", "NL10";
          skos:status "x".
```

Linked Data access is implemented by Graph Store integration (see section 3.4).

[16] http://www.w3.org/egov/wiki/Linked_Environment_Data
[17] http://www.w3.org/TeamSubmission/turtle/

Furthermore, there had been plans for some basic assistance for data matching by comparing names in batch mode and write a list of possibly matching terms. We finally dropped this approach as we found that the *Linking Framework for the Web of Data* (Silk) [6] is a specialized Open Source tool in this area which can be configured for SKOS-to-SKOS matching more conveniently.

3 Technologies

3.1 JRuby and Ruby on Rails

JRuby[18] is the Java-Implementation of Ruby. Most of the Ruby libraries work with JRuby unless the library has native extensions. With JRuby it is possible to run Ruby on Rails and hence iQvoc directly on top of the Java Virtual Machine.

Ruby on Rails[19] is a Ruby Web framework and organized around the Model-View-Controller (MVC) pattern. The core principles are:

- Don't Repeat Yourself (DRY)
- Convention Over Configuration

Whereas Ruby and Ruby on Rails work very well with relational databases (“Active Records”), the same cannot be said about Ruby and RDF. The initial RubyRDF package was an experimental system and is no longer under active development. Other approaches, such as ActiveRDF or Redland's Ruby interface are not officially abandoned but they lack a certain degree of maturity. A deeper discussion can be found in [7].

Considering the priority of a rapid transition into production, iQvoc internally uses a relational representation of the SKOS-XL model with an external RDF representation and SPARQL endpoint which are provided by Triple Store integration (see section 3.4).

3.2 RDF Rendering

For RDF Rendering we introduced a new Ruby library, iQrdf. It contains an internal DSL inspired by the popular Builder[20] DSL for XML markup. We plan to release this library separately under a EUPL license after testing and stabilizing it in our internal projects.

We designed the DSL by complying with the principles of the Turtle Syntax. On one hand, iQRdf supports abbreviation of triple groups. On the other hand, it does not

[18] http://jruby.org/
[19] http://guides.rubyonrails.org/getting_started.html
[20] http://builder.rubyforge.org/

support nested triple definitions (Subject Predicate AnotherSubject Predicate ...) except of Blank Nodes.

Unlike some other Ruby libraries (e.g. DataGraph[21]), another basic principle is the object oriented interpretation of triples. We interpret a triple (subject, predicate, object) as the call of a method (the predicate) on the subject with the object given as parameter: Subject.Predicate(Object). Namespaces can be addressed by using Ruby Modules: Subject.Namespace::Predicate(Object). Namespaces can be defined by simply assigning a URI to a prefix symbol.

To achieve this we make heavy use of Ruby's *method_missing* feature. This method, initially defined in the Object class, is called each time a method call could not be resolved due to the lack of a matching method. The default method_missing implementation (raising a MethodNotFound Exception) can be overwritten by the application.

Some examples:

```
doc = IqRdf::Document.new('http://iqvoc.de/concepts')
doc.namespace :skos => 'http://www.w3.org/2008/05/skos#'
doc << IqRdf::foo.Rdf::type(IqRdf::Skos::Concept)
puts doc.to_turtle
```

This would result in the following Turtle RDF:

```
@prefix : <http://iqvoc.de/concepts>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix skos: <http://www.w3.org/2008/05/skos#>.
:foo rdf:type skos:Concept.
```

RDF graphs of a single subject with multiple predicates or multiple objects of the same predicate can be rendered by using blocks respectively multiple method parameters:

```
…
doc << IqRdf::foo do |my_foo|
  my_foo.Skos::prefLabel('Foo')
  my_foo.Skos::related(IqRdf::bar, IqRdf::fubar)
end
```

This would result in:

```
…
:foo skos:prefLabel "Foo";
     skos:related :bar, :fubar.
```

IQrdf can also annotate strings with language tags:

```
…
doc << IqRdf::foo(:lang => :en) do |my_foo|
  my_foo.string1('String1')
```

[21] http://blog.datagraph.org/2010/03/rdf-for-ruby

```
    my_foo.string2('String2', :lang => :de)
    my_foo.string3('String3 "with quotes"', :lang => :none)
  end
```

The Result:

```
  …
  :foo :string1 "String1"@en;
       :string2 "String2"@de;
       :string3 "String3 \"with quotes\"".
```

rdf:lists are rendered by providing an array as method parameter:

```
  …
  doc << IqRdf::foo.myList(['a', 'b', 'c'])
```

With the result:

```
  …
  :foo :myList ("a" "b" "c").
```

3.3 jQuery

jQuery is a JavaScript library that "...simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development"[22]. iQvoc uses jQuery heavily in editing mode to increase user experience. For example, iQvoc supports comfortable date pickers, adding notes dynamically, and the widget for managing cross references already mentioned in section 2.5.

To hide the complexity of the relation editing process, iQvoc uses a tagging style approach. With this approach, it is quite simple for a user to add and remove relations to other concepts or labels. The relation widget is an extension of the "Tokenizing Autocomplete Text Entry" jQuery-Plugin[23].

The following plugin features are important for iQvoc:

- Autocompletion – the simplest way to handle large data sets
- Pre-populate lists – for already added relations.
- Clean and Simple Interface

With some extensions, it is now possible to add and remove relations on the fly. For example, if a user adds a relation an AJAX request is send from the browser to the server, and the relation will be added in the database. If an error occurs, the user will be informed by an error message.

[22] http://jquery.com/
[23] http://www.jqueryplugins.com/plugin/235/

3.4 Graph Store Integration

In the application context of the Environment Agency, we intend to use the Virtuoso Open Source Edition[24] as a triple store proxy for several applications that take part in Linked Environment Data. Each of these information systems simply needs to be enabled to render its data in RDF and synchronize the proxy whenever something changes. In this architecture, content negotiation, RDF dereferencing and a SPARQL endpoint are provided as shared services by the graph store rather than by redundant implementations of these features in each system. As a most welcome side effect, all participating databases will share a single SPARQL endpoint, so there is no need for any SPARQL federation within this network.

This architecture also overcomes the Turtle syntax limitation of the built-in RDF rendering of iQvoc (iQrdf, see sub-section 3.2). iQrdf is only used for synchronization with the triple store, but the open Linked Data URI dereferencing is provided by the triple store itself independently. This is why N-Tripples or RDF/XML are also available for public access.

4 Lessons Learned and Future Development

Given such in part non-trivial requirements, Ruby and JQuery have proved themselves as excellent scripting languages. One big drawback of Ruby has been described in section 3.1: the lack of some high quality RDF store integration which could be compared to the relational ActiveRecord package. As we did not have resources for such a development within the described project, we selected a relational data persistence layer of the editorial data.

This of course reduces flexibility. Looking forward to Linked Environment Data, we will need a tool to support different reference vocabularies which are not expressed in the SKOS or SKOS-XL schema, such as a species catalogue, a gazetteer and a chronicle. These three (and possibly more) cases should bundle their resources to extend the iQvoc flexibility with regard to different schemas.

With regard to the user interface, the editing features have intentionally been implemented with a trained expert team in mind. Some aspects may not appear self-explanatory for the general user. If there is demand for a more intuitive interface, this certainly needs revisiting the navigation design, but it does not touch any limits of the scripting languages.

Demo Access

iQvoc is still in the final testing phase. The production release is scheduled to go live by July 2010. A demo environment for the SFSW2010 workshop is temporarily deployed at http://iqvoc-eswc.dyndns.info.

[24] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/

References

1. "What is Knowledge Organization", a special issue of Knowledge Organization, Vol. 35 (2008) No.2-3.
2. SKOS Simple Knowledge Organization System. Reference. W3C Recommendation 18 August 2009. http://www.w3.org/TR/2009/REC-skos-reference-20090818/
3. Bandholtz, T.: Expressing Lexical Complexity in SKOS(XL). 5th ECOTERM MEETING at FAO, Rome, Italy, 05-06 October 2009. ecoterm09-Bandholtz.ppt at http://eea.eionet.europa.eu/Public/irc/envirowindows/jad/library?l=/ecoinformatics_indicator/ecoterm_5-6102009
4. Bizer, C et al.: How to Publish Linked Data on the Web. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/
5. Cool URIs for the Semantic Web. W3C Interest Group Note 03 December 2008. http://www.w3.org/TR/cooluris/
6. Volz, J. et al.: Silk – A Link Discovery Framework for the Web of Data. 2nd Workshop about Linked Data on the Web (LDOW2009), Madrid, Spain, April 2009. http://events.linkeddata.org/ldow2009/papers/ldow2009_paper13.pdf
7. Mainz, D.: Deep Integration of the OWL Ontology Language into Ruby Using Metaprogramming. Dissertation. Düsseldorf 2008. http://docserv.uni-duesseldorf.de/servlets/DerivateServlet/Derivate-10799/DominicMainz.pdf